Learning in Nonstationary Environments: A Survey

Date of publication: 13 October 2015

I. Introduction

The prevalence of mobile phones, the internet-of-things technology, and networks of sensors has led to an enormous and ever increasing amount of data that are now more commonly available in a streaming fashion [1]–[5]. Often, it is assumed – either implicitly or explicitly – that the process generating such a stream of data is stationary, that is, the data are drawn from a fixed, albeit unknown probability distribution. In many real-world scenarios, however, such an assumption is simply not true, and the underlying process generating the data stream is characterized by an intrinsic nonstationary (or evolving or drifting) phenomenon. The nonstationarity can be due, for example, to seasonality or periodicity effects, changes in the users' habits or preferences, hardware or software faults affecting a cyber-physical system, thermal drifts or aging effects in sensors. In such nonstationary environments, where the probabilistic properties of the data change over time, a non-adaptive model trained under the false stationarity assumption is bound to become obsolete in time, and perform sub-optimally at best, or fail catastrophically at worst.

Given the increasingly common applications that are driven by "nonstationary" or "drifting" data generation processes, the need for effective and efficient algorithms for learning from (and adapting to) evolving or drifting environments can hardly be overstated. Such a need has recently provided a welcome boost to research in learning in nonstationary environments. A comprehensive review of these recent advances is the primary focus of this paper. This survey article serves both as a supplementary as well as a complementary effort to the very short list of other review articles available on concept drift, e.g., [6], [7]. Specifically, we describe the problem of learning in nonstationary environments from two core perspectives: *active* versus *passive* approaches to learning in nonstationary environments. Furthermore, we also cover more recent efforts in the areas of learning from initially labeled nonstationary environments, and learning in nonstationary environments that provide imbalanced data, not previously reviewed elsewhere.

To set the stage, let us start with three real-world examples of applications driven by a nonstationary process:

- *Environmental monitoring and forecasting* involves a network of sensors that collects data from a physical phenomenon and

Gregory Ditzler
Department of Electrical & Computer Engineering at the University of Arizona, Tucson, AZ, USA
e-mail: ditzler@email.arizona.edu

Manuel Roveri and Cesare Alippi
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo Da Vinci, 32, 20133 Milano, ITALY
e-mails: manuel.roveri@polimi.it, cesare.alippi@polimi.it

Robi Polikar
Department of Electrical & Computer Engineering at Rowan University, Glassboro, NJ, USA
e-mail: polikar@rowan.edu

transmits them to a control location for further processing. In most real-world settings, sensing units typically suffer from inevitable aging effects, or faults in their embedded electronics/sensors. In addition, the physical phenomena under monitoring can also evolve with time, e.g., due to seasonality or climate changes. In such settings, the stationarity assumption is simply incorrect.

- *Recommendation systems* provide users with products or services in which they are likely to have an interest, based on their purchasing or browsing history. User interests in products and services can, of course, change due to a variety of reasons such as personal needs, current trends, employment status and age, among others [8], [9]. Building a model for a user and expecting it to be reliable in the distant future is therefore unrealistic. Hence, recommendation systems operate in nonstationary environments, and therefore the model that is providing the recommendations must be able to adapt to the users' changing interests.
- *Predicting energy demand* is one of the most important tasks for the effective operation of the power grid. Historical data are generally available to construct predictive models, but making energy demand predictions is a nonstationary problem due

to a variety of factors that affect supply and demand, such as climate fluctuations that change throughout the year. Energy-demand prediction algorithms must also be able to deal with long-term gradual changes due to, for example, increasing populations, improvements in the efficiency of the energy production, as well as increasingly ubiquitous new disruptive technologies such as electric vehicles and solar powered homes that can return access energy to the grid.

As these examples illustrate, the problem of learning in nonstationary environments – also referred to as learning in dynamic, evolving or uncertain environments, or more commonly as learning "concept drift" – requires novel and effective approaches that can track and adapt to changes in the data generating process.

Against this background, our aim in this paper is twofold. First, we formalize the process of learning in nonstationary environments for classification tasks, and present the broad spectrum of scenarios that can be categorized under the nonstationary environment framework. Second, we describe the two primary families of strategies commonly used for learning concept drift. These two families are generally referred to as *active* and *passive* approaches, terms that are first coined in [10]. They differ in the adaptation mechanism employed to cope with the change: active approaches rely on an explicit detection of the change in the data distribution to activate an adaptation mechanism, while passive approaches continuously update the model over time (without requiring an explicit detection of the change). We present and review commonly cited algorithms from both strategies. Finally, we describe the open problems for current and future research for learning in nonstationary environments, and provide pointers to several freely-available software tools and benchmark datasets to serve as references in hopes of stimulating future research.

II. Learning in Nonstationary Environments as a Framework

A. The Data Generating Process

Let $\mathcal{P}$ be the data generating process providing a sequence of tuples $(\boldsymbol{x}_t, y_t)$ sampled from an unknown probability distribution $p_t(\boldsymbol{x}, y)$, and let $p_t(y|\boldsymbol{x})$ and $p_t(\boldsymbol{x})$ be posterior and evidence distributions, respectively, at some arbitrary time t. The distributions are deliberately subscripted with time t to explicitly emphasize their time-varying nature. In particular, $\boldsymbol{x}_t \in \mathcal{R}^d$ represents a feature vector modeled as a random variable, and $y_t \in \Lambda$ is a discrete class label, both at time t. The data may arrive in an online manner, i.e., one single instance at a time, or in a batch setting. In a single instance setting, only the tuple $\mathcal{S}_t = \{(\boldsymbol{x}_t, y_t)\}$ is provided to the learning algorithm, whereas a batch setting provides a finite set of tuples $\mathcal{S}_t = \{(\boldsymbol{x}_t^1, y_t^1), \ldots, (\boldsymbol{x}_t^N, y_t^N)\}$. Obviously, when $N = 1$ the learning problem would be reduced to a single instance setting. Specific terminology is often used to indicate the cause or nature of changes in the data. In terms of "what" is changing [6], we have:

- *Real Drift*: the posterior probability $p_t(y|\boldsymbol{x})$ varies over time, independently from variations in the evidence $p_t(\boldsymbol{x})$;
- *Virtual Drift*: the evidence or the marginal distribution of the data, $p_t(\boldsymbol{x})$, changes without affecting the posterior probability of classes $p_t(y|\boldsymbol{x})$.

The change in probability distributions can be further broken down with respect to the rate at which the drift is taking place. For example, the concept drift can be abrupt, resulting in a sudden drift, e.g., drift induced by faults affecting a sensor. Such cases are also referred to as "abrupt concept drift" or "concept change." The concept drift can also be gradual [11], [12], which is defined as slowly evolving distributions over time, e.g., as induced by thermal drifts or aging effects in a sensor. Such cases are referred to as "gradual concept drift."

The drifts, whether abrupt or gradual, can be further classified as:

- *Permanent*: the effect of the variation is not limited in time, or
- *Transient*: after a certain amount of time, the effect of the drift disappears.

The types of drifts in the data stream can be further characterized as cyclical or recurrent variations, the latter of which is also known as *recurrent concepts*. In such settings, the ability to retrieve previously acquired knowledge from similar concepts is a valuable and desired quality sought in adaptive algorithms.

B. Algorithmic Considerations for Learning in Nonstationary Environments

There are several important considerations in designing an algorithm for learning and making predictions in nonstationary environments. First, recall that the process $\mathcal{P}$ generates a sequence of data $\mathcal{S}_t$ for $t = 1, 2, \ldots$, assumed to be sampled from potentially different probability distributions. If data are sampled from a potentially infinite (or very long) length sequence, then it is unrealistic to expect that all acquired data can always be available, a consideration that is especially acute with big data applications [13]. A more realistic assumption, which also defines incremental learning, is therefore to accept that $\mathcal{S}_t$ is only available for learning or evaluation at the time first presented to the algorithm [14]–[18], which is also characterized as *one-pass learning*.

Second, most concept drift algorithms expect that the predictions made by the classifier will be verified with the labels for $\mathcal{S}_t$ arriving along with the next training dataset $\mathcal{S}_{t+1}$. This setting allows the algorithm to measure a loss at each time step and is referred to as the *test-then-train* scenario [19], [20], where an evaluation of the previous dataset is conducted prior to training with the next dataset. If the labels do not become available immediately as the next batch of data arrives, this scenario is called "verification latency," the extreme case of which – labels never becoming available beyond the initialization step – leads to "initially labeled environments" as discussed in Section III-C.

Finally, concept drift can also be simply perceived, rather than actual, caused by insufficient, unknown or unobservable attributes, a phenomenon known as "hidden context" [21], or *unknown unknown*. In hidden context, there is a static

underlying process, which is hidden from the learner's view. Having the benefit of the hidden context would remove the nonstationarity. Since the hidden context can never be known, the learner must rely on the aforementioned probabilistic definition of concept drift to describe nonstationary environments.

All these aspects should be taken into account in designing algorithms for learning in nonstationary environments.

C. Related Problems Under the Nonstationary Learning Framework

Learning in nonstationary environments can be seen as a framework, under which several machine learning concepts and problem domains can be listed, as depicted in the mindmap of Figure 1. First, there is the choice of a learning modality, such as supervised, unsupervised, or semi-supervised [22], [23], and the rate at which data arrive (e.g., incremental [14], [24], [25], or in an online manner [26]). Each of these learning modalities would traditionally assume that the data for both training and testing are sampled from a fixed unknown probability distribution. Concept drift detection mechanisms represent effective solution to detect the occurrence of changes within incremental and online learning algorithms (please refer to Section III-A for an analysis of available concept drift detection mechanisms). These different learning modalities by themselves do not necessarily describe a formal nonstationary environment; they are, however, still at the core of learning in a nonstationary setting.

The fields of covariate shift, domain adaption and transfer learning are all characterized by some shift from the training to testing probability distributions, but only for one set of consecutive time instances, rather than a streaming setting. For example, covariate shift describes a perceived change in sampled data distributions between training (source) and test (target) data, without an actual change in the true labeling function, and hence assumes that $p_t(y|\boldsymbol{x}) = p_{t+1}(y|\boldsymbol{x})$, with, $p_t(\boldsymbol{x}) \neq p_{t+1}(\boldsymbol{x})$, where p_t and p_{t+1} denote probability distributions on the source and target [27]–[30].

FIGURE 1 Mindmap of concept drift describing the connections the field has with different areas within machine learning and applications where concept drift can be found.

Transfer learning addresses the issue that training and future data must be in the same feature space, and have the same distribution [31]. In domain adaptation, training and test data are sampled from different but related domains (e.g., in a movie recommendation system, given training data sampled from romantic comedies, the problem is to predict the user interest on test data sampled from dramas) [32]–[34]. These problem domains are subsets of the nonstationary learning problem, as the data distribution changes from training data to test data. However, unlike streaming data examples, there is no notion of continuing time. The source (training) and target (test) data can be interpreted as obtained at time instances $t = 1$ and $t = 2$, respectively, with no future data.

The most general form of a nonstationary environment typically involves streaming or continuously arriving data from time-series applications [35], such as tweet classification [36], or genomics [37], [38], etc. Big Data applications, which may or may not be associated with a streaming time-series data, also constitute one of the major application domains of learning in nonstationary environments. It is important to note that time-series analysis of data does not imply that the learning process is nonstationary.

III. Learning in Nonstationary Environments: Active and Passive Approaches

Adaptation algorithms for learning in the presence of concept drift are primarily based on either an active or passive approach [10], [39]. Algorithms following the active approach specifically aim at detecting concept drift, while algorithms following the passive one continuously update the model every time new data are presented, regardless whether drift is present. Both active and passive approaches intend to provide an up-to-date model; however, the mechanisms used by each to do so are different.

We emphasize that both active and passive approaches can be successful in practice; however, the reason for choosing one approach over the other is typically specific to the application. In fact, before choosing a specific algorithm for learning in a nonstationary environment, it is important to consider the dynamics of the learning scenario (e.g., drift rates, whether the data arrive online or in batches, etc.), computational resources available (e.g., embedded systems or high-performance computers), and any assumptions that can be made about the distributions of the data. In general, passive approaches have been shown to be quite effective in prediction settings with gradual drifts and recurring concepts [10]. While coping with gradual drift can be achieved with active approaches (e.g., see [40]), the change detection with gradual drift is nevertheless more difficult. Active approaches work quite well in settings where the drift is abrupt. In addition, passive approaches are generally better suited for batch learning, whereas active approaches have been shown to work well in online settings as well (e.g., [41]–[43]).

In the following section, we discuss active and passive approaches, and highlight popular implementations of these approaches. A more formal and comprehensive treatment of learning in nonstationary environments can be found in [11].

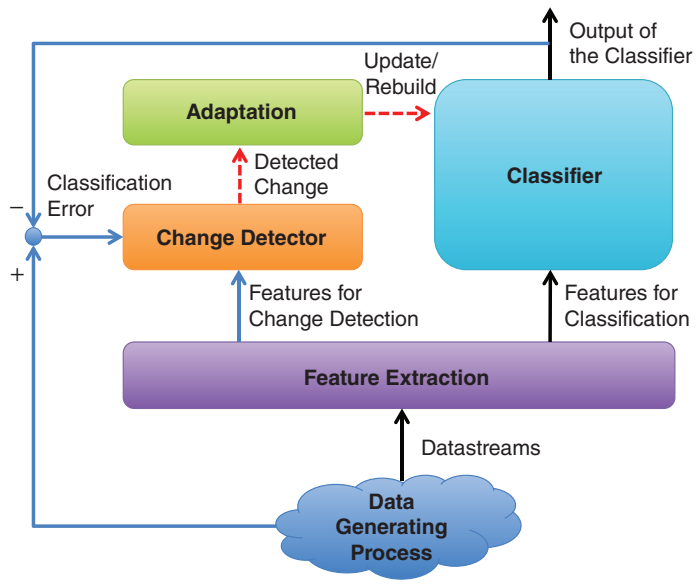

FIGURE 2 Active approach for learning a classifier in nonstationary environments. The fea-ture extraction aims at extracting features from the data-generating process both for change detection and classification. The change detector inspects features extracted for change detection purposes and/or the classification error evaluated over labeled samples. Once a change has been detected, the adaptation phase is activated to update or rebuild the classifier. The black, blue and red dashed lines refer to the classification, the change detection and the adaptation phase, respectively.

A. Active Approaches: Change Detection & Adaptation

As shown in Figure 2, the active approach for learning in presence of concept drift is based on a change detection mechanism that triggers, whenever advisable, an adaptation mechanism aiming at reacting to the detected change by updating or building a new classifier. The change detector aims at asserting "there is a change in the process $\mathcal{P}$" [44] by inspecting features extracted from the data-generating process for change detection purposes and/or analysis of the classification error (evaluated over labeled samples): the analysis of the extracted features monitors the stationarity of the estimated $p_t(\boldsymbol{x})$, whereas the analysis of the classification error aims at detecting variations in the estimated $p_t(y|\boldsymbol{x})$. The adaptation phase, which updates or rebuilds the classification model, is activated only when a change is detected. Adaptive strategies following this mechanism are also known as "detect & react" approaches [11]: once a change is detected, the classifier discards the obsolete knowledge, and adapts to the new environment. Popular change detection mechanisms are reviewed below.

1) *Change Detection*: Change detection mechanisms rarely operate directly on the raw data [45]–[47]. Rather, change detection is typically carried out by inspecting independent and identically distributed (i.i.d.) features extracted from the incoming data stream, e.g., the sample mean, the sample variance [42], [43], [48]–[52], and/or the classification error [41], [43], [52]–[56].

Most existing approaches to detect changes in data generating processes can be grouped into four main families: *Hypothesis Tests*, *Change-Point Methods*, *Sequential Hypothesis Tests*, and *Change Detection Tests*. These families of change detection mechanisms share the ability to inspect variations through theoretically-grounded statistical techniques, but differ in the way data are processed.

The aim of *Hypothesis Tests* (HTs) is to assess the validity of a hypothesis according to a predetermined confidence (e.g., a set of samples has been drawn from a distribution with a specific mean value, two sets of samples have been drawn from two distributions with the same mean value, or two sets of samples have been drawn from the same distribution). These statistical techniques operate on fixed-length sequences (no sequential analysis), and can control the false positive rate in change detection. Examples of HTs applied to the concept drift scenarios can be found in [47], [57]. In particular, the use of the normalized Kolmogorov-Smirnov distance measuring the differences between cumulative density functions estimated on training samples and a window of recent data is suggested in [47]. A change detection mechanism based on the statistical test of equal proportions to inspect variations in the classification error is proposed in [57].

Similarly to HTs, *Change-Point Methods* (CPMs) operate on a fixed data sequence. These statistical techniques [58] aim at verifying whether the sequence contains a change-point (i.e., the time instant the data-generating process changes its statistical behavior) or not by analyzing all possible partitions of the data sequence. The main characteristic of this family of statistical techniques is the ability to jointly address the problems of detecting the presence of a change, and estimating the time instant where the change occurred. The main drawback of such techniques is the high computational complexity that is required to analyze all the partitions of the data sequence, which makes their use in a streaming scenario costly. Approximate formulations of CPMs, meant to work in an online manner, have been recently presented in the literature (e.g., [59]), but the complexity of these solutions remains a significant concern.

Differently from HTs and CPMs that operate on fixed data sequences, *Sequential Hypothesis Tests* (SHTs) are able to sequentially inspect incoming samples (one at the time) up to when a decision to accept or refuse the no-change hypothesis can be taken. In other words, these statistical techniques analyze the stream of data until they have enough statistical confidence to decide either that a "change" or "no change" has occurred. Samples acquired after the decision are not considered. Examples of SHTs are the sequential probability ratio test [60] and the repeated significance test [61]. The main drawback of SHTs resides in the need to make a decision about the null hypothesis (i.e., either change or no-change) once they gain enough statistical confidence. In fact, after the decision, the SHTs stop analyzing the datastreams (once the decision is made by the SHT, there is no need to analyze additional data) and this is a strong limitation in a sequential analysis where the goal is to keep on operating up to when a concept drift affected the data generating process.

The need to operate in a fully sequential manner is addressed by *Change Detection Tests* (CDTs), which are specifically designed to sequentially analyze the statistical behavior of streams of data/features. These methods are usually characterized by a reduced computational complexity (since they have to continuously monitor the data streams), but cannot guarantee a control of the false positive rates (as HTs, CPMs and SHTs do).

The simplest CDT is based on a threshold: a change is detected whenever a feature value or the classification error exceeds the threshold. For example, a fixed threshold based on the Hoeffding bound applied to the difference between sample means of two non-overlapping data windows is suggested in [42], [48]. A different solution is proposed in [54], where the detection of the change is triggered by comparing the validation error computed on the latest data window with the validation error coming from a window of data randomly sampled from previously acquired data.

Another thresholding mechanism based on the classification error is proposed in [53], where the threshold is a function of the variance of the difference between the training and validation error rates. A thresholding mechanism, based on the analysis of the Bernoulli Exponential Weighted Moving Average (EWMA) of errors can be introduced by the last-added classifier as suggested in [55], where the threshold is a function of the proportion of errors of the last-added classifier and a user-defined sensitivity parameter. The mechanism suggested in [41] detects a change when the classification error overcomes a threshold function of the standard deviation of the associated Bernoulli distribution. This mechanism has been extended in [56] by relying on the analysis of the distance between two classification errors (i.e., the current and the lowest value) instead of the proportion of errors. The distance based comparison allows the suggested mechanism to improve the detection performance in cases of slow concept drift. A concept change detection mechanism aiming at assessing variations in the expectation of the classification error between a reference and a sliding detection window is suggested in [62], where the threshold is based on Bernstein bounds. A more effective detection threshold paired with a random sampling mechanism to store samples in the detection window has been presented in [63]. Similarly, a two-moving average mechanisms where the detection thresholds are based on Hoeffding's Bounds is suggested in [64].

The use of the Hellinger distance to measure the distribution divergence between the current data distribution estimated on batches of data and a reference one is suggested in

[In many applications] the fundamental and rather naïve assumption made by most computational intelligence approaches – that the training and testing data are sampled from the same fixed, albeit unknown, probability distribution – is simply not true.

[46] with the adaptive threshold based on the t-statistics. In line with [46], a family of distance measures between distributions (based on the comparison between windows of data) and a threshold-based algorithm to inspect variations in both discrete and continuous distributions is proposed in [45].

While thresholding mechanisms are quite straightforward to design and implement, their main drawback is the difficulty to set the threshold at design time (without assuming any a priori information about the possible changes): too low values may induce many false positive detections, while false negative ones may occur in cases of too large thresholds.

A different approach is suggested in [49], where an adaptive CDT based on the CUmulative SUM (CUSUM) test [44] for monitoring the stationarity of sample mean of the data over time is presented. Here, the log-likelihood ratio between two automatically estimated pdfs (i.e., the null and an alternative pdf) is sequentially evaluated over time to inspect changes in the data-generating process. A computational intelligence extension of the adaptive CUSUM test to inspect variations in sample statistical moment features as well as internal core variables coming from other statistical tests is presented in [50]. The Intersection of Confidence Intervals (ICI) CDT and its variants have been presented in [43], [51], [52], [65]. These CDTs are particularly effective when features are generated by a Gaussian distribution with a fixed variance. ICI CDTs come with a refinement procedure that provides an estimate of the time instant the change occurred (once detected). This ability is crucial for the adaptation phase of Just-in-Time Adaptive Classifiers described in the next subsection.

Interestingly, HTs can be jointly used with CDTs to validate a change detected in a data stream. Change detection mechanisms following this approach are generally referred to as hierarchical CDTs, and are typically able to provide a reduction in false positive detections without increasing the change detection delay [66]. CPMs can also be jointly considered with CDTs within a hierarchical approach. For example, the joint use of the change detection layer based on the ICI CDT and a validation layer based on CPM is suggested in [67].

2) *Adaptation*: Once a change has been detected, the classifier needs to adapt to the change by learning from the newly available information, and discarding the obsolete one. The difficulty consists in designing adaptive mechanisms able to effectively distinguish between obsolete and up-to-date samples. The adaptation mechanisms for active classifiers can be grouped into three main families: windowing, weighting and random sampling.

Windowing is the most common and easiest mechanism. Once a change is detected, a sliding window over the last acquired samples includes only the up-to-date training set for the learner, while previous samples that have fallen out of the window are considered obsolete. Then, all samples within the current window are used to re-train the classifier (or the CDT when needed), whereas older ones are simply discarded. The choice of the appropriate window length is a critical issue and can be determined based on the expected change ratio as suggested in [39], or be adaptive as proposed in [41]–[43], [48], [52], [53], [65]. An adaptive length windowing mechanism based on the analysis of the mean values of subwindows opened on the latest samples is proposed in [42]: the window widens in stationary conditions and shrinks when a change is detected. A detection mechanism based on separate warning and detection thresholds applied to the classification error is suggested in [41], where the length of the window is modified to collect all samples acquired between the instant a feature overcomes the warning threshold and the time instant the detection threshold is exceeded.

A new generation of adaptive classifiers, called Just-In-Time (JIT) adaptive classifiers, able to operate in nonstationary environments is proposed in [43], [52], [65]. These algorithms rely on an adaptive window whose length is estimated through the ICI-based refinement procedure. These algorithms suggested the use of two CDTs to jointly monitor the distributions of the input data and the classification error. In addition, these JIT adaptive classifiers are able to integrate supervised information coming from the data-generating process over time to improve the classification accuracy in stationary conditions. More recently, a JIT adaptive classifier specifically designed to operate with gradual concept drifts has been proposed in [68]. There, a CDT aims at detecting variations in the polynomial trend of the expectation of the data generating process. Once a change has been detected, an adaptive length windowing mechanism based an estimate of the drift dynamics is used to modify the window length.

A pseudocode of the JIT adaptive classifier family is given in Figure 3. An initial training sequence S_{T_0} is used to configure both the classifier and the ICI-based CDT (line 1). After the training phase, when a new sample $\boldsymbol{x}_i$ arrives (with supervised information y_i whenever available), the CDT monitors the stationarity of $\mathcal{P}$ (line 5). If a change is detected at time T, an estimate $\hat{T}$ of the time instance the change occurred is provided by the ICI-based refinement procedure (line 7). All samples acquired before $\hat{T}$ are considered to belong to the previous state of the process and, thus, are discarded. The samples acquired between $\hat{T}$ and T, representing the up-to-date data of the adaptive window, are coherent with the new status, and are used to retrain both the classifier and the CDT (line 8). In stationary conditions, the supervised information $(\boldsymbol{x}_i, y_i)$ is integrated into the classifier to improve (whenever possible) its classification accuracy (line 11).

A hybrid fixed-adaptive approach where the learner is initially trained on a fixed length data window, followed by an adaptation mechanism modifying the window length is suggested in [53].

Differently from windowing approaches, which select a subset of samples from the data stream, weighting mechanisms consider all available samples but, suitably weighted, e.g., according to their age or relevancy with respect to the classification accuracy of the last batch(es) of supervised data [40], [69]–[71]. A gradual-forgetting weighting mechanism is suggested in [69], where the weights of the samples linearly decrease with time (recent samples have larger weights than older ones). Similarly, a time-based weighting mechanism is presented in [70]. There, a set of decay functions for the weights (ranging from polynomial to exponential) is presented and compared. A different approach is presented in [40], where weights depend on a change index measuring the variation of the data-generating process over time (w.r.t. a reference training set). As suggested in [71], samples can also be weighted according to the classification accuracy/error computed on the last batch of supervised data. The main drawback of weighting mechanisms is the need to keep in memory all previously acquired data, an assumption hard to meet in big data applications.

Sampling represents a viable alternative to windowing and weighting. In particular, reservoir sampling [72] is a well known sampling technique (based on randomization) able to select a subset of elements (without replacement) from a data stream. The basis of reservoir sampling is as follows: the sample $(\boldsymbol{x}_t, y_t)$ acquired at time t is stored in the reservoir with a probability $p = k/t$, where k is the user-defined size of the reservoir; if a sample is inserted beyond the reservoir capacity, one randomly selected sample present in the reservoir must be discarded. An example of the use of reservoir sampling in presence of stream evolution can be found in [73], while a reservoir-sampling based change detection mechanism is described in [74].

While ensembles of models are mainly considered in passive approaches (as described in the next section), a few active approaches based on ensemble models are also available in the literature. For example, the idea to create a new model in the ensemble as soon as a triggering mechanism (based on the analysis of the classification error) gets activated is suggested in [55]. JIT adaptation mechanisms have also been proposed in the scenario of ensemble of classifiers [52].

Input: A Training Sequence $\mathcal{S}_{T_0} := \{(\mathbf{x}_i, y_i) : i \in \{1, ..., T_0\}\}$;

1: Configure the classifier and the ICI-based CDT on $\mathcal{S}_{T_0}$;
2: $i = T_0 + 1$;
3: **while** (1) **do**
4: Input receive new data $\mathbf{x}_i$ (with supervised information y_i whenever available);
5: **if** (ICI-based CDT detects a variation in the statistical distribution of inputs or in the classification error) **then**
6: Let T be the time of detection;
7: Activate the ICI-based refinement procedure to provide an estimate $\hat{T}$ (the time the change started);
8: Characterize the new $\mathcal{S}_{\hat{T}}$ as the set of samples acquired between $\hat{T}$ and T;
9: Configure the classifier and the CDT on $\mathcal{S}_{\hat{T}}$;
10: **else**
11: Integrate the available information $(\mathbf{x}_i, y_i)$ in the knowledge base of the classifier;
12: **end if**
13: Predict the output $\hat{y}_i$ of the input samples $\mathbf{x}_i$ (whenever y_i is not available);
14: **end while**

FIGURE 3 An active approach for learning in nonstationary environments: the JIT adaptive classifier.

B. Passive Approaches

As the name indicates, passive approaches do not seek to "actively" detect the drift in the environment, but rather simply accept that the underlying data distributions may (or may not) change at any time with any rate of change. To accommodate the uncertainty in the presence of change, passive approaches perform a continuous adaptation of the model parameters every time new data arrive. The continuous adaptation allows passive approaches to maintain an up-to-date model at all times, thus, avoiding the potential pitfall associated with the active approaches, that is, failing to detect a change or falsely detecting a non-existent change (false alarm).

There are two main categories of passive approaches, those that are based on updating a single classifier and those that add/remove/modify members of an ensemble based system.

1) *Single Classifier Models*: The single classifier approaches generally provide a lower computational cost than an ensemble based approach, which makes single-classifier approaches an attractive solution for massive data stream. Decision trees are the mostly common classifiers used for data stream mining with the very-fast decision tree (VFDT) learner being one of the most popular [75]. The concept drift VFDT (CVFDT) was proposed to cope with a nonstationary data stream by using an adaptive sliding window for training [76]. CVFDT was extended to examine multiple options at each node whenever a node needs to be split [77]. Another single classifier method is the online information network (OLIN), a fuzzy-logic based approach that also exploits a sliding window over the training data stream [78], [79]. More recently, neural networks have also been gaining a renewed popularity for learning in nonstationary environments. For example, a recent work described an online extreme learning machine (ELM) combined with a time-varying neural network for learning from nonstationary data [80].

2) *Ensemble Classifier Models*: Among all passive based approaches for learning in nonstationary environments, ensemble based models appear to be more popular, perhaps with justifiable reasons. Ensemble based approaches provide a natural fit to the problem of learning in a nonstationary setting

Learning in an environment where the labels do not become immediately available is also known as verification latency, and requires a mechanism to propagate class information forward through several time steps of unlabeled data.

and offer some distinct advantages: (i) they tend to be more accurate than single classifier-based systems due to reduction in the variance of the error; (ii) they have the flexibility to easily incorporate new data into a classification model when new data are presented, simply by adding new members to the ensemble; (iii) they provide a natural mechanism to forget irrelevant knowledge, simply by removing the corresponding old classifier(s) from the ensemble [16], [81]. The latter two points can be summarized by the so-called *stability-plasticity dilemma* [82], which refers to the ability of a model either to retain existing knowledge or learn new knowledge, but not being able to do both at the same time equally well. Ensemble-based systems provide a delicate balance along the stability-plasticity spectrum, thanks to their ability to add or remove classifiers (see Figure 4). This quality also makes ensemble systems a good fit for learning in nonstationary environments, as the drift may only impact some subset of the existing knowledge base, while leaving others portions of the previously acquired knowledge still relevant. Ensembles can continuously adapt the voting weights of the classifiers in a strategic manner by measuring the loss of a single model on the most recent data to provide more smaller error rates than a single classifier solution.

The advantage of ensemble based learning in nonstationary environments has also been shown theoretically, specifically proving that ensemble-based systems can provide more stable results than single classifier based approaches in nonstationary settings [83], [84]. Theoretical advantages of ensemble systems have also been shown with respect to their diversity, as the diversity of an ensemble has been of particular interest to the nonstationary learning community. Recent work has shown that using both high and low diversity ensembles can be beneficial for tracking different rates of drift in the data stream [85], [86].

The *streaming ensemble algorithm* (SEA) was one of the earliest examples of the ensemble approaches for learning in nonstationary environments [87]. SEA simply adds new classifiers as new batches of data arrive. Once the ensemble reaches a predetermined size, classifiers are removed from the ensemble, based on a measure of quality of the classifier (e.g., an examination of a single classifier's predictions versus the ensemble's prediction, or simply the age of the classifier). Such a strategy makes it possible for SEA to reduce any effect of the stability-plasticity dilemma. Other similar approaches have also been proposed that follow the "remove the least contributing member" philosophy [81], [88].

Some of the other popular approaches for passive learning include clever modifications of traditional learning algorithms. For example, online bagging & boosting form the basis of online nonstationary boosting algorithm (ONSBoost) [89], which adds an update period to Oza's traditional online boosting [35] to remove classifiers with poor performance. Bifet et al. developed several popular extensions to online bagging/boosting, some of which have integrated techniques from passive and active approaches to track fast and gradual drifts [15], [19]. Dynamic weighted majority (DWM) [90], is an extension of the weighted majority algorithm (WM) [26] that extends WM to data streams with concept drift, and uses an updated period to add/remove classifiers. While sounding similar to ONSBoost, DWM allows for an adaptive ensemble size, whereas ONSBoost has a fixed sized ensemble. Other approaches, such as the accuracy updated ensemble (AUE), follow a similar methodology of examining how to keep/remove classifiers in a fixed ensemble size [91]. Brieman's popular random forest algorithm has also been extended to learning nonstationary data streams, as described in [92].

Another popular batch-based learning algorithm for nonstationary environments is Learn^{++}.NSE (NSE for nonstationary environments) [10], whose pseudocode is shown in Figure 5. Learn^{++}.NSE maintains an ensemble that applies a time-adjusted loss function to favor classifiers that have been performing well in recent times, not just the most recent chunk of data. One of the advantages of the time-adjusted, or discounted, loss is that it allows a classifier that performed poorly a long time ago – and hence previously received a low or zero voting weight – to be reactivated and be given a large current voting weight, if it becomes relevant again, based on its performance on the current environment, perhaps due to a recurring or cyclic drift [83]. The algorithm processes a sequence of datasets $\mathcal{S}_t$, sampled from different, or drifting, probability distributions, $p_t(\boldsymbol{x}, y)$. At each time step Learn^{++}.NSE measures the loss of the existing ensemble on the most recent data in $\mathcal{S}_t$ (line 2 and Equation (1)). Similar to Adaboost [93], Learn^{++}.NSE holds a set of weights over the instances (not to be confused with voting weights

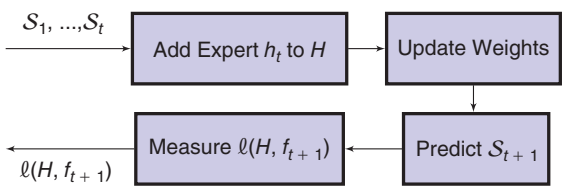

FIGURE 4 High-level block diagram used by incremental learning ensembles in nonstationary environments. Data are received in batches $\mathcal{S}_t$ over time. A classifier h_t is built with the new data, which is then added to the ensemble H. Unlabeled data from S_{t+1} is classified using the ensemble H and a loss is measured when labels from S_{t+1} arrive.

over classifiers) in the data such that a large sampling weight corresponds to instances that are more difficult to classify than those with a low weight (line 3 and Equation (2)). In the context of a drifting distribution, an instance from a new distribution is yet unlearned, and hence difficult to classify with the existing ensemble. Unlike Adaboost, however, when building a new classifier (line 4), Learn^{++}.NSE does not minimize the loss on $\mathcal{S}_t$ according to the weight distribution, but uses the time-adjusted loss (see Equations (3), (4) and (5)), giving the performance on recent times a higher voting weight than the performances at distant past. Specifically, unlike other ensemble approaches that use the most recent loss [87], [94], Learn^{++}.NSE applies a sigmoidal averaging (line 6) to the classifiers' loss history, which then favors classifiers that are performing well in recent times. This time-adjusted loss is one of the key strengths of Learn^{++}.NSE that allows the ensemble decision to be most up to date with recent data. Learn^{++}.NSE time-adjusted loss function has been empirically evaluated against existing ensemble approaches, such as SEA, and the time-adjusted weighting has shown to be quite effective for leveraging stability by recalling previously learned concepts [10], [83]. Unlike many of the other ensemble approaches, Learn^{++}.NSE does not discard old classifiers, but instead simply gives them a dynamic voting weight, which of course allows the classifiers to be reactivated during recurrent concepts. A three-way comparison of age (i.e., time) and error (i.e., accuracy) based weighting that keeps the ensemble size fixed to discounted loss discussed above is described in [95], which showed that retaining all classifiers and simply adjusting their weights using the sigmoidal discounted loss function is preferable over fixed ensemble approaches when classification performance is the most important figure of merit.

Ensemble based approaches have also been applied to other nonstationary learning settings, such as transfer learning and multi-task learning (see Figure 1). For example, Efficient Lifelong Learning Algorithm (ELLA) was proposed to extend the concept of multi-task learning to learn sparsely shared basis for all task models presented over time [96], [97]. Knowledge is transferred from a shared basis of task models to aid in learning new tasks as they are presented.

Input: Datasets $\mathcal{S}_t := \{(\mathbf{x}_i, y_i) : i \in [N_t]\}$, supervised learning algorithm BASE, and parameters a & b.
Initialize: $h_1 = \text{BASE}(\mathcal{S}_1)$ and $W_1^1 = 1$.

1: **for** $t = 2,3, \ldots$ **do**
2: Compute loss of the existing ensemble

$$E_t = \frac{1}{N_t} \sum_{j=1}^{N_t} 1_{H_{t-1}(\mathbf{x}_j) \neq y_j}, \tag{1}$$

where 1_τ evaluates to 1 if $\tau =$ `True` otherwise it is 0.
3: Update instance weights

$$D_t(j) = \frac{1}{Z_t} \begin{cases} E_t & H_{t-1}(\mathbf{x}_j) = y_j \\ 1 & \text{otherwise} \end{cases}, \tag{2}$$

where Z_t is a normalization constant.
4: $h_t = \text{BASE}(\mathcal{S}_t)$
5: Evaluate existing classifiers with new data

$$\varepsilon_k^t = \sum_{j=1}^{N_t} D_t(j) 1_{h_k(\mathbf{x}_j) \neq y_j}. \tag{3}$$

Set $\beta_k^t = \varepsilon_k^t / (1 - \varepsilon_k^t)$.
6: Compute time-adjusted loss

$$\varphi_k^t = \frac{1}{Z_t'} \frac{1}{1 + \exp(-a(t - k - b))}, \tag{4}$$

$$\rho_k^t = \sum_{j=0}^{t-k} \varphi_k^{t-j} \beta_k^{t-j}. \tag{5}$$

7: Update classifier voting weights: $W_k^t = \log \frac{1}{\rho_k^t}$.
8: **end for**

Output: Learn^{++}.NSE's prediction on $\mathbf{x}$

$$H_t(\mathbf{x}) = \arg\max_{\omega \in \Omega} \sum_{k=1}^{t} W_k^t 1_{h_k(\mathbf{x}) = \omega}. \tag{6}$$

FIGURE 5 Learn^{++}.NSE is a passive approach for learning in nonstationary environments.

C. Recent Challenges and Trends in Nonstationary Environments

While learning from a nonstationary data stream is itself challenging, additional constraints, some of which are well known standalone problems of machine learning on their own right, can make the problem even more difficult. For example, class imbalance, which occurs when the number of data instances (or class priors) from different classes are disproportionately different, is a well-studied problem in machine learning [98]–[100]. Furthermore, in class imbalance problems, it is generally the case that the under-represented class is the one that has the higher misclassification cost. While class imbalance has been extensively studied in stationary conditions, the field of nonstationary learning of imbalanced data has received relatively less attention. Uncorrelated bagging is one of the first algorithms to address the joint problem of nonstationary and imbalanced data by considering an ensemble of classifiers trained on under sampled data from the majority class and combining the ensemble models using an average of the classifier outputs [101]–[103]. The Selectively Recursive Approach (SERA) and Recursive Ensemble Approach (REA) are similar approaches to uncorrelated bagging, which use a weighted majority vote [94], [104], [105], though these approaches do require access to historical data. Learn^{++}.CDS (Concept Drift with SMOTE) is a more recent batch-based incremental learning algorithm for imbalanced-nonstationary data streams that does not require access to historical data [106], [107].

Most existing approaches to detect changes in data generating processes can be grouped into four main families: Hypothesis Tests, Change-Point Methods, Sequential Hypothesis Tests, and Change Detection Tests.

More recent works have extended these concepts to online algorithms, such as those in [108], [109]. Developing a true online algorithm for concept drift that does not require access to historical data is extremely challenging due to difficulties associated with measuring minority class statistics without violating the one-pass assumption using only a single instance at a time. A recent effort examined the aforementioned learning problem with multi-label classification [110], however, the field of multi-label classification in nonstationary environments still remains an open area of research [110].

Another set of machine learning topics that are well-established and studied in stationary settings is semi-supervised, unsupervised, transductive and active learning modalities, whose applications to nonstationary environments have only recently been examined. In semi-supervised and transductive learning, unlabeled data from the test set is leveraged to help tune the parameters of the model [111]–[113]. In unsupervised learning/clustering, the learning is done without using labeled data [114]–[116], whereas in active learning (not to be confused with active approaches to learning in nonstationary environments, discussed in Section III-A) the algorithm identifies the most important instances for the learning problem and requests labels for those instances [117].

A particularly challenging form of semi-supervised or unsupervised learning in nonstationary environments involves the very practical scenario, where labeled data are scarce or only available initially, followed by a stream of unlabeled data drawn from a drifting distribution. We refer to such data as *initially labeled nonstationary streaming* (ILNS) data, whose examples include management of power grids, remote-sensing, cyber security and malware detection, and data collection from hazardous or hard to reach locations (e.g., nuclear plants, toxic sites, underground pipelines), where labeled data can be rare or expensive due to human expertise required for annotation.

Learning in an environment where the labels do not become immediately available is also known as *verification latency*, and requires a mechanism to propagate class information forward through several time steps of unlabeled data. Zhang et al. proposed an ensemble approach that combines classifiers and clusters [118], which works well when labeled data are available at least intermittently: labeled data are used to train a classifier, whereas unlabeled data are used to form clusters. New instances are then labeled by a majority vote that includes label mapping between classifiers and clusters of the ensemble. Another approach involves representing each drifting class as a mixture of subpopulations, each drawn from a particular parametric distribution. Given initial labeled data, the subpopulations of the unlabeled data can be tracked and matched to those known sub-populations, as shown in [119], [120], [121], and Krempl's Arbitrary subPopulation Tracker (APT) algorithm [122]. The aforementioned approaches generally assume that (i) the drift is gradual and can be represented as a piecewise linear function; (ii) each subpopulation is present at initialization, whose covariance matrix remains unchanged; and (iii) the rate of drift remains constant. APT involves a two-step process: first, expectation maximization is used to determine the optimal one-to-one assignment between the unlabeled and the drift-adjusted labeled data (based on piecewise linearity of the drift), and then the classifier is updated to reflect the population parameters of newly received data.

Most recently, the COMPOSE framework (COMPacted Object Sample Extraction) was introduced, which can handle multiclass data, including the scenario of new classes or new subpopulations, making only the gradual (limited) drift assumption [123], [124]. Given labeled data only at the initial time step, followed by entirely unlabelled data from a drifting distribution, COMPOSE iteratively: (i) combine initial (or current) labeled data with the new unlabeled data and train a semi-supervised learning (SSL) algorithm to label the unlabeled data; (ii) for each class, form a tight-envelope around the data by using a density estimation approach that can model multi-modal regions, such as α-shapes or Gaussian mixture model; and (iii) compact (shrink) this envelope to obtain the *core support region* of each class from which labeled samples, *core supports*, can be drawn. These samples constitute the new "labeled" instances to be used at the next iteration, and are combined with the new unlabeled data, which are then labeled using the SSL algorithm. COMPOSE is intended for extreme verification latency, where new labeled data is never available. However, if the nonstationary environment provides additional labeled data, perhaps only intermittently, such data can naturally be used to update the core supports, and also help relax or remove the algorithm's limited drift assumption. Furthermore, if the problem domain allows additional labeled data to be requested from the user in an active learning setting, COMPOSE can easily be integrated with an active learning algorithm to take advantage of such an availability [125].

IV. Open Source Software and Available Benchmarks

Many authors have made the code and data used in their publications available to the public. The references provided in this section contain software implementations for algorithms that can learn in nonstationary environments, and data sets that have become standard benchmarks in the field. We do not claim this list to be exhaustive, however, we believe that it provides several opportunities for novices to get started, and established

researchers to expand their contributions, all the while advancing the field by solving some of the open problems described in the next section.

- *Hierarchical ICI-based Change-Detection Tests* (Matlab): Implementation of the hierarchical ICI-based CDT composed of the ICI-based CDT at the detection layer and the Multivariate Hotelling HT at the validation layer [126]. The ICI-based CDT and the Hotelling HT can also be used as stand-alone routines.
  http://home.deib.polimi.it/boracchi/Projects/HierarchicalICIbasedCDT.html
- *Learn^{++}.NSE* (Matlab): Implementation of Learn^{++}.NSE (see Figure 5) with a CART base classifier [10].
  https://github.com/gditzler/IncrementalLearning
- *Massive Online Analysis* (Java): Collection of online supervised, unsupervised and active learning models in Java [127].
  http://moa.cms.waikato.ac.nz/
- *Scalable Advanced Massive Online Analysis* (Java): Collection of distributed algorithms for mining big data streams in Java [128].
  http://jmlr.org/papers/v16/morales15a.html
- *Online Nonstationary Boosting* (Java): Pocock's et al.'s implementation of ONSBoost [89].
  http://www.cs.man.ac.uk/~pococka4/ONSBoost.html

The following datasets and code for generating datasets are commonly used for assessing the performances of proposed concept drift algorithms.

- *Minku & Yao's Concept Drift Generator* (Matlab): Framework for generating synthetic data streams [85].
  http://www.cs.bham.ac.uk/~minkull/opensource.html
- *Kuncheva's Concept Drift Generator* (Matlab): Framework for generating data streams with concept drift [129].
  http://pages.bangor.ac.uk/~mas00a/EPSRC_simulation_framework/changing_environments_stage1a.htm
- *Airlines Flight Delay Prediction*: 100M+ instances contain flight arrival and departure records. The goal is to predict if a flight is delayed.
  http://sourceforge.net/projects/moa-datastream/files/Datasets/Classification/airlines.arff.zip
- *Spam Classification*: Collection of spam & ham emails collected over two years [130].
  http://www.comp.dit.ie/sjdelany/Dataset.htm
- *Chess.com*: Game records for a player over approximately three years [131].
  https://sites.google.com/site/zliobaite/resources-1
- *KDD Cup 1999*: Collection of network intrusion detection data.
  http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
- *POLIMI Rock Collapse and Landslide Forecasting*: Sensor measurements coming from monitoring systems for rock collapse and landslide forecasting deployed on the Italian Alps.
  http://roveri.faculty.polimi.it/software-and-datasets

More software and data – with links provided – can be found at http://github.com/gditzler/ConceptDriftResources and http://roveri.faculty.polimi.it/software-and-datasets.

V. Topics of Future Interest & Conclusions

Learning in nonstationary environments represents a challenging and promising area of research in machine learning and computational intelligence due to its increasing prevalence in real-world applications, which has received a further recent boost with proliferation of streaming and big data applications. In such applications, using traditional approaches that ignore the underlying drift is inevitably bound to fail, necessitating effective algorithms that can track and adapt to changes. In this paper, we provided a survey of the field of learning in nonstationary environments, the associated problems and challenges, and recent developments for addressing those challenges.

While there is now a significant body of work, there are still several open problems in learning in nonstationary environments. Some of these open problems – certainly not an exhaustive list – include the following.

- **Theoretical frameworks for learning**: The field of learning in nonstationary environments can benefit from a more in-depth theoretical analysis of a general framework, where performance bounds can be established with respect to the drift type and rate.
- **Nonstationary consensus maximization** [132]–[134]: Data sets are assumed to be labeled when presented to a supervised algorithm, or unlabeled for an unsupervised one. However, what if the data stream contains a mixture of labeled and unlabeled data? Consensus maximization aims at providing a framework to build and combine multiple supervised and unsupervised models for prediction. One interesting avenue of research is to examine the use of consensus maximization in nonstationary environments.
- **Unstructured and heterogeneous data streams**: One of the central issues with mining from big data is the need to accommodate vast amounts of unstructured and heterogenous data (e.g., texts, images, graphs). Furthermore, the data acquired for learning may have different characteristics, such as multi-dimensionality, multi-label, multi-scale and spatial relationships. The ongoing research on learning in presence of concept drift should include new modeling and adaptive strategies to be able to cope with such data.
- **Definition of limited/gradual drift**: "Limited" or "gradual" drift is one of the primary assumptions commonly made by algorithms for learning in nonstationary environments, particularly for unsupervised or semi-supervised approaches. However, the formal definition of what constitutes limited drift is an elusive one. Not only do we not have established approaches to address those cases when the limited drift assumption is violated, we do not even have a formal definition of the limited drift that follows a concise mathematical formulation. A mathematical definition would allow the community to better understand the limitations of an algorithm in a nonstationary environment.
- **Transient concept drift and limited data**: This setting refers to evolving environments where concept drift is transient, and the number of instances related to the change in stationarity may be very limited. This is particularly challenging

because estimating the features that are used by change detection mechanisms are then computed using a very small sample size, thus, adding an extra level of difficulty to confidently learn the parameters of the nonstationary distribution.

Acknowledgments

This material is based in part upon work supported by the U.S. National Science Foundation under Grant No ECCS-1310496.

References

[1] Z.-H. Zhou, N. V. Chawla, Y. Jin, and G. J. Williams, "Big data opportunities and challenges: Discussions from data analytics perspectives," *IEEE Comput. Intell. Mag.*, vol. 9, no. 4, pp. 62–74, Nov. 2014.
[2] P. Huijse, P. A. Estevez, P. Protopapas, J. C. Principe, and P. Zegers, "Computational intelligence challenges and applications on large-scale astronomical time series databases," *IEEE Comput. Intell. Mag.,* vol. 9, no. 3, pp. 27–39, Aug. 2014.
[3] Y. Zhai, Y.-S. Ong, and I. W. Tsang, "The emerging 'big dimensionality'," *IEEE Comput. Intell. Mag.*, vol. 9, no. 3, pp. 14–26, Aug. 2014.
[4] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowledge Data Eng.*, vol. 26, no. 1, pp. 97–107, Jan. 2014.
[5] National Research Council, *Frontiers in Massive Data Analysis*. Washington, D.C.: National Academies Press, 2013.
[6] A. Tsymbal, "The problem of concept drift: Definitions and related work," Comput. Sci. Dept., Trinity College, Dublin, Ireland, Tech. Rep., Apr. 2004, vol. 106.
[7] J. Gama, I. Žliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, p. 44, Apr. 2014.
[8] D. H. Widyantoro, T. R. Iorger, and J. Yen, "An adaptive algorithm for learning changes in user interests," in *Proc. 8th Conf. Information Knowledge Management*, 1999, pp. 405–412.
[9] D. H. Widyantoro, T. R. Ioerge, and J. Yen, "Tracking changes in user interests with a few relevance judgments," in *Proc. ACM Int. Conf. Information Knowledge Management*, 2003, pp. 548–551.
[10] 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.
[11] C. Alippi, *Intelligence for Embedded Systems*. Berlin, Germany: Springer-Verlag, 2014.
[12] J. Sarnelle, A. Sanchez, R. Capo, J. Haas, and R. Polikar, "Quantifying the limited and gradual concept drift assumption," in *Proc. Int. Joint Conf. Neural Networks*, 2015.
[13] W. Zang, P. Zhang, C. Zhou, and L. Guo, "Comparative study between incremental and ensemble learning on data streams: Case study," *J. Big Data*, vol. 1, no. 5, pp. 1–16, June 2014.
[14] R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. Syst. Man Cybern.*, vol. 31, no. 4, pp. 497–508, Nov. 2001.
[15] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavalda, "New ensemble methods for evolving data streams," in *Proc. Knowledge Data Discovery*, 2009, pp. 139–148.
[16] L. I. Kuncheva, "Classifier ensembles for changing environments," in *Proc. 5th Int. Workshop Multiple Classifier Systems*, 2004, pp. 1–15.
[17] G. Carpenter and S. Grossberg, "ART 2: Self-organization of stable category recognition codes for analog input patterns," *Appl. Opt.*, vol. 26, no. 23, pp. 4919–4930, Dec. 1987.
[18] P. Domingos and G. Hulten, "A general framework for mining massive data streams," *J. Comput. Graph. Stat.*, vol. 12, no. 4, pp. 945–949, 2003.
[19] A. Bifet, G. Holmes, B. Pfahringer, and R. Gavalda, "Improving adaptive bagging methods for evolving data streams," in *Proc. 1st Asian Conf. Machine Learning: Advances Machine Learning*, 2009, pp. 27–37.
[20] A. Bifet, "Adaptive learning and mining for data streams and frequent patterns," Ph.D. dissertations, Universitat Politecnica de Catalunya, Catalunya, Spain, 2009.
[21] 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.
[22] C. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer-Verlag, 2006.
[23] O. Chapelle, B. Scholköpf, and A. Zien, *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006.
[24] G. Carpenter, S. Grossberg, and J. Reynolds, "ARTMAP: A self-organizing neural network architecture for fast supervised learning and pattern recognition," in *Proc. Int. Joint Conf. Neural Networks*, 1991, pp. 863–868.
[25] G. Carpenter, S. Grossberg, N. Markuzon, J. Reynolds, and D. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 698–713, Sept. 1992.
[26] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Inform. Comput.*, vol. 108, no. 2, pp. 212–261, Feb. 1994.
[27] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence, *Dataset Shift in Machine Learning*. Cambridge, MA: MIT Press, 2009.
[28] M. Sugiyama and M. Kawanabe, *Machine Learning in Non-Stationary Environments*. Cambridge, MA: MIT Press, 2012.
[29] M. Sugiyama, S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe, "Direct importance estimation with model selection and its application to covariate shift adaptation," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2008, pp. 1433–1440.
[30] M. Sugiyama, M. Krauledat, and K. R. Müller, "Covariate shift adaptation by importance weighted cross validation," *J. Mach. Learn. Res.*, vol. 8, pp. 985–1005, May 2007.
[31] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowledge Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[32] Y. Mansour, M. Mohri, and A. Rostamizadeh, "Domain adaptation with multiple sources," in *Advances Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2009, pp. 1041–1048.
[33] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Mach. Learn.*, vol. 79, nos. 1–2, pp. 151–175, May 2010.
[34] G. Schweikert, C. Widmer, B. Schölkopf, and G. Rätsch, "An empirical analysis of domain adaptation algorithms for genomic sequence analysis," in Proc. *Advances Neural Information Processing Systems*, 2009, pp. 1433–1440.
[35] N. Oza, "On-line ensemble learning," Ph.D. dissertation, Univ. California, Berkeley, CA, 2001.
[36] A. Bifet and E. Frank, "Sentiment knowledge discovery in Twitter streaming data," in *Proc. Int. Conf. Discovery Science*, 2010, pp. 1–15.
[37] J. G. Caporaso, C. L. Lauber, E. K. Costello, D. Berg-Lyons, A. Gonzalez, J. Stombaugh, D. Knights, P. Gajer, J. Ravel, N. Fierer, J. Gordon, and R. Knight, "Moving pictures of the human microbiome," *Genome Biol.*, vol. 12, no. 5, p. R50, 2011.
[38] N. Fierer and J. Ladau, "Predicting microbial distributions in space and time," *Nature Methods*, vol. 9, no. 6, pp. 549–551, 2012.
[39] 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.
[40] C. Alippi, G. Boracchi, and M. Roveri, "Just in time classifiers: Managing the slow drift case," in *Proc. Int. Joint Conf. Neural Networks*, 2009, pp. 114–120.
[41] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. Advances Artificial Intelligence–SBIA*, 2004, pp. 286–295.
[42] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proc. SIAM Int. Conf. Data Mining*, 2007.
[43] 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.
[44] M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes: Theory and Application*, vol. 104. Englewood Cliffs, NJ: Prentice Hall, 1993.
[45] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *Proc. 30th Int. Conf. Very Large Data Bases*, 2004, vol. 30, pp. 180–191.
[46] G. Ditzler and R. Polikar, "Hellinger distance based drift detection for nonstationary environments," in *Proc. IEEE Symp. Computational Intelligence Dynamic Uncertain Environments*, 2011, pp. 41–48.
[47] J. P. Patist, "Optimal window change detection," in *Proc. 7th IEEE Int. Conf. Data Mining Workshops*, 2007, pp. 557–562.
[48] A. Bifet and R. Gavalda, "Kalman filters and adaptive windows for learning in data streams," in *Proc. Int. Conf. Discovery Science*, 2006, pp. 29–40.
[49] C. Alippi and M. Roveri, "An adaptive cusum-based test for signal change detection," in *Proc. Int. Symp. Circuits Systems*, 2006, pp. 1–4.
[50] 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, July 2008.
[51] C. Alippi, G. Boracchi, and M. Roveri, "Change detection tests using the ICI rule," in *Proc. Int. Joint Conf. Neural Networks*, 2010, pp. 1–7.
[52] C. Alippi, G. Boracchi, and M. Roveri, "Just-in-time ensemble of classifiers," in *Proc. Int. Joint Conf. Neural Networks*, 2012, pp. 1–8.
[53] L. Cohen, G. Avrahami-Bakish, M. Last, A. Kandel, and O. Kipersztok, "Real-time data mining of non-stationary data streams from sensor networks," *Inform. Fusion*, vol. 9, no. 3, pp. 344–353, July 2008.
[54] M. Harel, K. Crammer, R. El-Yaniv, and S. Mannor, "Concept drift detection through resampling," in *Proc. Int. 31st Conf. Machine Learning*, 2014. pp. 1009–1017.
[55] K. Nishida and K. Yamauchi, "Learning, detecting, understanding, and predicting concept changes," in *Proc. Int. Joint Conf. Neural Networks*, 2009, pp. 2280–2287.
[56] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno, "Early drift detection method," in *Proc. 4th Int. Workshop Knowledge Discovery from Data Streams*, 2006, pp. 1–4.
[57] K. Nishida and K. Yamauchi, "Detecting concept drift using statistical testing," in *Discovery Science*. Berlin, Germany: Springer-Verlag, 2007, pp. 264–269.
[58] D. M. Hawkins, Q. Peihua, and W. K. Chang, "The changepoint model for statistical process control," *J. Qual. Technol.*, vol. 35, no. 4, pp. 355–366, Oct. 2003.
[59] G. J. Ross, D. K. Tasoulis, and N. M. Adams, "Nonparametric monitoring of data streams for changes in location and scale," *Technometrics*, vol. 53, no. 4, pp. 379–389, 2011.
[60] A. Wald, "Sequential tests of statistical hypotheses," *Ann. Math. Stat.*, vol. 16, no. 2, pp. 117–186, June 1945.
[61] P. Armitage and P. Armitage, *Sequential Medical Trials*. Oxford, U.K.: Blackwell, 1975.
[62] S. Sakthithasan, R. Pears, and Y. S. Koh, "One pass concept change detection for data streams," in *Advances in Knowledge Discovery and Data Mining*, Berlin, Germany: Springer-Verlag, 2013, pp. 461–472.
[63] R. Pears, S. Sakthithasan, and Y. S. Koh, "Detecting concept change in dynamic data streams," *Mach. Learn.*, vol. 97, no. 3, pp. 259–293, Jan. 2014.
[64] I. Frías-Blanco, J. del Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds," *IEEE Trans. Knowledge Data Eng.*, vol. 27, no. 3, pp. 810–823, Aug. 2014.

[65] C. Alippi, G. Boracchi, and M. Roveri, "A just-in-time adaptive classification system based on the intersection of confidence intervals rule," *Neural Netw.*, vol. 24, no. 8, pp. 791–800, Oct. 2011.
[66] C. Alippi, G. Boracchi, and M. Roveri, "A hierarchical, nonparametric, sequential change-detection test," in *Proc. Int. Joint Conf. Neural Networks*, 2011, pp. 2889–2896.
[67] G. Boracchi, M. Michaelides, and M. Roveri, "A cognitive monitoring system for contaminant detection in intelligent buildings," in *Proc. Int. Joint Conf. Neural Networks*, July 2014, pp. 69–76.
[68] C. Alippi, G. Boracchi, and M. Roveri, "An effective just-in-time adaptive classifier for gradual concept drifts," in *Proc. Int. Joint Conf. Neural Networks*, 2011, pp. 1675–1682.
[69] I. Koychev, "Gradual forgetting for adaptation to concept drift," in *Proc. ECAI Workshop Current Issues Spatio-Temporal Reasoning*, 2000, pp. 101–106.
[70] E. Cohen and M. Strauss, "Maintaining time-decaying stream aggregates," in *Proc. 22nd ACM SIGMOD-SIGACT-SIGART Symp. Principles Database Systems*, 2003, pp. 223–233.
[71] R. Klinkenberg, "Learning drifting concepts: Example selection vs. example weighting," *Intell. Data Anal.*, vol. 8, no. 3, pp. 281–300, Aug. 2004.
[72] J. S. Vitter, "Random sampling with a reservoir," *ACM Trans. Math. Softw.*, vol. 11, no. 1, pp. 37–57, Mar. 1985.
[73] C. C. Aggarwal, "On biased reservoir sampling in the presence of stream evolution," in *Proc. 32nd Int. Conf. Very Large Data Bases*, 2006, pp. 607–618.
[74] W. Ng and M. Dash, "A test paradigm for detecting changes in transactional data streams," in *Database Systems for Advanced Applications*. Berlin, Germany: Springer-Verlag, 2008, pp. 204–219.
[75] P. Domingos and G. Hulton, "Mining high-speed data streams," in *Proc. 6th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, 2000, pp. 71–80.
[76] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proc. Conf. Knowledge Discovery Data*, 2001, pp. 97–106.
[77] J. Liu, X. Li, and W. Zhong, "Ambiguous decision trees for mining concept-drifting data streams," *Pattern Recognit. Lett.*, vol. 30, no. 15, pp. 1347–1355, Nov. 2009.
[78] L. Cohen, G. Avrahami, M. Last, and A. Kandel, "Info-fuzzy algorithms for mining dynamic data streams," *Appl. Soft Comput.*, vol. 8, no. 4, pp. 1283–1294, Sept. 2008.
[79] L. Cohen, G. Avrahami-Bakish, M. Last, A. Kandel, and O. Kipersztok, "Real-time data mining of non-stationary data streams from sensor networks," *Inform. Fusion*, vol. 9, no. 3, pp. 344–353, July 2008.
[80] Y. Ye, S. Squartini, and F. Piazza, "Online sequential extreme learning machine in nonstationary environments," *Neurocomputing*, vol. 116, pp. 94–101, Sept. 2013.
[81] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, "Dynamic integration of classifiers for handling concept drift," *Inform. Fusion*, vol. 9, no. 1, pp. 56–68, Jan. 2008.
[82] S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architectures," *Neural Netw.*, vol. 1, no. 1, pp. 17–61, 1988.
[83] G. Ditzler, G. Rosen, and R. Polikar, "Discounted expert weighting for concept drift," in *Proc. IEEE Symp. Computational Intelligence Dynamic Uncertain Environments*, 2013, pp. 61–67.
[84] G. Ditzler, G. Rosen, and R. Polikar, "Domain adaptation bounds for multiple expert systems under concept drift," in *Proc. Int. Joint Conf. Neural Networks*, 2014, pp. 595–601.
[85] 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. Knowledge Data Eng.*, vol. 22, no. 5, pp. 731–742, May 2010.
[86] L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE Trans. Knowledge Discovery Data Eng.*, vol. 24, no. 4, pp. 619–633, Apr. 2012.
[87] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proc. 7th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, 2001, pp. 377–382.
[88] M. Scholz and R. Klinkenberg, "An ensemble classifier for drifting concepts," in *Proc. 2nd Int. Workshop Knowledge Discovery Data Streams*, 2005, pp. 53–64.
[89] A. Pocock, P. Yiapanis, J. Singer, M. Lujan, and G. Brown, "Online nonstationary boosting," in *Proc. Int. Workshop Multiple Classifier Systems*, 2010, pp. 205–214.
[90] J. Kolter and M. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, Dec. 2007.
[91] D. Brzezinski and J. Stephanowski, "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.
[92] H. Abdulsalam, D. Skillicorn, and P. Martin, "Classification using streaming random forests," *IEEE Trans. Knowledge Data Eng.*, vol. 23, no. 1, pp. 22–36, Jan. 2011.
[93] Y. Freund and R. Shapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, pp. 119–139, Aug. 1997.
[94] S. Chen, H. He, K. Li, and S. Sesai, "MuSeRA: Multiple selectively recursive approach towards imbalanced stream data mining," in *Proc. Int. Joint Conf. Neural Networks*, 2010, pp. 2857–2864.
[95] R. Elwell and R. Polikar, "Incremental learning in nonstationary environments with controlled forgetting," in *Proc. Int. Joint Conf. Neural Networks*, 2009, pp. 771–778.
[96] P. Ruvolo and E. Eaton, "ELLA: An efficient lifelong learning algorithm," in *Proc. Int. Conf. Machine Learning*, 2013, pp. 507–515.
[97] P. Ruvolo and E. Eaton, "Scalable lifelong learning with active task selection," in *Proc. AAAI Conf. Artificial Intelligence*, 2013, pp. 33–39.
[98] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Data Knowledge Discovery*, vol. 12, no. 9, pp. 1263–1284, Sept. 2009.
[99] 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, June 2002.
[100] N. V. Chawla, N. Japkowicz, and A. Kolcz, "Editorial: Special issue on learning from imbalanced data sets," *SIGKDD Expl.*, vol. 6, no. 1, pp. 1–6, June 2004.
[101] 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.–Dec. 2008.
[102] 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. 203–208.
[103] K. Wu, A. Edwards, W. Fan, J. Gao, and K. Zhang, "Classifying imbalanced data streams via dynamic feature group weighting with importance sampling," in *Proc. SIAM Int. Conf. Data Mining*, 2014, pp. 722–730.
[104] S. Chen and H. He, "SERA: Selectively recursive approach towards nonstationary imbalanced stream data mining," in *Proc. Int. Joint Conf. Neural Networks*, 2009, pp. 552–529.
[105] 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.
[106] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Trans. Knowledge Data Eng.*, vol. 25, no. 10, pp. 2283–2301, Oct. 2013.
[107] G. Ditzler and R. Polikar, "An incremental learning framework for concept drift and class imbalance," in *Proc. Int. Joint Conf. Neural Networks*, 2010, pp. 736–473.
[108] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," *IEEE Trans. Reliab.*, vol. 62, no. 2, pp. 434–443, June 2013.
[109] S. Wang, L. L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *IEEE Trans. Knowledge Data Eng.*, vol. 27, no. 5, pp. 1356–1368, May 2015.
[110] E. S. Xioufis, M. Spiliopoulou, G. Tsoumakas, and I. Vlahavas, "Dealing with concept drift and class imbalance in multi-label stream classification," in *Proc. Int. Joint Conf. Artificial Intelligence*, 2011, pp. 1583–1588.
[111] K. Yamauchi, "Incremental learning and model selection under virtual concept drifting environments," in *Proc. Int. Joint Conf. Neural Networks*, 2010, pp. 1–8.
[112] G. Ditzler, G. Rosen, and R. Polikar, "Transductive learning algorithms for nonstationary environments," in *Proc. Int. Joint Conf. Neural Networks*, 2012, pp. 1–8.
[113] G. Ditzler and R. Polikar, "Semi-supervised learning in nonstationary environments," in *Proc. Int. Joint Conf. Neural Networks*, 2011, pp. 2741–2748.
[114] M. Ackermann, C. Lammersen, M. Märtens, C. Raupach, C. Sohlerand, and K. Swierkot, "StreamKM++: A clustering algorithms for data streams," in *Proc. 12th Workshop Algorithm Engineering Experiments*, 2010, pp. 1–31.
[115] C. Aggarwal, J. Han, J. Wang, and P. Yu, "A framework for clustering evolving data streams," in *Proc. 29th Int. Conf. Very Large Data Bases*, 2003, pp. 81–92.
[116] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proc. SIAM Conf. Data Mining*, 2006, pp. 328–339.
[117] I. Zliobaite, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 1, pp. 27–39, Jan. 2014.
[118] P. Zhang, X. Zhu, J. Tan, and L. Guo, "Classifier and cluster ensembles for mining concept drifting data streams," in *Proc. Int. Conf. Data Mining*, 2010, pp. 1175–1180.
[119] G. Krempl and V. Hofer, "Classification in presence of drift and latency," in *Proc. IEEE Int. Conf. Data Mining Workshops*, 2011, pp. 596–603.
[120] R. Alaiz-Rodriguez, A. Guerrero-Curieses, and J. Cid-Sueiro, "Class and subclass probability re-estimation to adapt a classifier in the presence of concept drift," *Neurocomputing*, vol. 74, no. 16, pp. 2614–2623, Sept. 2011.
[121] V. Hofer and G. Krempl, "Drift mining in data: A framework for addressing drift in classification," *Comput. Stat. Data Anal.*, vol. 57, no. 1, pp. 377–391, Jan. 2013.
[122] G. Krempl, "The algorithm apt to classify in concurrence of latency and drift," *Adv. Intell. Data Anal.*, vol. 7014, pp. 222–233, 2011.
[123] K. Dyer and R. Polikar, "Semi-supervised learning in initially labeled non-stationary environments with gradual drift," in *Proc. Int. Joint Conf. Neural Networks*, 2012, pp. 1–9.
[124] K. B. Dyer, R. Capo, and R. Polikar, "COMPOSE: A semi-supervised learning framework for initially labeled non-stationary streaming data," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 1, pp. 12–26, Jan. 2013.
[125] R. Capo, K. B. Dyer, and R. Polikar, "Active learning in nonstationary environments," in *Proc. Int. Joint Conf. Neural Networks*, 2013, pp. 1–8.
[126] C. Alippi, G. Boracchi, and M. Roveri, "A hierarchical, nonparametric, sequential change-detection test," in *Proc. Int. Joint Conf. Neural Networks*, 2011, pp. 2889–2896.
[127] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive online analysis," *J. Mach. Learn. Res.*, vol. 11, pp. 1601–1604, Mar. 2010.
[128] G. D. F. Morales and A. Bifet, "SAMOA: Scalable advanced massive online analysis," *J. Mach. Learn. Res.*, vol. 16, pp. 149–153, Jan. 2015.
[129] A. Narasimhamurthy and L. I. Kuncheva, "A framework for generating data to simulate changing environments," in *Proc. IASTED Artificial Intelligence Applications*, 2007, pp. 384–389,
[130] S. Delany, P. Cunningham, and A. Tsymbal, "A comparison of ensemble and case-base maintenance techniques for handling concept drift in spam filtering," in *Proc. Int. Conf. Artificial Intelligence*, 2006, pp. 340–345.
[131] I. Žliobaité, "Change with delayed labeling: When is it detectable?" in *Proc. IEEE Int. Conf. Data Mining Workshops*, 2010, pp. 843–850.
[132] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han, "Graph-based consensus maximization among multiple supervised and unsupervised models," in *Proc. Advances Neural Information Processing Systems*, 2009, pp. 585–593.
[133] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han, "Graph-based consensus maximization among multiple supervised and unsupervised models," *IEEE Trans. Knowledge Discovery Data Eng.*, vol. 25, no. 1, pp. 15–28, Oct. 2011.
[134] S. Xie, J. Gao, W. Fan, D. Turaga, and P. S. Yu, "Class-distribution regularized consensus maximization for alleviating overfitting in model combination," in *Proc. 20th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, 2014, pp. 303–312.