Online Learning With Adaptive Rebalancing in Nonstationary Environments

Kleanthis Malialis, Christos G. Panayiotou, *Senior Member, IEEE*, and Marios M. Polycarpou, *Fellow, IEEE*

***Abstract*— An enormous and ever-growing volume of data is nowadays becoming available in a sequential fashion in various real-world applications. Learning in nonstationary environments constitutes a major challenge, and this problem becomes orders of magnitude more complex in the presence of class imbalance. We provide new insights into learning from nonstationary and imbalanced data in online learning, a largely unexplored area. We propose the novel Adaptive REBAlancing (AREBA) algorithm that selectively includes in the training set a subset of the majority and minority examples that appeared so far, while at its heart lies an adaptive mechanism to continually maintain the class balance between the selected examples. We compare AREBA with strong baselines and other state-of-the-art algorithms and perform extensive experimental work in scenarios with various class imbalance rates and different concept drift types on both synthetic and real-world data. AREBA significantly outperforms the rest with respect to both learning speed and learning quality. Our code is made publicly available to the scientific community.**

***Index Terms*— Class imbalance, concept drift, neural networks, nonstationary environments, online learning.**

I. Introduction

EFFICIENT and effective analysis methods for the ever-increasing volume of sequential data in a wide range of applications are of paramount importance. In practical applications, data are evolving or drifting over time, i.e., data are drawn from nonstationary distributions. Various factors can trigger a nonstationarity effect or concept drift, for example, seasonality or periodicity effects, changes in users' habits, interests or preferences, and hardware or software faults [1]. Learning in nonstationary environments constitutes a major challenge. In such environments, a classifier with learning capabilities is of vital importance as it will provide an adaptive behavior and help maintain optimal performance. The problem becomes significantly more complex if class imbalance coexists with concept drift. In this case, class imbalance refers to sequential data that have skewed distributions and is a difficult problem as it causes a traditional learning algorithm to be ineffective because of its poor generalization ability and its weak prediction power for the minority class examples [2].

Learning from nonstationary and imbalanced data has been studied separately, but several key challenges remain open when the joint problem is considered. The majority of existing works focus on batch (or chunk-by-chunk) learning, i.e., when examples arrive in batches (or chunks). In this article, we address the combined challenges of drift and imbalance in online (or one-by-one) learning, i.e., when a single example arrives at each step. The design of batch learning algorithms differs significantly from that of online learning, and therefore, the majority are typically unsuitable for online learning tasks [3]. Addressing these key challenges can have a significant impact on various applications areas, e.g., in critical infrastructure systems, smart buildings, finance and banking, security and crime, healthcare, and environmental sciences [3]–[6].

The desired properties of an online classifier learning from nonstationary and imbalanced data are as follows [4], [7].

1) *Learning New Knowledge:* The classifier should learn novel knowledge as new data are arriving.
2) *Preserving Previous Knowledge:* Being able to preserve previous knowledge relies on the ability of the classifier to determine what previous knowledge is still relevant (and hence to preserve it) and what has now become irrelevant (and hence to discard or "forget" it).
3) *High Performance:* The classifier should obtain high performance on both the majority and minority classes.
4) *Fast Operation:* The classifier should operate in less than the example (or batch) arrival time.
5) *Fixed Storage:* The classifier should use no more than a fixed amount of memory for any storage; ideally, it should be capable of incremental learning, i.e., when learning occurs on a single instance (or batch) without considering (and hence storing) previous data.

Balancing the tradeoffs between the aforementioned properties is a challenging task.

Manuscript received July 24, 2019; revised January 27, 2020 and June 17, 2020; accepted August 8, 2020. Date of publication September 22, 2020; date of current version October 6, 2021. This work was supported in part by the EU's Horizon 2020 Research and Innovation Programme under Grant Agreement 867433 (Fault-Learning) and Grant Agreement 739551 (KIOS CoE) and in part by the Republic of Cyprus through the Directorate General for European Programmes, Coordination, and Development. *(Corresponding author: Marios M. Polycarpou.)*

Kleanthis Malialis is with the KIOS Research and Innovation Center of Excellence, University of Cyprus, 1678 Nicosia, Cyprus (e-mail: malialis.kleanthis@ucy.ac.cy).

Christos G. Panayiotou and Marios M. Polycarpou are with the KIOS Research and Innovation Center of Excellence, University of Cyprus, 1678 Nicosia, Cyprus, and also with the Department of Electrical and Computer Engineering, University of Cyprus, 1678 Nicosia, Cyprus (e-mail: christosp@ucy.ac.cy; mpolycar@ucy.ac.cy).

This article has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the authors.

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2020.3017863

The contributions made are as follows.

1) We provide new insights into learning from nonstationary and imbalanced data, a largely unexplored area that focuses on the combined challenges of class imbalance and concept drift in online learning.
2) We propose the novel Adaptive REBAlancing (AREBA) algorithm that maintains the aforementioned desired properties. AREBA selectively includes in the training set a subset of the positive and negative examples that appeared so far, while at its heart lies an adaptive mechanism to continually maintain class balance.
3) We compare AREBA to strong baselines and state-of-the-art algorithms and perform extensive experimental work in scenarios with various imbalance rates and different drift types on both synthetic and real-world data. AREBA significantly outperforms the rest with respect to both learning speed and learning quality.
4) To our knowledge, this article is one of the very few studies that examine online imbalance learning under each type of drift independently. For reproducibility of our results, we make the data sets used and our code publicly available to the community.[1]

The organization of this article is given as follows. Section II provides the background material necessary to understand the contributions made. Section III provides an in-depth review of related work. AREBA is presented in Section IV. Our experimental setup is described in Section V. An analysis of the proposed method is given in Section VI, followed by a comprehensive comparative study in Section VII. We conclude in Section VIII where we discuss some important remarks, the pros and cons of AREBA, and pointers for future work.

II. Background

We consider a data generating process that provides, at each time step t, a sequence of examples or instances $S^t = \{(x_i^t, y_i^t)\}_{i=1}^M$ from an unknown probability distribution $p^t(x, y)$, where $x^t \in \mathbb{R}^d$ is a d-dimensional input vector belonging to input space $X \subset \mathbb{R}^d$, $y^t \in Y$ is the class label where $Y = \{0, 1\}$, and M is the number of instances arriving at each step. The focus of this article is on binary classification, and as a convention, the positive class represents the minority class. When the observed sequence S^t consists only of a single instance (i.e., $M = 1$), it is termed **online** (or one-by-one) learning; otherwise, it is termed **batch** (or chunk-by-chunk) learning [1]. The design of batch learning algorithms differs significantly from that of online learning as they are designed to process batches of data, possibly by utilizing an offline learning algorithm [3]. Therefore, the majority of batch learning algorithms are typically not suitable for online learning tasks [3]. This work focuses on online learning.

An online classifier receives a new example x^t at time step t and makes a prediction $\hat{y}^t$ based on a concept $h : X \to Y$ such that $\hat{y}^t = h(x^t)$. The classifier receives the true label y^t, and its performance is evaluated using a loss function and is then trained, i.e., its parameters are updated accordingly based on the loss incurred. This process is repeated at each time step. Depending on the application, new examples do not necessarily arrive at regular and predefined intervals.

If data are sampled from a long, potentially infinite, sequence, which is typically the case for big data applications, it is unrealistic to expect that all the previously observed data will always be available. If learning occurs on the most recent single instance only without taking into account previously observed data, it is termed **incremental** learning [1]. For online and incremental learning, the cost at time t is calculated using the loss function l as follows $J = l(y^t, \hat{y}^t)$.

This framework is suitable for human-in-the-loop learning. As mentioned, the label becomes available as the next example arrives, i.e., verification latency does not exist. Algorithms of this framework, including AREBA, are typically trained from user interaction by domain experts. Various and widely studied domains exist, which fit into this framework, and hence, this assumption is satisfied, e.g., in financial fraud [6], a banker can provide every few minutes if a credit card transaction is fraudulent. For rain prediction [4], an expert can provide every few hours if rain precipitation was observed. In healthcare applications [8], a doctor can provide the X-ray result as soon as it is completed. The framework may not be ideal in some cases (e.g., for real-time applications). Relaxing this assumption will be part of our future work, but we take the first step toward this direction and examine the robustness of all algorithms under conditions where this assumption is violated.

According to the Bayesian decision theory, a classification can be described by the prior probability $p(y)$ and the class conditional probability or likelihood $p(x|y)$ for all classes y [7]. The classification decision is made according to the posterior probability, which, for class y, is expressed as

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} \tag{1}$$

where $p(x) = \sum_{y=\{0,1\}} p(x|y)p(y)$.

Class imbalance [2] is a key challenge in learning and occurs when at least one data class is underrepresented, thus constituting a minority class. For a binary classification problem, class imbalance at time t occurs if

$$p^t(y = 0) \gg p^t(y = 1) \tag{2}$$

where class 0 (negative) and 1 (positive) represent the majority and minority classes, respectively.

Concept drift represents a change in the joint probability, and the drift between time step t_0 and t_1 is defined as follows:

$$\exists x \quad p^{t_0}(x, y) \neq p^{t_1}(x, y). \tag{3}$$

Concept drift can occur in three forms: 1) a change in prior probability $p(y)$; 2) a change in class-conditional probability or likelihood $p(x|y)$; and 3) a change in posterior probability $p(y|x)$. In real-world applications, the three forms can appear together. A change in posterior probability $p(y|x)$ that may or may not be due to a change in $p(x)$ is known as real drift because the true decision boundary is changed. A change in the distribution of the incoming data $p(x)$ without affecting $p(y|x)$ is known as virtual drift because the true decision

[1] https://github.com/kmalialis/areba/

boundary remains unchanged; however, the classifier's learned decision boundary may drift away from the true one.

While other drift characteristics [9] are important, our focus is on the drift type. As mentioned, along with [3], this article is one of the few studies that examine online imbalance learning under each type of concept drift independently.

III. Related Work

This section provides an in-depth review of related work and describes the state-of-the-art methods. For exhaustive surveys, the interested reader is directed toward these excellent articles: [1], [7] for drift methods, [2] for imbalance methods, [3] for methods that address both, and [10] for online ensembles.

A. Concept Drift

Concept drift algorithms are classified as memory-based, change detection-based, and ensembling [7].

1) Memory-Based: Memory-based algorithms typically employ a sliding window approach to maintain a set of recent examples that a classifier is trained on; a representative algorithm of this category is FLORA [11]. A key challenge is to determine *a priori* the window size as a larger window is better suited for a gradual drift, while a smaller window is suitable for an abrupt drift. To address this, methods use an adaptive sliding window [11] or multiple sliding windows [12]. This is also known as an abrupt forgetting approach because the examples that fall outside the window are immediately dropped out of memory. Alternatively, gradual forgetting approaches employ the full memory, but the influence of older examples is deteriorating, e.g., using an exponential decay weighting strategy [13].

2) Change Detection-Based: Change detection-based algorithms employ explicit mechanisms to detect concept drift. These methods are based on sequential analysis and control charts, e.g., Page-Hinkley (PH) test [14], cumulative sum (CUSUM) [14], and just-in-time (JIT) classifiers [15], [16], and on monitoring two distributions, e.g., adaptive windowing (ADWIN) [17]. These approaches are also known as active detectors and are generally suitable for detecting abrupt concept drift but may fail to work well in prediction settings with gradual or recurring concept drift [1] although recently JIT classifiers have been extended to address recurring concept drift [18].

3) Ensembling: An ensemble of classifiers can improve performance and provide the flexibility of injecting new data by adding classifiers or "forgetting" irrelevant data by removing or updating existing classifiers [19]. It can be computationally costly; recall that one of the desired properties of a classifier is to be able to operate fast in less than the example arrival time. Popular methods are the streaming ensemble algorithm (SEA) [20], Learn++.NSE [21], diversity for dealing with drifts (DDD) [22], and online bagging (OB) [23]. Another method is the accuracy updated ensemble (AUE) [24] that combines accuracy-based weighting mechanisms known from chunk-based ensembles with the incremental nature of Hoeffding Trees. Its follow-up work, online AUE (OAUE), is provided in [25]. The interested reader is directed toward [10] for a survey on ensemble learning for data streams.

None of the aforementioned approaches consider class imbalance. In the following, we discuss how these are combined with class imbalance methods to address the joint problem.

B. Class Imbalance

Cost-sensitive learning and resampling algorithms have recently shown particular success in this area [3].

1) Cost-Sensitive Learning: The cost-sensitive online gradient descent method (CSOGD) uses this loss function

$$J = \left(I_{y^t=0} + I_{y^t=1} \frac{c_p}{c_n} \right) \quad l(y^t, \hat{y}^t) \tag{4}$$

where $I_{\text{condition}}$ is the indicator function that returns 1 if condition is satisfied and 0 otherwise, and $c_p, c_n \in [0, 1]$ and $c_p + c_n = 1$ are the misclassification costs for positive and negative classes, respectively [26]. The authors use the perceptron classifier and stochastic gradient descent and apply the cost-sensitive modification to the hinge loss function, achieving excellent results. The downside of this method is that the costs need to be predefined; however, the extent of the class imbalance may not be known in advance. Moreover, in nonstationary environments, it cannot cope with imbalance changes (i.e., $p(y)$ drift) as the predefined costs remain static.

This issue can be resolved by introducing an adaptive cost strategy. One way to achieve an adaptive cost strategy is by using class imbalance detection (CID) [27] to determine the imbalance rate in an online manner. The authors define a time-decayed class size metric, where, for each class k, its size s_k is updated at each time t according to the following equation:

$$s_k^t = \theta s_k^{t-1} + I_{y^t=k}(1 - \theta) \tag{5}$$

where $0 < \theta < 1$ is a predefined time decay factor that gives less emphasis to older data. This metric can determine the imbalance rate at any given time; for instance, for a binary classification problem where the positive class is the minority ($s_p^t < s_n^t$), the imbalance rate at time t is given by s_n^t / s_p^t.

Another method that uses an adaptive cost strategy with a perceptron-based classifier is RLSACP [28]. EONN [29] uses an ensemble of cost-sensitive online neural networks to cope with drift and imbalance. As with CSOGD, the costs are predefined, thus limiting its adaptability to evolving data.

2) Resampling: Traditionally, in offline learning, resampling techniques alter the training set to deal with the skewed data distribution; especially, oversampling techniques "grow" the minority class, while undersampling techniques "shrink" the majority class. The simplest and most popular technique is random oversampling (or undersampling) where data examples are randomly added (or removed), respectively [30]. More sophisticated techniques exist; for example, the use of Tomek links [31] discards borderline examples, while the SMOTE [32] algorithm generates new minority class examples based on the similarities to the original ones. Recently, generative adversarial networks (GANs) have been used to approximate the distribution and generate data for the minority class [33].

Resampling has been demonstrated to be a powerful technique for addressing online imbalance learning problems as well. Uncorrelated bagging (UCB) [34] is an ensembling technique that is trained on all the minority examples observed so far, plus a subset of the most recent majority examples. This technique has two drawbacks; it assumes that the distribution of the minority class is stationary, and it does not handle the accumulated minority class examples for lifelong learning. SERA [35] and REA [36] are based on UCB and use more intelligent oversampling techniques. Other notable examples are the Learn++CDS and Learn++NIE [4] methods where both of them use the aforementioned Learn++NSE method (see Section III-A3) to address concept drift. The former combines the SMOTE algorithm to address the class imbalance, while the latter combines a variation of bagging. Despite their effectiveness, all these techniques are only suitable for batch learning and not for online learning, which is the focus of this article.

ESOS-ELM [37] is an ensemble of online sequential extreme learning machines that are trained on balanced subsets of the data stream. It relies, however, on the assumption that drift does not affect the minority class. Oversampling-based OB (OOB) [38] is an online ensembling method that extends the OB method (see Section III-A3) that addresses concept drift. It works by adjusting the learning bias from the majority to the minority class adaptively through resampling by utilizing the CID method (see Section III-B1) and its time-decayed class size metric defined in (5). Its basic idea is as follows. OOB updates each classifier of the ensemble K times. If a minority example arrives, the value of K increases; otherwise, it decreases. The effectiveness of OOB has been demonstrated using two types of classifiers: Hoeffiding trees and neural networks. The ensemble size varies for each study, e.g., in [38], 50 trees and 50 neural networks were used, while, in [3], 15 neural networks were used. The approach can be computationally costly and may hinder online learning in high-speed sequential applications for two reasons. The first reason is due to the multiple classifiers (ensembling), and the second is because each classifier gets updated multiple times per time step. Analogous to OOB, its authors also introduce [38] the Undersampling-based OB (UOB) algorithm.

C. Open Challenges

Several key challenges still remain open when the joint problem of imbalance and drift is considered. Krawczyk *et al.* [10] state that "working with class-imbalanced and evolving streams is still in early stages," while this study [3] "reveals research gaps in the field of online imbalance learning with concept drift." Especially, many existing methods are capable of addressing only a single problem, either imbalance or drift, but not the joint problem. In other methods that address the joint problem, weaknesses are revealed under conditions where one or both the problems become very challenging. For instance, we will demonstrate these weaknesses under conditions where the class imbalance is extreme (e.g., 0.1%). This article introduces the concept of maintaining separate and balanced queues for each class and has a dual-nature as it merges ideas from memory-based and resampling algorithms.

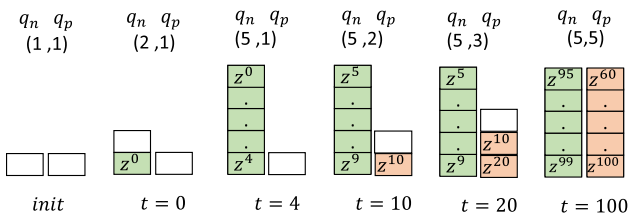

Fig. 1. Example of QBR for $B = 10$. Negative examples are shown in green, positive ones in light red, and the minority class is the positive class. It takes 100 time steps for the queues to become balanced.

Also, besides its type, drift can be classified by its severity, speed, predictability, frequency, and recurrence [39]. A more recent study characterizes drift by subject, frequency, transition, reoccurrence, and magnitude [9]. Therefore, in practice, it is very difficult to characterize concept drift. Our focus is on learning the concept drift without its explicit characterization and detection. As discussed in Section III-A2, explicit or active drift detectors can perform well under specific drift characteristics. However, no detector can universally perform satisfactorily under any combination of drift characteristics [40]. In fact, detailed characteristics of drift have not been consistently investigated in the literature [10]. Our proposed approach learns the concept drift without its explicit characterization and detection and adapts the classifier continuously.

IV. Proposed Method

We now introduce the AREBA algorithm. Its central idea is to selectively include in the training set a subset of the positive and negative examples that appeared so far. At its heart lies an adaptive mechanism that dynamically modifies the queue sizes to maintain the class balance between the selected examples. AREBA extends our recently introduced algorithm queue-based resampling (QBR) [41].

A. Queue-Based Resampling

The memory size $B \in 2\mathbb{Z}^+$ (i.e., $B \geq 2$ and even) determines how many previously observed examples can be stored. The selection of the examples is achieved by maintaining at any given time t two separate windows of capacity (maximum length) $(B/2)$. The windows are implemented using queues, i.e., q_n^t and q_p^t contain the negative and positive examples, respectively

$$q_n^t = \{(x_i, y_i)\}_{i=1}^{|q_n^t|}$$
$$q_p^t = \{(x_i, y_i)\}_{i=1}^{|q_p^t|} \tag{6}$$

where $|q_n^t|, |q_p^t| \in [0, \frac{B}{2}]$ are the current lengths of the queues. Let $z_i = (x_i, y_i)$, for any two $z_i, z_j \in q_n^t$ (or q_p^t), such that $j > i$, and z_j arrived more recently in time.

An example showing how QBR works when $B = 10$ for 100 time steps is shown in Fig. 1. Negative examples are shown in green, positive ones in light red, and the minority class is a positive class. The class imbalance is set

Algorithm 1 QBR

1: **Input:**
2: f: classifier
3: B: total storage size ($B \geq 2$)
4: **Initialization:**
5: queues $q_p^0, q_n^0 = \{\}$
6: queue capacities $q_p^0.cap = q_n^0.cap = 1$
7: **for** each time step t **do**
8: receive example $x^t \in \mathbb{R}^d$
9: predict class $\hat{y}^t \in \{0, 1\}$
10: receive true label $y^t \in \{0, 1\}$
11: **if** $y^t == 1$ **then**
12: $q_p^t = q_p^{t-1}.append\big((x^t, y^t)\big)$
13: **else**
14: $q_n^t = q_n^{t-1}.append\big((x^t, y^t)\big)$
15: **if** $q_p^t.is\_full()$ **then**
16: **if** $q_p^t.cap < \frac{B}{2}$ **then**
17: $q_p^t.cap = q_p^t.cap + 1$ ▷ increase capacity
18: **else if** $q_p^t.cap == \frac{B}{2}$ **then**
19: pass ▷ queue no longer grows
20: **if** $q_n^t.is\_full()$ **then**
21: **if** $q_n^t.cap < \frac{B}{2}$ **then**
22: $q_n^t.cap = q_n^t.cap + 1$
23: **else if** $q_n^t.cap == \frac{B}{2}$ **then**
24: pass
25: prepare the training set $q^t = q_p^t \cup q_n^t$
26: calculate cost J on q^t using Eq (7)
27: update classifier once $f.train()$

to CI = 10%, i.e., $p(y = 1) = 0.1$, and for the sake of illustration, positive instances arrive at times multiple of ten ($t = 10, 20, \ldots, 100$). Initially, both queues are empty (shown as empty boxes), but their capacity is set to one (shown in the parenthesis). At $t = 0$, a negative example (z^0) arrives, which is appended to the negative queue, and the queue's capacity is incremented by one. At $t = 4$, the negative queue is full and has reached the full capacity ($B/2$). At $t = 10$, the first positive example (z^{10}) arrives, which is appended to the positive queue, and the queue's capacity is incremented. At $t = 100$, the positive queue is full and has reached the full capacity ($B/2$). Note that, at this time, both queues contain the most recent ($B/2$) examples, i.e., $z^{95}, \ldots, z^{99}$ and $z^{60}, \ldots, z^{100}$ for the negative and positive queues, respectively.

The union of the two queues is then taken to form the new training set. The cost function is given by

$$J = \frac{1}{|q^t|} \sum_{i=1}^{|q^t|} l(y_i, h(x_i)) \tag{7}$$

where $q^t = q_p^t \cup q_n^t$ and $|q^t| \in [1, B]$. At each step, the classifier is updated once based on the cost J incurred. QBR's pseudocode is shown in Algorithm 1. In lines 5 and 6, the queues are initially empty with a capacity of one each. In lines 11–14, a new example is appended in its relevant queue based on its true label. The append function behaves exactly as in Fig. 1, i.e., it inserts the most recent example in a queue while discarding the oldest one. Consider the case of $t = 10$ where the most recent example in the negative queue is z^9. When the example z^9 arrived at $t = 9$, the example z^4 was discarded from the queue. In lines 15–24, the capacity of the relevant queue is incremented. In lines 25–27, the training set is prepared, the cost is calculated, and the classifier is updated once.

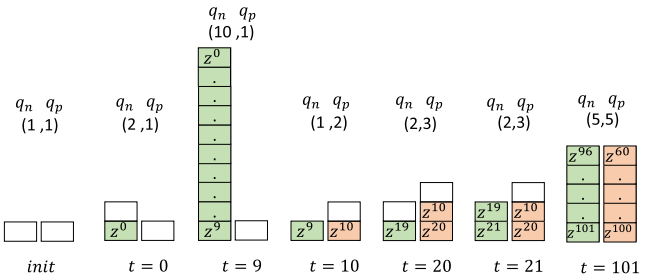

Fig. 2. Example of AREBA for $B = 10$. Negative examples are shown in green, positive ones in light red, and the minority class is the positive class. Rebalancing enforces the queues to remain balanced throughout the time.

Of particular importance is the observation that the original class imbalance problem still persists in the queues for a sustained period of time. Let us revisit time $t = 10$ in Fig. 1. While q_n^t is full, q_p^t only contains a single example. Recall that to train the classifier, we first take the union of the queues and then calculate the cost. Class imbalance is reduced as positive examples arrive, and the problem eventually disappears at $t = 100$ when the queues become balanced.

B. Adaptive Rebalancing

AREBA introduces a novel element that dynamically modifies the queue lengths in order to constantly maintain the balance between the queues. Without this element, the initial class imbalance problem would still persist in the queue-based system as discussed in Section IV-A.

We describe in Fig. 2 how AREBA works through the same example as before. Negative examples are shown in green, positive ones in light red, and the minority class is a positive class. The class imbalance is set to CI = 10%, i.e., $p(y = 1) = 0.1$, and for the sake of illustration, positive instances arrive at times multiple of ten ($t = 10, 20, \ldots, 100$). Initially, both queues are empty (shown as empty boxes), but their capacity is set to one (shown in the parenthesis). At $t = 0$, a negative example (z^0) arrives, which is appended to the negative queue, and the queue's capacity is incremented by one. Contrary to QBR, each queue is allowed to have a maximum capacity of B (rather than ($B/2$)). Since no positive examples are observed in the beginning, at $t = 9$, q_n is full and has reached the maximum capacity B.

The first positive example (z^{10}) arrives at $t = 10$ and is appended to q_p. Rebalancing is now initiated, and the capacity of q_n and q_p becomes 1 and 2, respectively. q_n only contains its most recent example (z^9); hence, the queues are now balanced. The queues remain balanced until the second positive example (z^{20}) arrives at $t = 20$. q_p contains now the two most recent positive examples (z^{20} and z^{10}),

Algorithm 2 AREBA

1: **Input:**
2: f: classifier
3: B: total storage size ($B \geq 2$)
4: θ: decay factor for class size metrics s_p^t, s_n^t
5: **Initialization:**
6: class sizes $s_p^0, s_n^0 = 0$
7: queues $q_p^0, q_n^0 = \{\}$
8: queue capacities $q_p^0.cap = q_n^0.cap = 1$
9: **for** each time step t **do**
10: receive example $x^t \in \mathbb{R}^d$
11: predict class $\hat{y}^t \in \{0, 1\}$
12: receive true label $y^t \in \{0, 1\}$
13: update positive class size $s_p^t = \theta s_p^{t-1} + I_{y^t=1}(1-\theta)$
14: update negative class size $s_n^t = \theta s_n^{t-1} + I_{y^t=0}(1-\theta)$
15: **if** $y^t == 1$ **then**
16: $q_p^t = q_p^{t-1}.append\big((x^t, y^t)\big)$
17: **else**
18: $q_n^t = q_n^{t-1}.append\big((x^t, y^t)\big)$
19: **if** $q_p^t.is\_empty()$ **then**
20: **if** $q_n^t.cap < B$ **then**
21: $q_n^t.cap = q_n^t.cap + 1$ ▷ increase capacity
22: **else if** $q_n^t.cap == B$ **then**
23: pass ▷ queue no longer grows
24: **else if** $q_n^t.is\_empty()$ **then**
25: **if** $q_p^t.cap < B$ **then**
26: $q_p^t.cap = q_p^t.cap + 1$
27: **else if** $q_p^t.cap == B$ **then**
28: pass
29: **else**
30: **if** $s_n^t > s_p^t$ **then** ▷ if positive class is minority
31: **if** $q_p^t.is\_full()$ **then**
32: **if** $q_p^t.cap < \frac{B}{2}$ **then**
33: $q_p^t.cap = q_p^t.cap + 1$
34: $q_n^t.cap = q_p^t.cap - 1$
35: **else if** $q_p^t.cap == \frac{B}{2}$ **then**
36: **if** $q_n^t.cap \neq q_p^t.cap$ **then**
37: $q_n^t.cap = q_p^t.cap$
38: **if** $s_n^t \leq s_p^t$ **then** ▷ if negative class is minority
39: **if** $q_n^t.is\_full()$ **then**
40: **if** $q_n^t.cap < \frac{B}{2}$ **then**
41: $q_n^t.cap = q_n^t.cap + 1$
42: $q_p^t.cap = q_n^t.cap - 1$
43: **else if** $q_n^t.cap == \frac{B}{2}$ **then**
44: **if** $q_p^t.cap \neq q_n^t.cap$ **then**
45: $q_p^t.cap = q_n^t.cap$
46: prepare the training set $q^t = q_p^t \cup q_n^t$
47: calculate cost J on q^t using Eq (7)
48: update classifier once $f.train()$

while q_n contains the most recent negative example (z^{19}). At $t = 20$, the capacity of q_n and q_p becomes 2 and 3, respectively. At $t = 21$, another negative example (z^{21}) arrives, which is appended to the relevant queue; thus, the queues are again balanced. At $t = 101$, each queue is full and has a capacity of $(B/2)$.

AREBA proposes an adaptive mechanism that dynamically alters the queue sizes to maintain a balance between the examples contained in the queues. To achieve this, it is necessary to be able to decide in an online fashion which class is the minority and which the majority. AREBA adopts the CID method's time-decayed class size metrics defined in (5).

AREBA is shown in Algorithm 2. In lines 7 and 8, the capacity of each queue is initialized to 1. In lines 13 and 14, the class size metrics are updated. The new example is appended in its relevant queue (lines 15–18). We then check if one of the queues is empty (lines 19–28). For instance, if the positive queue is empty, we increase the capacity of the negative queue by 1; this corresponds to the cases $t = 0$ to $t = 9$ in Fig. 2. Since the positive class is the minority class, the rest of the cases in Fig. 2 are captured by lines 30–37. Line 31 checks if the positive queue is full (in our illustration, this occurs at $t = 10, 20, \ldots, 100$), and then we adapt the capacities accordingly (lines 32–37) depending on whether the capacity of the positive queue has reached $(B/2)$. Similarly, lines 38–45 are applicable when the negative class is the majority class.

In summary, AREBA introduces the concept of maintaining separate and balanced queues for each class, and its effectiveness is attributed to its dual-nature as it combines ideas from memory-based and resampling methods. These and other important remarks are discussed in detail in Section VIII.

V. Experimental Setup

This section describes the synthetic and real-world data sets used along with any data preprocessing steps performed. It also describes the baseline and state-of-the-art methods used along with their selected parameters. It further describes the performance metrics and the evaluation method used.

A. Data Sets

1) Synthetic Data Sets: They provide the flexibility to control the imbalance level, when to introduce drift, and control the drift type. We experiment with an imbalance of 10% (mild), 1% (severe), and 0.1% (extreme) and examine each drift type individually to inspect the advantages and limitations of all the compared methods. The three synthetic data sets used are described below where we will use their balanced and imbalanced versions, with and without drift.

a) Circle [42]: It has the two features $x_1, x_2 \in [0, 1]$. The classification function is a circle $(x_1 - x_{1c})^2 + (x_2 - x_{2c})^2 = r_c^2$, where (x_{1c}, x_{2c}) is its center and r_c its radius. The circle with $(x_{1c}, x_{2c}) = (0.4, 0.5)$ and $r_c = 0.2$ is created. Instances inside the circle are classified as positive, otherwise as negative.

b) Sine [42]: It consists of the features $x_1 \in [0, 2\pi]$ and $x_2 \in [-1, 1]$. The classification function is $\sin(x_1)$. Instances below the curve are classified as positive, otherwise as negative. Feature rescaling has been performed so that $x_1, x_2 \in [0, 1]$.

c) Sea [20]: It has the features $x_1, x_2 \in [0, 10]$. Instances that satisfy $x_1 + x_2 \leq 7$ are classified as positive, otherwise as negative. Rescaling has been performed so that $x_1, x_2 \in [0, 1]$.

2) Real-World Data Sets: They are typically more complex than synthetic ones and have a large number of noisy features, but the true nature of concept drift may be unknown. The five data sets used in this article cover various application domains, specifically, healthcare, security and crime, finance and banking, image classification, and environmental sciences.

a) Cervical cancer [8]: The data set was collected at the Hospital Universitario de Caracas in Caracas, Venezuela, and contains demographic information, habits, and historical medical records of 858 patients. The task is to predict the outcome of a biopsy with respect to cervical cancer, and each class label was decided by a team of six experts. The data set is highly imbalanced as 55 out of the 858 cases (6.4%) correspond to cases of cervical cancer. The number of features is 45. The exact nature of drift is unknown, but it is expected to occur due to the fact that cancer cells gain genetic variation over time and also due to changes in patients' habits.

b) Fraud [6]: The data set contains transactions made by credit cards in September 2013 by European cardholders. This data set presents transactions that occurred in two days, where we have 492 frauds out of 284 807 transactions. The data set contains 30 features and is severely imbalanced as the positive class (frauds) accounts for 0.172% of all transactions. The exact nature of concept drift is unknown, but it is expected to occur due to the adaptive nature of adversarial actions.

c) Credit score [43]: The data set contains demographic and financial/banking information. The task is to predict the credit score (good and bad) of a customer. The data set contains 1000 entries, out of which 300 correspond to bad credit, i.e., the class imbalance is 30%. The number of features is 24. The exact nature of concept drift is unknown but may occur due to customer attempts to fake or improve their credit score.

d) MNIST [44]: It is a database of handwritten digits ("0"–"9") where each image is 28×28. Contrary to the rest of the data sets where their data type is numeric, MNIST is commonly used for training image processing systems for visual tasks. The data set is diverse as the digits were written by ∼250 adult and student writers. We selected digit "7" to be the majority class with 6000 instances and digit "2" to be the minority class with 60 instances; therefore, the imbalance is close to 1%. Drift can occur as new handwriting styles appear during learning.

e) Forest cover type [45]: The data sets consist of cartographic information obtained from the U.S. Forest Service. The task is to predict the forest cover type for given 30×30 m cells from the Roosevelt National Forest in Colorado. We have selected the cover type "1" to be the majority class with 200 000 instances and type "4" to be the minority class with 2000 instances; therefore, the imbalance is close to 1%. This data set has been used in many concept drift studies, e.g., [24].

B. Compared Methods

All compared methods share the same base classifier that is a fully connected neural network of one eight-neuron hidden layer, except for MNIST that consists of two 512-neuron hidden layers to deal with its complex data type (i.e., images). The base classifier is configured as follows: He Normal [46] weight initialization, the Adam [47] optimization algorithm, LeakyReLU [48] as the activation function of the hidden neurons, sigmoid activation for the output neuron, and the binary cross-entropy loss function. The learning rate is 0.01 for the synthetic, Credit Score, and Forest Cover Type data sets, 0.1 for Cervical Cancer, and 0.0001 for Fraud. For MNIST, the learning rate is 0.001, and $L2$ regularization is set to 0.01.

1) Baseline: A baseline algorithm where no mechanisms to address class imbalance or concept drift exists. This baseline method is an online and incremental learning algorithm.

2) Sliding Window: A memory-based method uses a single sliding window to address drift, but no mechanism to address imbalance exists. This is in contrast to QBR that utilizes one sliding window per class. The window size is set to $W = 100$. It is an online but not incremental learning algorithm as it requires access to $W - 1$ previously observed data examples.

3) Adaptive_CS: The state-of-the-art adaptive cost-sensitive learning method. It uses the CSOGD cost function defined in (4) initialized to $c = (c_p/c_n) = (0.95/0.05) = 19$ as suggested by its authors. These costs are adapted according to the CID approach, where its time-decayed class size metrics are defined in (5). The time-decayed factor is set to $\theta = 0.99$. To overcome stability issues in performance, we set an upper bound to the ratio, i.e., $1 \leq c \leq 50$, and we update the costs every 250 steps. This method is both online and incremental.

4) OOB: The state-of-the-art online resampling algorithm (see Section III-B2). We will consider OOB with 20 classifiers and the special case OOB_single where only a single classifier is used. The time-decayed factor is set to $\theta = 0.99$. OOB is an online and incremental learning algorithm, but unlike other methods, it performs multiple updates of the classifier at each time step.

5) AREBA: The proposed method whose pseudocode is provided in Algorithm 2. The time-decayed factor is set to $\theta = 0.99$. AREBA is an online learning algorithm. When $B = 2$, the method (referred to as AREBA_2) becomes a near-incremental learning algorithm as it requires access only to a single old example. In our study, we will examine various values of memory size, which we will refer to as AREBA_B.

We discuss, now, some aspects concerning the computational cost of algorithms. For all methods, one-pass learning is used, i.e., the base classifier is updated once (#epochs = 1) at every step, except for OOB that is updated K times (#epochs = K). A single batch update is performed within an epoch; this is to allow fast learning in high-speed applications. For AREBA, q^t is the batch, while, for Sliding, the window W is the batch. For the rest, the batch size is 1 as they are incremental algorithms. The same classifier gets updated throughout the duration of an experiment. Even in the presence of drift, we never reset the classifier or introduce any new classifier(s).

C. Performance Metrics

Traditionally, classifiers are evaluated using the overall accuracy metric. When the class imbalance exists, accuracy becomes problematic as it is biased toward the majority class. This occurs because any metric that uses values from both

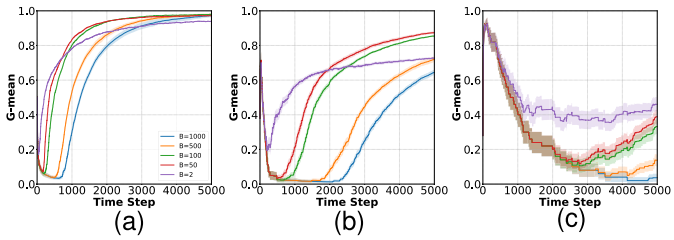

Fig. 3. QBR's behavior for the Circle data set (stationary) with different memory sizes $B = 2, 50, 100, 500$, and 1000 and class imbalance rates of (a) CI = 10%, (b) CI = 1%, and (c) CI = 0.1%. QBR is very sensitive to the memory size as the learning speed and/or the final performance are heavily affected by the choice of B.

columns of the confusion matrix will be inherently sensitive to class imbalance [2]. Therefore, it is necessary to adopt a performance metric that is not sensitive to class imbalance.

The geometric mean G-mean [2] is such a suitable metric that evaluates the degree of inductive bias in terms of a ratio of positive accuracy Acc^+ (or recall) and negative accuracy Acc^- (or specificity), as defined in the following [49]:

$$G\text{-mean} = \sqrt{\text{Acc}^+ * \text{Acc}^-} = \sqrt{(\text{TP/P}) * (\text{TN/N})} \tag{8}$$

where TP, P, TN, and N are the numbers of true positives, total positives, true negatives, and total negatives, respectively. G-mean has some desirable properties as it is high when both Acc^+ and Acc^- are high and when their difference is small.

D. Evaluation Method

To evaluate, compare, and assess predictive sequential learning algorithms, we adopt the prequential error with fading factors method. It has been proven that for learning algorithms in stationary data, this method converges to the Bayes error [50]. This method does not require a holdout set, and the predictive model is always tested on unseen data. In our experimental study, the fading factor is set to $\theta = 0.99$.

In all simulations, we plot the prequential metric (e.g., G-mean) in every step averaged over 50 repetitions, including the error bars displaying the standard error around the mean. In addition, in all experiments, we test for statistical significance using a one-way repeated measures ANOVA and then using posthoc multiple comparisons tests with Fisher's least significant difference correction procedure to show which of the compared method is significantly different from the others.

VI. Empirical Analysis of AREBA

This section presents a twofold analysis of the proposed method. In particular, we examine and discuss the roles of the AREBA mechanism and the memory size.

A. Role of the Adaptive Rebalancing Mechanism

This analysis has been conducted on the stationary Circle data set, i.e., without concept drift. Fig. 3(a)–(c) depicts how the memory size affects the learning performance of QBR. The left, middle, and right columns correspond to experiments with class imbalance of CI = 10%, 1%, and 0.1%, respectively.

In Fig. 3(a), where the imbalance is mild, i.e., CI = 10%, the final performance remains unaffected for $B \geq 50$, while, for $B < 50$, it performs slightly worse. However, the learning speed is severely affected by the choice of B, i.e., it gets slower with an increasing value of B. For instance, QBR with $B = 500$ equalizes the performance of $B = 2$ at about $t = 2000$.

In Fig. 3(b) where the class imbalance is severe, i.e., CI = 1%, both the learning speed and final performance are severely affected by the choice of B. The learning speed gets slower with an increasing value of B; for instance, QBR with $B = 500$ equalizes the performance of $B = 2$ at about $t = 5000$. In regard to the final performance, after a certain threshold (here, $B \geq 50$), it significantly deteriorates as the value of B increases. For instance, QBR with $B = 1000$ does not even equalize the performance of $B = 2$ after 5000 time steps.

In Fig. 3(c) where the imbalance is extreme (CI = 0.1%), the learning speed and final performance are severely affected by B, as previously. QBR with $B = 2$ is by far the best algorithm. QBR with $B = 50$ and 100 only start closing the gap after 5000 time steps, while $B = 500$ and 1000 perform poorly at $t = 5000$.

The analogous experiments for AREBA are shown in Fig. 4(a)–(c). Irrespective of the imbalance severity, after a certain threshold (here, $B \geq 50$), all AREBA versions behave almost identically. To sum up, these important remarks can be made.

1) Without the AREBA mechanism, QBR is very sensitive to the choice of the memory size B, and as a result, the learning speed and/or the final performance are significantly affected. The problem becomes more acute as class imbalance becomes more severe.
2) The AREBA mechanism is robust to the choice of the memory size B, irrespective of the imbalance severity. Especially, the higher the value of B, the better; however, after a (data set-specific) threshold, the improvement is negligible.

B. Role of the Memory Size

This section identifies the role of the memory size in the presence of outdated concepts. We examine both the leaning speed and final performance. To examine the learning speed, we present the learning curves; when the curve is steep, it means that the algorithm is learning faster. The learning curves present the prequential G-mean at every time step. To examine the learning quality, we present the final performance, i.e., the one obtained at the last time step of the curve.

The analysis is conducted on the synthetic data sets in two settings, based on [3]. The short setting (5000 steps, drift at $t = 2500$) allows us to examine AREBA's behavior immediately after the drift, while the long (20 000 steps, drift at $t = 10\,000$) allows us to examine AREBA's long-term behavior.

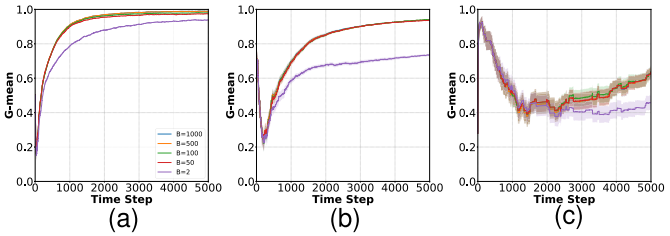

Fig. 4. Role of the AREBA mechanism for the Circle data set (stationary) with different memory sizes (B) and class imbalance (CI) rates. The AREBA mechanism is robust to different values of the memory size. (a) CI = 10%. (b) CI = 1%. (c) CI = 0.1%.

For the Circle data set, the drift is defined as follows:

$$\begin{aligned} p(y=1|x \text{ inside circle}) &= 1.0 \longrightarrow 0.0 \\ p(y=1|x \text{ outside circle}) &= 0.0 \longrightarrow 1.0 \\ p(y=0|x \text{ outside circle}) &= 1.0 \longrightarrow 0.0 \\ p(y=0|x \text{ inside circle}) &= 0.0 \longrightarrow 1.0. \end{aligned} \tag{9}$$

Fig. 5 shows the results for the Circle data set under posterior drift. Fig. 5(a)–(c) depicts AREBA's learning curves for various memory sizes with class imbalance of CI = 10%, 1%, and 0.1%, respectively. Fig. 5(d) and (e) depicts the final performances before and after the drift, respectively.

We start with the case of mild imbalance [see Fig. 5(a)]. For the part of the curves before the drift, we conclude that the higher the value of B the better; however, after a (data set-specific) threshold, the improvement is negligible. This has been studied in Section VI-A [see Fig. 4(a)]; therefore, from now on, we turn our attention to the part of the learning curves after the drift.

For the learning speed, AREBA with $B = 500$ and 1000 is slower than $B = 20$ and 100, which, in turn, is slower than $B = 2$. For the final performance, AREBA with $B = 2$ significantly outperforms the rest. We observe that the smaller the value of B, the better the results. This is also clear in Fig. 5(e). This is expected as immediately after a drift, a larger value of B means that more outdated examples exist in the queues.

Surprisingly, this no longer holds when imbalance becomes severe or extreme [see Fig. 5(b) and (c)] where all AREBA versions yield the same final performance. This is attributed to two reasons. First, a small value of B is suitable in case of outdated examples with mild imbalance [see Fig. 5(a)]. Second, in the case of severe imbalance without drift, a large value of B is suitable [see Fig. 4(b)]. In the presence of both drift and severe imbalance, the aforementioned are in conflict with each other, and interestingly, it appears that imbalance becomes the key issue when it is severe rather than drift.

Overall, an important tradeoff exists. In stationary settings, the higher the value of B, the better the results, as shown in Fig. 4(a). In nonstationary settings, the smaller the value of B, the better the performance, as shown in Fig. 5(a). The effect of this tradeoff appears to be fading away in the presence of severe or extreme imbalance, as shown in Fig. 5(b) and (c). The importance of this tradeoff is high and should be carefully assessed, as the optimal value of B depends on the data set.

For the Sine data set, the posterior drift is defined as follows:

$$\begin{aligned} p(y=1|x \text{ below curve}) &= 1.0 \longrightarrow 0.0 \\ p(y=1|x \text{ above curve}) &= 0.0 \longrightarrow 1.0 \\ p(y=0|x \text{ above curve}) &= 1.0 \longrightarrow 0.0 \\ p(y=0|x \text{ below curve}) &= 0.0 \longrightarrow 1.0. \end{aligned} \tag{10}$$

Fig. 6 shows the results for the Sine data set under posterior drift. Fig. 6(a)–(c) depicts AREBA's learning curves, while Fig. 6(d) and (e) depicts the final performances. For mild imbalance [see Fig. 6(a)], the learning speed of AREBA with $B = 500$ and 1000 is slower than $B = 20$ and 100. In case of severe imbalance [see Fig. 6(b)], we can conclude that the key issue becomes the imbalance rather than the drift. Notably, AREBA with $B = 1000$ outperforms $B = 2$ and equalizes the performance of $B = 100$ and 500 despite containing significantly more outdated examples in its queues. In Fig. 6(e), the best final performance when the imbalance is mild (CI = 10%) occurs when $B = 100$. The best final performance when the imbalance is severe (CI = 1%) occurs when $B = 20$. Under conditions of mild imbalance [see Fig. 6(a)], the aforementioned tradeoff is not as clear as it was for the Circle data set [see Fig. 5(a)]. Therefore, we note that the need for some experimentation to find a suitable choice of B becomes even more important.

For the Sea data set, the posterior drift is defined in 11. Fig. 7 shows the results for the Sea data set under posterior drift. Fig. 7(a)–(c) depicts AREBA's learning curves, while Fig. 7(d) and (e) depicts the final performances

$$\begin{aligned} p(y=1|x_1+x_2 \leq 7) &= 1.0 \longrightarrow 0.0 \\ p(y=1|x_1+x_2 > 7) &= 0.0 \longrightarrow 1.0 \\ p(y=0|x_1+x_2 > 7) &= 1.0 \longrightarrow 0.0 \\ p(y=0|x_1+x_2 \leq 7) &= 0.0 \longrightarrow 1.0. \end{aligned} \tag{11}$$

Notice that the experiment runs for 20 000 steps to inspect AREBA's long-term behavior. We start with the case of mild imbalance [see Fig. 7(a)]. While, immediately after the drift, AREBA with $B = 500$ and 1000 appears to be learning slower, given additional time, the two eventually outperform the rest. This is expected because after a long time without any recurring drift, the data can be considered as stationary, and hence, we reach the same conclusion as previously [see Fig. 4(a)].

For severe imbalance [see Fig. 7(b)], we conclude that the key issue becomes the imbalance rather than the drift. Notably, AREBA with $B = 1000$ outperforms $B = 2$ and equalizes $B = 20$ and 500 despite containing significantly more outdated examples in its queues. The best final performance when the imbalance is severe occurs when $B = 100$. This is also shown in Fig. 7(e). For the case of extreme class imbalance, i.e., CI = 0.1%, all AREBA versions (except when $B = 2$) perform similarly although $B = 20$ has a slight advantage.

To sum up, the following important remarks can be made.

1) For mild class imbalance, the general trend is that the lower the value of the memory size B, the better the AREBA performs. This is attributed to the fact that a smaller number of outdated examples are contained in

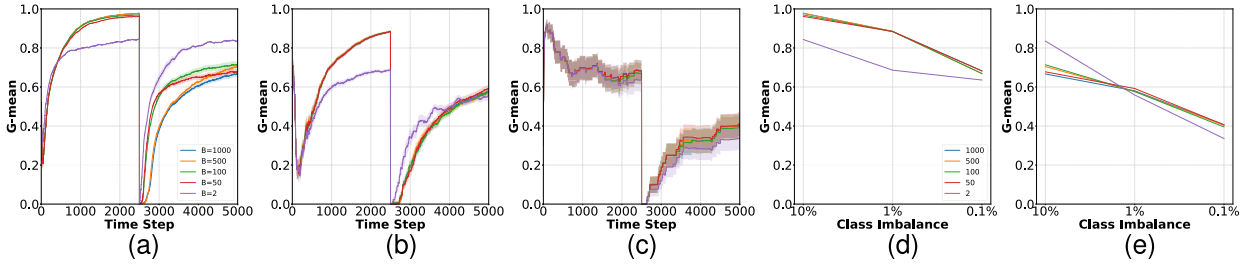

Fig. 5. Role of the memory size (B) for AREBA in the Circle data set with posterior probability drift and various class imbalance (CI) rates. The last two figures depict the final performance before ($t = 2499$) and after ($t = 5000$) the drift. (a) CI = 10%. (b) CI = 1%. (c) CI = 0.1%. (d) Predrift. (e) Postdrift.

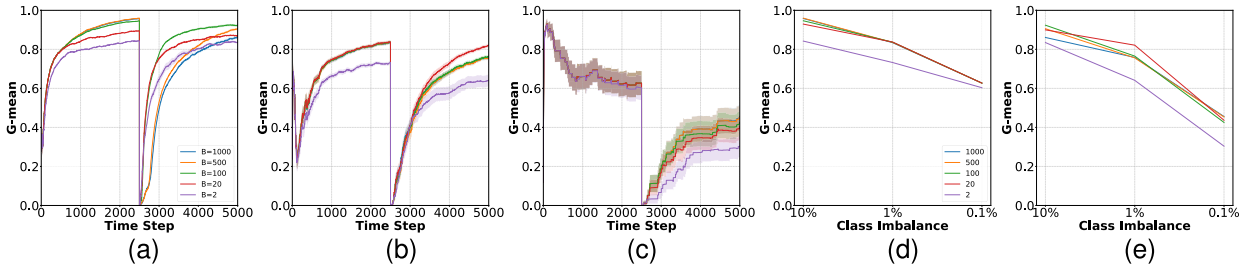

Fig. 6. Role of the memory size (B) for AREBA in the Sine data set with posterior probability drift and various class imbalance (CI) rates. The last two figures depict the final performance before ($t = 2499$) and after ($t = 5000$) the drift. (a) CI = 10%. (b) CI = 1%. (c) CI = 0.1%. (d) Predrift. (e) Postdrift.

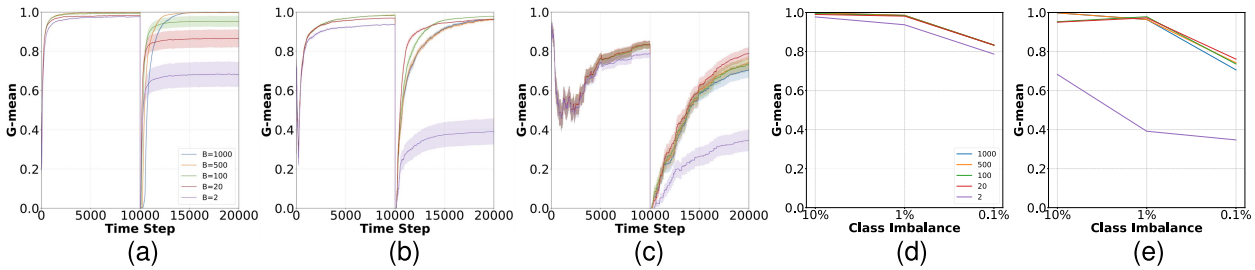

Fig. 7. Role of the memory size (B) for AREBA in the Sea data set with posterior probability drift and various class imbalance (CI) rates. The last two figures depict the final performance before ($t = 9999$) and after ($t = 20\,000$) the drift. (a) CI = 10%. (b) CI = 1%. (c) CI = 0.1%. (d) Predrift. (e) Postdrift.

the queues. The optimal value of B depends on the data set.

2) When a severe imbalance exists, it appears to be the key problem rather than drift. Surprisingly, AREBA with large values of B has been shown to perform similarly or even better than AREBA with lower values. Interestingly, this is somewhat in contradiction to the previous point as the former contains more outdated examples in the queues.
3) The previous guidelines are useful to help with the selection of the memory size parameter. However, some experimentation is still necessary to obtain the best value of B. We discuss the role of B further in Section VIII.

VII. Comparative Study

A. Stationary Data

We describe our work on stationary synthetic data. Figs. 8–10 show the results for the Sine data set with imbalance of CI = 10%, 1%, and 0.1%, respectively. For completeness, we also present the prequential recall and specificity.

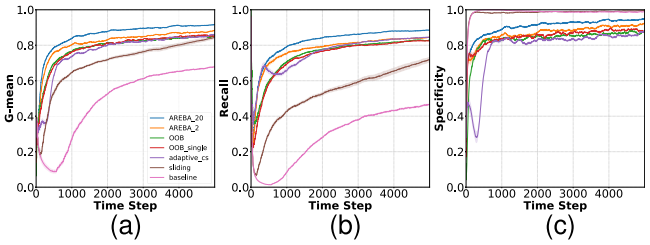

Fig. 8. Comparative study on Sine (stationary) with mild imbalance 10%. (a) G-mean. (b) Acc^{+}. (c) Acc^{-}.

Based on the analysis in Section VI, we choose $B = 20$ for the Sine data set.

In Fig. 8(a), the best performance is achieved by AREBA_20. The rest reach a similar performance with the exception of the Baseline. Both AREBA versions learn faster with OOB/OOB_single being the second best. Noteworthy, the 20-classifier ensemble (OOB) performs similar to the single classifier OOB_single. While this may seem surprising at first, it is, in fact, consistent with the results of their authors [38], where they have concluded that resampling, and

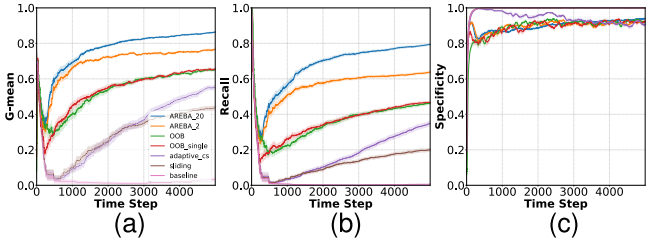

Fig. 9. Comparative study on Sine (stationary) with severe imbalance 1%. (a) G-mean. (b) Acc^+. (c) Acc^-.

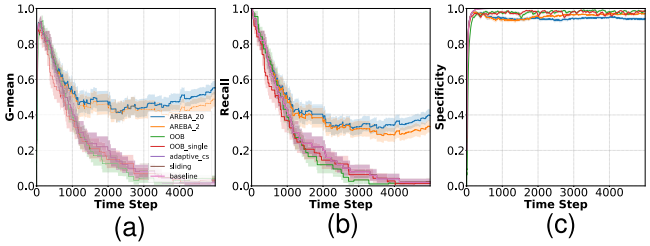

Fig. 10. Comparative study on Sine (stationary) with extreme imbalance 0.1%. (a) G-mean. (b) Acc^+. (c) Acc^-.

not ensembling, is the reason behind the effectiveness of the approach.

In Fig. 9(a) where severe imbalance exists, the two AREBA versions significantly outperform the rest. Under extreme imbalance [see Fig. 10(a)], AREBA performs more than ten times better than the rest. AREBA's effectiveness is attributed to the following. While all algorithms perform well on the majority class [see Figs. 9(c) and 10(c)], AREBA has a superior performance on the minority class [see Figs. 9(b) and 10(b)]. This means that the rest of the algorithms find it difficult to classify correctly positive examples. Recall that the G-mean is high when both Acc^+ and Acc^- are high and when their difference is small.

The reason for AREBA's advantage of minority class examples is due to the concept of maintaining separate and balanced queues for each class. Let us first consider the incremental learning algorithms Baseline, Adaptive_CS, OOB_single, and OOB. These algorithms use only the most recent arriving example which they later discard. As a result, under severe or extreme imbalance, they do not experience many minority class examples. Let us now consider Sliding that is a memory-based algorithm as it implements a single sliding window. Under severe or extreme imbalance, this method may still suffer from the same problem if only a small number of minority class examples are found in its sliding window. The problem can be alleviated with a larger window size; however, it would not be able to rapidly react to concept drift. In contrast, the proposed AREBA can afford to have a small window size and, at the same time, experience a sufficient number of minority class examples. AREBA, like all methods, depends on the classifier being able to forget old knowledge quickly enough to react to drift. Therefore, as discussed in Section VI-B, the optimal value of B is application-dependent, and tuning is required. To sum up, these remarks are made.

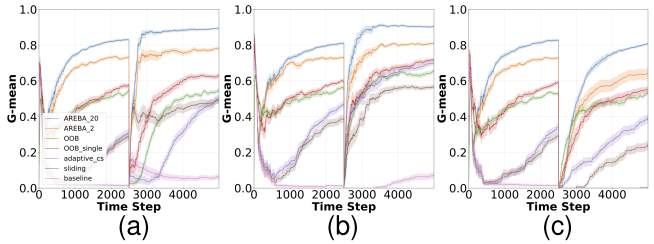

Fig. 11. Comparative study on Sine with imbalance of 1% and different drift types. AREBA is robust to drift, can fully recover, and significantly outperforms other state-of-the-art methods as it had been the case before the drift occurred. (a) Prior drift. (b) Likelihood drift. (c) Posterior drift.

1) AREBA_20 outperforms all algorithms in all imbalance scenarios, while AREBA_2 is the second best. Under extreme class imbalance, AREBA has been shown to perform ten times better than state-of-the-art algorithms.
2) Interestingly, resampling methods (AREBA and OOB_single) seem to better handle online imbalance learning than cost-sensitive learning methods (Adaptive_CS).
3) As the imbalance becomes severe, the performance of all algorithms declines significantly. AREBA has been shown to be affected less seriously, thus being more robust to it.
4) Methods without a mechanism to handle a class imbalance in online learning perform poorly (Baseline and Sliding).

B. Nonstationary Data

We describe our work on nonstationary synthetic data. To examine each drift type independently, we devise the following experiments on the Sine data set based on [3]. In all experiments, the imbalance is set to CI $=$ 1%, i.e., $p(y=1)=0.01$.

A drift in prior probability $p(y)$ occurs, as shown in the following equation:

$$\begin{aligned} p(y=1) &= 0.01 \longrightarrow 0.99 \\ p(y=0) &= 0.99 \longrightarrow 0.01. \end{aligned} \tag{12}$$

A drift in likelihood $p(x|y)$ occurs, as shown in the following equation:

$$\begin{aligned} p(x_1 < 0.6|y=0) &= 0.9 \longrightarrow 0.1 \\ p(x_1 \geq 0.6|y=0) &= 0.1 \longrightarrow 0.9. \end{aligned} \tag{13}$$

A posterior probability drift occurs, as shown in (10). Fig. 11(a)–(c) shows a comparative study on the Sine data set with a prior, class-conditional, and posterior probability drift, respectively. Overall, the proposed AREBA_20 outperforms the rest in all cases, while AREBA_2 is the second best.

The first point to note is about virtual drift; recall that this drift type does not alter the true decision boundary, but it can affect the learned one. Generally, the state-of-the-art (AREBA, OOB_single, and Adaptive_CS) algorithms not only are robust to virtual drift, but an overall slight improvement is observed [see Fig. 11(a) and (b)]. This is because more feature space is

revealed to the algorithm after the drift occurs; for instance, in (13), more examples with $x_1 \geq 0.6$ will be observed.

The second point is about the posterior drift. In this case, the performance of all algorithms declines significantly after the drift [see Fig. 11(c)]. Recall from (10) that the way we defined posterior drift is by performing a "concept swap". Therefore, all algorithms should start relearning the new concept.

The third point is about OOb_single. It outperforms its ensemble version when drift is virtual [see Fig. 11(a) and (b)]. After a posterior drift occurs, OOB is significantly better than OOB_single, but they obtain similar final performance. Again, this is in alignment with [38] as, on a number of occasions, the single classifier has outperformed its ensemble version. From now on, we will consider only OOB_single. To sum up, the following important remarks can be made.

1) AREBA_20 outperforms all algorithms in all drift scenarios, while AREBA_2 is the second best.
2) A drift in $p(y|x)$ is the most severe type of data alteration, and this clearly reflects on all algorithms' performance.
3) Algorithms with a mechanism to address drift (e.g., Sliding) but without one to address imbalance perform poorly.
4) In settings where there is solely drift (no imbalance), AREBA_B and Sliding significantly outperform all algorithms; they are almost identical when $W = 2B$. AREBA_2 and Baseline jointly follow. Adaptive_CS and OOB_single are outperformed by the rest (the figures for these results are not included).

C. Data With Noisy Class Labels

We study each algorithm under conditions where the class label (i.e., ground truth) is incorrectly provided. The study aims at emulating realistic scenarios that can be encountered in deployed settings. One such scenario is when an expert is unable to provide the label (violation of the label availability assumption). Another scenario is when the label can be provided, but not always timely, i.e., before the arrival of the next example (violation of the no verification latency assumption). In these scenarios, an estimated label could be provided by an automated mechanism that may or may not be true.

To model this behavior, we specify a probability threshold by which the true label cannot be provided. The degree to which the aforementioned assumptions are violated in our experiments, i.e., the probability threshold, is 10%. Specifically, with a probability of 10%, we revert the true class label of the arriving example; therefore, each algorithm is not always trained using the ground truth. We repeat the experiments of Fig. 11(a)–(c) with the added noise and present the results in Fig. 12(a)–(c). Similar results for the other synthetic data sets were obtained and are included in the Supplementary Material. These important remarks can be made.

1) In the presence of noise in the class labels, the performance of all algorithms declines substantially.
2) AREBA_20 outperforms the rest in all drift scenarios.

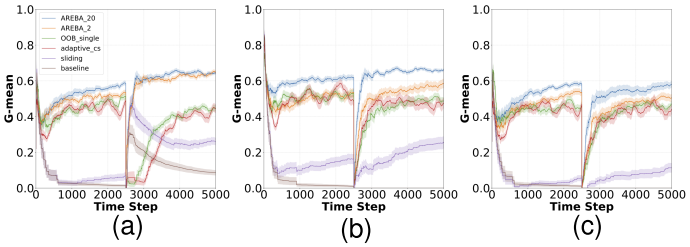

Fig. 12. Comparative study on the Sine data set with class imbalance of CI = 1% and 10% noise in the class labels. (a) Prior drift. (b) Likelihood drift. (c) Posterior drift.

3) Compared with AREBA_2, the performance gap is no longer big; as B gets large, more noisy examples are included in the queues. Interestingly, however, $B = 2$ does not yield the best results. Hence, smaller values of B are suggested, but still some experimentation is necessary.
4) Interestingly, the performance of AREBA_20 in all drift scenarios with 10% noise and 1% (severe) imbalance is G-mean $\simeq 0.6$, which is close to the performance obtained by the proposed algorithm in the case of 0.1% (extreme) imbalance without drift [see Fig. 10(a)].

D. Real-World Data

We describe, now, our work on real-world data. To examine the learning speed, we present the learning curves in Fig. 13(a)–(e). The final mean performances are shown in Table I; the best performing algorithm based on ANOVA and its posthoc tests (see Section V-D) is shown in bold font that denotes statistical significance over the others, and the standard deviation is shown in brackets. The tables with the p-values are found in the Supplementary Material. Following the guidelines derived from our analysis in Section VI-A, we use AREBA_20 in three data sets and AREBA_50 in the other two.

In Fig. 13(a), AREBA_50 achieves G-mean $= 0.8$ at $t \approx 350$, while Adaptive_CS obtains this score at $t \approx 650$. The learning speed is of utmost importance as, in practice, this means that Adaptive_CS would observe about 300 more biopsies to equalize AREBA. In Fig. 13(c), AREBA_20 achieves G-mean $= 0.6$ at $t \approx 50$, while Adaptive_CS obtains this score at $t \approx 450$. In Fig. 13(e), all algorithms equalize AREBA_20 at $t \approx 10\,000$. In Fig. 13(b), AREBA and OOB_single behave similarly.

In Table I and Cervical_Cancer, AREBA_20/50 is joint first with AREBA_2 and outperforms the second Adaptive_CS by more than 9%. In Credit_Score, AREBA_20/50 outperforms the second AREBA_2 by 3.5% and the third Adaptive_CS by 6%. In MNIST, AREBA_20/50 achieves a superior performance, outperforming the second Adaptive_CS by more than 12% and the third AREBA_2 by more than 15%. In Fraud, AREBA_2 outperforms the rest but the improvement over AREBA_20/50, and OOB_single is less than 1%. In Forest Cover Type, all algorithms obtain the same final performance. Surprisingly, Adaptive_CS performs better overall

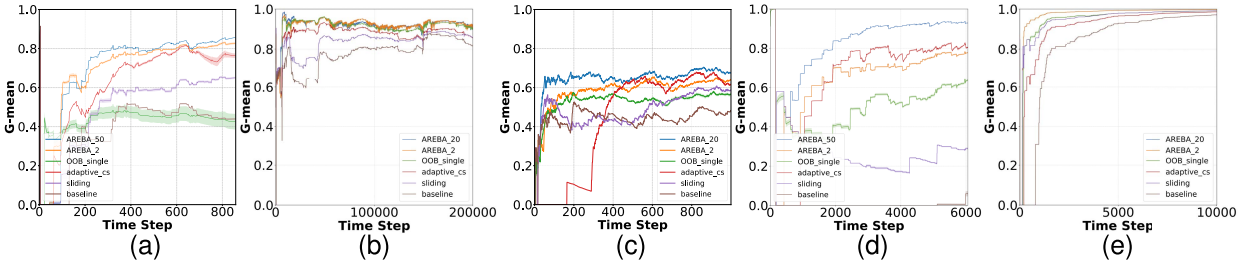

Fig. 13. Comparative study on the real-world data sets. AREBA is the best (or joint best) performing algorithm. (a) Cervical cancer. (b) Fraud. (c) Credit score. (d) MNIST. (e) Forest type.

TABLE I

FINAL PERFORMANCE (G-MEAN) ON THE REAL-WORLD DATA SETS DISPLAYING THE MEAN AND STANDARD DEVIATION

Algorithm	Cervical Cancer	Fraud	Credit Score	MNIST	Forest Cover Type
AREBA_20/50	**0.8555 (0.0104)**	0.8900 (0.0038)	**0.6746 (0.0052)**	**0.9289 (0.0047)**	1.0000 (0.0000)
AREBA_2	**0.8272 (0.0396)**	**0.8985 (0.0000)**	0.6392 (0.0095)	0.7771 (0.0174)	1.0000 (0.0000)
OOB_single	0.4246 (0.2842)	0.8968 (0.0019)	0.5645 (0.061)	0.6388 (0.0313)	1.0000 (0.0000)
Adaptive_CS	0.7627 (0.0848)	0.8564 (0.0000)	0.6141 (0.0023)	0.8051 (0.0252)	1.0000 (0.0000)
Sliding	0.6499 (0.0383)	0.8538 (0.0000)	0.5851 (0.0024)	0.2899 (0.0305)	1.0000 (0.0000)
Baseline	0.4391 (0.0394)	0.8233 (0.0000)	0.4807 (0.0158)	0.0592 (0.0737)	1.0000 (0.0000)

than OOB_single, while the opposite hold true in the synthetic data sets. To sum up, these important remarks can be made.

1) AREBA outperforms other algorithms in all real data sets. These are cases where it would have an impact in practise.
2) In regard to the choice of memory size B, the results are consistent with those observed in our studies with synthetic data. Specifically, AREBA_2 either performs similar to AREBA_20 or closely but worse.

VIII. CONCLUSION AND DISCUSSION

In the following, we discuss important aspects of AREBA, its advantages and limitations, and directions for future work.

A. Dual Nature

AREBA's effectiveness is attributed to a few important characteristics. Maintaining separate and balanced queues for each class helps to address the imbalance problem. Propagating past examples in the most recent training set is viewed as a form of oversampling. The fact that examples are carried over a series of steps allows the classifier to "remember" old concepts. Also, to address the drift challenge, the classifier needs to also be able to "forget" old concepts. This is achieved by AREBA's memory-based nature, i.e., by bounding the length of queues, these are essentially behaving like sliding windows. Hence, AREBA can cope with both imbalance and drift. We have shown that the proposed synergy is seamless, and it significantly outperforms algorithms that belong to resampling or memory-based methods solely and even algorithms that belong to other types, e.g., cost-sensitive methods. Finally, recall that no drift detector can perform satisfactorily under any situation. When domain expertise can foresee the drift's nature, AREBA can work in cooperation with change detection-based methods to complement each other.

B. QBR Versus AREBA

QBR was first introduced in our preliminary study [41] that ran experiments only on synthetic data sets and examined only a limited range of imbalance rates and a single drift type. In this work, we conducted extensive experimental work and stress-tested QBR under conditions of severe imbalance and different drift types. We have identified, discussed, and analyzed QBR's limitations, and under which conditions these occur. It turns out that QBR's limitations arise from the fact that the imbalance problem persists in the queues (as described with reference to Fig. 1). To overcome this, this article proposes AREBA with two major design changes over QBR. First, it allows each queue to be of size B (rather than $(B/2)$). Second, the queues remain balanced throughout the time using a dynamic mechanism (AREBA). These changes allow AREBA to be very effective when dealing with online learning tasks under drift and imbalance.

C. Role of the Memory Size

B's role is of great importance and goes beyond that of just controlling the maximum queue lengths. It controls the "level" of incremental learning. The smaller its value, the closer to being incremental, especially, when $B = 2$ AREBA becomes near-incremental. Finding a suitable value is data set-specific and requires some trial-and-error. To reduce this tedious process, we have provided in Section VI-B some guidelines to help us determine B.

D. Choice of Classifier

AREBA does not impose any restrictions on the selection of the classifier. Some classifiers, however, are more suitable than others for online learning. Our scope was on neural networks that have been shown to work well (e.g., [3] and [26]). Hoeffding Trees have also been shown to work well (e.g., [24]).

Future work will apply AREBA with trees to examine if the observed improvement can generalize.

E. Verification Latency

The learning framework used is suitable for human-in-the-loop learning and assumes that no verification latency exists. Involving humans, however, may cause delays in receiving the labels. In practice, avoiding or reducing potential delays would require mechanisms to collect labels in an automatic manner. We have shown in Section VII-C that AREBA maintains its dual nature benefits and still outperforms other state-of-the-art algorithms in conditions where the assumption is violated. Future work will relax this assumption and examine other paradigms, such as active learning [51].

To conclude, we introduced the novel AREBA algorithm to addresses the problem of class imbalance in nonstationary environments. We provided new interesting insights toward the joint problem of imbalance and concept drift. Our study compared AREBA with the other four baseline and state-of-the-art algorithms and showed that it significantly outperforms them in the vast majority of compared settings.

REFERENCES

[1] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Comput. Intell. Mag.*, vol. 10, no. 4, pp. 12–25, Nov. 2015.

[2] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[3] S. Wang, L. L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4802–4821, Oct. 2018.

[4] 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.

[5] E. Kyriakides and M. Polycarpou, Eds., *Intelligent Monitoring, Control, and Security of Critical Infrastructure Systems*, vol. 565. Berlin, Germany: Springer-Verlag, 2014.

[6] A. D. Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, "Calibrating probability with undersampling for unbalanced classification," in *Proc. IEEE Symp. Ser. Comput. Intell.*, Dec. 2015, pp. 159–166.

[7] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, p. 44, 2014.

[8] K. Fernandes, J. S. Cardoso, and J. Fernandes, "Transfer learning with partial observability applied to cervical cancer screening," in *Proc. Iberian Conf. Pattern Recognit. Image Anal.* Cham, Switzerland: Springer, 2017, pp. 243–250.

[9] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining Knowl. Discovery*, vol. 30, no. 4, pp. 964–994, Jul. 2016.

[10] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Inf. Fusion*, vol. 37, pp. 132–156, Sep. 2017.

[11] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, Apr. 1996.

[12] M. M. Lazarescu, S. Venkatesh, and H. H. Bui, "Using multiple windows to track concept drift," *Intell. Data Anal.*, vol. 8, no. 1, pp. 29–59, Mar. 2004.

[13] R. Klinkenberg, "Learning drifting concepts: Example selection vs. example weighting," *Intell. Data Anal.*, vol. 8, no. 3, pp. 281–300, Aug. 2004.

[14] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, nos. 1–2, pp. 100–115, 1954.

[15] C. Alippi and M. Roveri, "Just-in-time adaptive classifiers—Part I: Detecting nonstationary changes," *IEEE Trans. Neural Netw.*, vol. 19, no. 7, pp. 1145–1153, Jul. 2008.

[16] C. Alippi and M. Roveri, "Just-in-time adaptive classifiers—Part II: Designing the classifier," *IEEE Trans. Neural Netw.*, vol. 19, no. 12, pp. 2053–2064, Dec. 2008.

[17] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2007, pp. 443–448.

[18] C. Alippi, G. Boracchi, and M. Roveri, "Just-in-time classifiers for recurrent concepts," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 4, pp. 620–634, Apr. 2013.

[19] D. Brzezinski and J. Stefanowski, "Ensemble classifiers for imbalanced and evolving data streams," *Ser. Mach. Perception Artif. Intell.*, vol. 83, no. 1, pp. 44–68, 2018.

[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 (KDD)*, 2001, pp. 377–382.

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

[22] 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.

[23] N. C. Oza, "Online bagging and boosting," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, vol. 3, Oct. 2005, pp. 2340–2345.

[24] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 81–94, Jan. 2014.

[25] D. Brzezinski and J. Stefanowski, "Combining block-based and online methods in learning ensembles from concept drifting data streams," *Inf. Sci.*, vol. 265, pp. 50–67, May 2014.

[26] J. Wang, P. Zhao, and S. C. H. Hoi, "Cost-sensitive online classification," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 10, pp. 2425–2438, Oct. 2014.

[27] S. Wang, L. L. Minku, and X. Yao, "A learning framework for online class imbalance learning," in *Proc. IEEE Symp. Comput. Intell. Ensemble Learn. (CIEL)*, Apr. 2013, pp. 36–45.

[28] A. Ghazikhani, R. Monsefi, and H. S. Yazdi, "Recursive least square perceptron model for non-stationary and imbalanced data stream classification," *Evolving Syst.*, vol. 4, no. 2, pp. 119–131, Jun. 2013.

[29] A. Ghazikhani, R. Monsefi, and H. S. Yazdi, "Ensemble of online neural networks for non-stationary and imbalanced data streams," *Neurocomputing*, vol. 122, pp. 535–544, Dec. 2013.

[30] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, pp. 63–77, Jan. 2006.

[31] I. Tomek, "Two modifications of CNN," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, no. 11, pp. 769–772, Nov. 1976.

[32] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[33] G. Douzas and F. Bacao, "Effective data generation for imbalanced learning using conditional generative adversarial networks," *Expert Syst. Appl.*, vol. 91, pp. 464–471, Jan. 2018.

[34] J. Gao, B. Ding, W. Fan, J. Han, and P. S. Yu, "Classifying data streams with skewed class distributions and concept drifts," *IEEE Internet Comput.*, vol. 12, no. 6, pp. 37–49, Nov. 2008.

[35] S. Chen and H. He, "SERA: Selectively recursive approach towards nonstationary imbalanced stream data mining," in *Proc. Int. Joint Conf. Neural Netw.*, Jun. 2009, pp. 522–529.

[36] S. Chen and H. He, "Towards incremental learning of nonstationary imbalanced data stream: A multiple selectively recursive approach," *Evolving Syst.*, vol. 2, no. 1, pp. 35–50, Mar. 2011.

[37] 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.

[38] S. Wang, L. L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1356–1368, May 2015.

[39] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 5, pp. 730–742, May 2010.

[40] R. S. M. Barros and S. G. T. C. Santos, "A large-scale comparison of concept drift detectors," *Inf. Sci.*, vols. 451–452, pp. 348–370, Jul. 2018.

[41] K. Malialis, C. Panayiotou, and M. M. Polycarpou, "Queue-based resampling for online class imbalance learning," in *Proc. Int. Conf. Artif. Neural Netw. (ICANN)*. Cham, Switzerland: Springer, 2018, pp. 498–507.

[42] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. Brazilian Symp. Artif. Intell.* Berlin, Germany: Springer, 2004, pp. 286–295.
[43] H. Hofmann. *German Credit Data*. Accessed: Aug. 31, 2020. [Online]. Available: https://www.kaggle.com/uciml/german-credit/
[44] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[45] J. A. Blackard and D. J. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *Comput. Electron. Agricult.*, vol. 24, no. 3, pp. 131–151, Dec. 1999.
[46] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.
[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.
[48] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 1–4.
[49] M. Kubat, R. Holte, and S. Matwin, "Learning when negative examples abound," in *Proc. Eur. Conf. Mach. Learn.* Berlin, Germany: Springer, 1997, pp. 146–153.
[50] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Mach. Learn.*, vol. 90, no. 3, pp. 317–346, Mar. 2013.
[51] K. Malialis, C. G. Panayiotou, and M. M. Polycarpou, "Data-efficient online classification with siamese networks and active learning," in *Proc. IEEE WCCI Int. Joint Conf. Neural Netw.*, Jul. 2020.

Kleanthis Malialis received the M.Eng. degree in computer systems and software engineering and the Ph.D. degree with a focus on multiagent systems and reinforcement learning from the Department of Computer Science, University of York, York, U.K., in 2010 and 2015, respectively.

His Ph.D. degree was funded by an EPSRC DTA Scholarship. He was a Data Scientist with The Telegraph, London, U.K., where his focus was on building predictive models using machine learning algorithms. He was a Research Associate with the Department of Computer Science, University College London (UCL), London, U.K., as a part of an Innovate UK Knowledge Transfer Partnership between UCL and a data analytics startup. He subsequently joined the startup as a Data Scientist. He is currently a Research Associate and Marie Skłodowska-Curie Fellow with the KIOS Research and Innovation Center of Excellence (KIOS CoE), University of Cyprus (UCY), Nicosia, Cyprus, where he is interested in online machine learning.

Christos G. Panayiotou (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Massachusetts at Amherst, Amherst, MA, USA, in 1994 and 1999, respectively, and the M.B.A. degree from the Isenberg School of Management, University of Massachusetts at Amherst, in 1999.

He is currently a Professor with the Electrical and Computer Engineering Department, University of Cyprus (UCY), Nicosia, Cyprus, where he serves as the Deputy Director of the KIOS Research and Innovation Center of Excellence (KIOS CoE) for which he is also a Founding Member. His research interests include distributed control systems, wireless, *ad hoc* and sensor networks, computer communication networks, quality-of-service (QoS) provisioning, optimization and control of discrete-event systems, resource allocation, simulation, transportation networks, and manufacturing systems.

Dr. Panayiotou is a reviewer for various conferences and journals. He has served on the organizing and program committees of various international conferences.

Marios M. Polycarpou (Fellow, IEEE) received the B.A. degree in computer science and the B.Sc. degree in electrical engineering from Rice University, Houston, TX, USA, in 1987, and the M.S. and Ph.D. degrees in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 1989 and 1992, respectively.

He is currently a Professor of electrical and computer engineering and the Director of the KIOS Research and Innovation Center of Excellence, University of Cyprus, Nicosia, Cyprus. He has published more than 300 articles in refereed journals, edited books and refereed conference proceedings, and coauthored seven books. He holds six patents. His teaching and research interests are in intelligent systems and networks, adaptive and cooperative control systems, computational intelligence, fault diagnosis, and distributed agents.

Prof. Polycarpou is a fellow of International Federation of Automatic Control (IFAC). He was a recipient of the 2016 IEEE Neural Networks Pioneer Award. He received, with his coauthors, the 2014 Best Paper Award for the journal Building and Environment (Elsevier). He has served as the President of the IEEE Computational Intelligence Society from 2012 to 2013 and as the Editor-in-Chief of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS from 2004 to 2010. He is also the President of the European Control Association (EUCA). He has participated in more than 60 research projects/grants, funded by several agencies and industry in Europe and the United States, including the prestigious European Research Council (ERC) Advanced Grant.