Noname manuscript No.
(will be inserted by the editor)

Learning from Streaming Data with Concept Drift and Imbalance: An Overview

T. Ryan Hoens · Robi Polikar · Nitesh V. Chawla

the date of receipt and acceptance should be inserted later

Abstract The primary focus of machine learning has traditionally been on learning from data assumed to be sufficient and representative of the underlying fixed, yet unknown, distribution. Such restrictions on the problem domain paved the way for development of elegant algorithms with theoretically provable performance guarantees. As is often the case, however, real world problems rarely fit neatly into such restricted models. For instance class distributions are often skewed, resulting in the "class imbalance" problem. Data drawn from non-stationary distributions is also common in real world applications, resulting in the "concept drift" or "non-stationary learning" problem which is often associated with streaming data scenarios. Recently, these problems have independently experienced increased research attention, however the combined problem of addressing all of the above mentioned issues has enjoyed relatively little research. If the ultimate goal of intelligent machine learning algorithms is to be able to address a wide spectrum of real world scenarios, then the need for a general framework for learning from, and adapting to, a non-stationary environment that may introduce imbalanced data can be hardly overstated. In this paper, we first present an overview of each of these challenging areas, followed by a comprehensive review of recent research for developing such a general framework.

T. Ryan Hoens
Department of Computer Science and Engineering,
University of Notre Dame,
Notre Dame, IN 46556
E-mail: thoens@cse.nd.edu

Robi Polikar
Electrical and Computer Engineering,
Rowan University,
Glassboro, NJ 08028
E-mail: polikar@rowan.edu

Nitesh V. Chawla Department of Computer Science and Engineering,
University of Notre Dame,
Notre Dame, IN 46556
E-mail: nchawla@cse.nd.edu

1 Introduction

Classification is one of the most widely studied problems in the data mining and machine learning communities. Traditionally, the classification problem consists of attempting to learn concepts from a static dataset, the instances of which belong to an underlying distribution defined by a generating function. This dataset is therefore assumed to contain all information necessary to learn the relevant concepts pertaining to the underlying generating function. This model, however, has proven unrealistic for many real world scenarios, e.g., intrusion detection, spam detection, fraud detection, loan recommendation, climate data analysis, long term epidemiological studies, etc. Instead of all training data being available from the start, data is often received over time in streams of instances or batches. Such data traditionally arrives in one of two different ways (shown in Figure 1), either *incrementally* (e.g., hourly temperature readings as in Figure 1(a)), or in *batches* (e.g., daily internet usage dumps as in Figure 1(b)). The challenge is then to use all the information up to a specific time step t to predict new instances arriving at time step $t + 1$.

Learning under such conditions is known as *incremental learning*. While a variety of definitions

for incremental learning exist in the literature, we propose a general definition due to Muhlbaier, Topalis, and Polikar [62], outlined by several authors [35, 51, 30, 52]. Namely, a learning algorithm is incremental if, for a sequence of training instances (potentially batches of instances), it satisfies the following criteria:

1. it produces a sequence of hypotheses such that the current hypothesis describes all data seen thus far.
2. it only depends on the current training data and a limited number of previous hypotheses.

Given this definition, learning from such data streams requires a classifier that can be updated incrementally in order to leverage the newly available data, while simultaneously maintaining the performance of the classifier on old data. The competing motivations of this goal give rise to the *stability-plasticity dilemma* [36], which asks how a learning system can be designed to remain stable and unchanged to irrelevant events (e.g., outliers), while plastic (i.e., adaptive) to new, important data (e.g., changes in concepts).

Therefore, the stability-plasticity dilemma represents a continuum on which incremental learning classifiers can exist. On the stability end of the continuum are *batch learning* algorithms (i.e., algorithms trained on a single batch of data). Batch learners ignore all new data, instead focusing entirely on previously learned concepts. On the other end of the continuum are *online learning* [67] algorithms, where the model is adapted immediately upon seeing the new instance, and the instance is then immediately discarded.

While batch learners exist at one end of the continuum of the stability-plasticity dilemma, they are not, by definition, incremental learners. This is due to the fact that batch learners are incapable of describing any new instances once they have been learned, and thus fail criterion 1. When used as incremental learning algorithms, this limitation is often mitigated by creating ensembles of batch learners, where new batch learners can be learned on the new data, and then combined through a voting mechanism.

1.1 Contributions

In addition to the stability-plasticity dilemma, presenting the data in a stream can lead to new challenges that must be addressed. We begin by defining these challenges (Section 2), and demonstrating how they affect learning in data streams.

We also aim to provide a comprehensive overview of the work done to combat the class imbalance problem in data streams (defined more formally in Section 2.2) which exhibit concept drift (defined in Section 2.1). We also hope to spur research in this field, as we will demonstrate that there is a distinct lack of research into the problem.

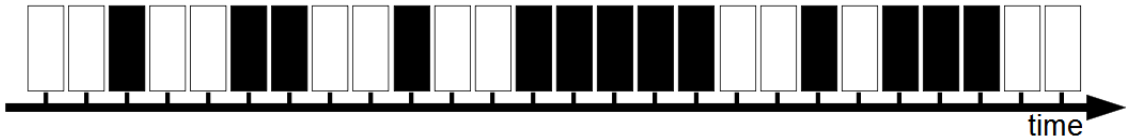
Finally, this work aims to be a complement to the work by Moreno-Torres and Herrera [58], focusing mainly on the underlying concept drift problem, and highlighting research which pays special attention to the class imbalance problem.

2 Challenges of Learning in Data Streams

One of the main assumptions of traditional data mining is that each dataset is generated from a single, static, hidden function. That is, the function generating data for training is the same as that used for testing. In the streaming data model this need not be true, i.e., the function which generates instances at time step t need not be the same function as the one that which generates instances at time step $t + 1$. This (potential) variation in the underlying function is known as *concept drift*, whose formal definition is provided below.

The major assumption with concept drift is that the (hidden) generating function of the new data is unknown to the learner, and hence the concept drift is unpredictable. If the generating function for the drifting concepts was known, one could merely learn an appropriate classifier for each relevant concept, and apply the correct classifier for all new data (which is known as the multitask learning problem). In the absence of such knowledge, then, we must design a unified classifier that can handle such changes in concepts over time.

Another challenge arises when it is assumed that the prevalence of each class in the dataset is, and will remain, equivalent. While class prevalence in traditional data mining problems remains constant, such an assumption is particularly impractical in streaming data applications, where the class distributions can become severely imbalanced. Thus the positive (rare) events, which are already underrepresented in a static dataset, can become even more severely underrepresented in streaming data. Hence, when combined with potential concept drift, class imbalance poses a significant challenge that needs to be addressed by any algorithm that proposes to deal with learning from streaming data.



(a) Incremental

(b) Batch

(c) Static

Fig. 1 Graphical representation of the 3 different different types of datasets, where each rectangle represents an instance, and the color of the instance represents the class. Data arrives at each of the tick marks, and instances outlined in gray denote a "batch" of instances all arriving simultaneously.

2.1 Concept Drift

When learning from data streams, we assume that at time step t the learning algorithm $\mathcal{L}$ is presented with a set of labeled instances $\{\mathbf{X}_0, \ldots, \mathbf{X}_t\}$, where $\mathbf{X}_i$ is a p-dimensional feature vector and each instance has a corresponding class label y_i. Given an unlabeled instance X_{t+1}, the learning algorithm provides a (potentially probabilistic) class label for $\mathbf{X}_{t+1}$. Once the label is predicted, we assume that the true label y_{t+1} and a new testing instance $\mathbf{X}_{t+2}$ become available for testing. Furthermore, we call the hidden function f generating the instance at time t as f_t.

Concept drift is said to occur when the underlying function (f) changes over time. There are multiple ways in which this change can occur. Consider classifying $\mathbf{X}_{t+1}$: in order to optimally classify $\mathbf{X}_{t+1}$, we need to know two pieces of information. First, the prior probability of observing each class, $p(c_i)$, and second, the conditional probability of observing $\mathbf{X}_{t+1}$ given each class, $p(\mathbf{X}_{t+1}|c_i)$. Bayes' theorem then allows us to compute the probability that $\mathbf{X}_{t+1}$ is an instance of class c_i as:

$$p(c_i|\mathbf{X}_{t+1}) = \frac{p(c_i)p(\mathbf{X}_{t+1}|c_i)}{p(\mathbf{X}_{t+1})}, \quad (1)$$

where $p(\mathbf{X}_{t+1})$ is the probability of observing $\mathbf{X}_{t+1}$. Note, however, that $p(\mathbf{X}_{t+1})$ is constant for all classes c_i, and can thus be ignored.

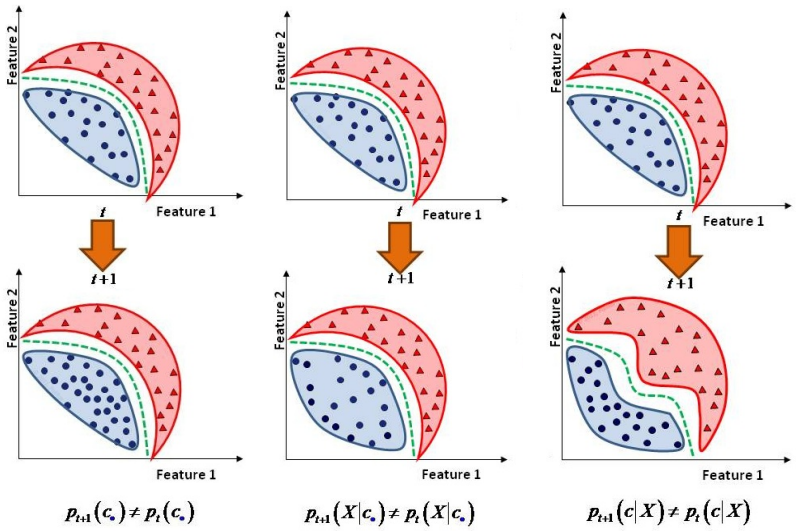
As noted by Kelly, Hand, and Adams [44], concept drift can then occur with respect to any of the three major variables in Bayes' theorem:

1. $p(c_i)$ may change (i.e., class priors).
2. $p(\mathbf{X}_{t+1}|c)$ may change (i.e., the distributions of the classes).
3. $p(c|\mathbf{X}_{t+1})$ may change (i.e., the posterior distributions of class membership).

While Kelly, Hand, and Adams claim that it is only the change in posterior probability that is important, we do not distinguish between the various forms of concept drift, which are depicted in the cartoon representation shown in Figure 2.

In Figure 2(a), the prior probability of the circle class instances increases after concept drift. This models the change in $p(c_i)$; the first type of concept drift. Such concept drift can be problematic, as the change in class priors can cause well calibrated classifiers to become miscalibrated. Furthermore, severe changes in the class priors can lead to the imbalance problem, which we study further in Section 2.2.

The second type of concept drift is demonstrated in Figure 2(b), i.e., a change in $p(\mathbf{X}_{t+1}|c)$.



(a) Drift type 1. Note that the circle class occurs more frequently after drift.

(b) Drift type 2. Note that the boundary of the circle class instances changes drastically after drift.

(c) Drift type 3. Note that the dashed class boundary between the circle and triangular instances changes drastically after drift.

Fig. 2 Graphical representation of the 3 different drift types detailed in Section 2.1.

The drift results in the boundary for the circle class instances to become altered.

Finally, in Figure 2(c) the posterior probability of an instance belonging to a particular class changes after concept drift, as modeled by the shifting dashed class boundary. The added uncertainty, due to a change in $p(c|\mathbf{X}_{t+1})$, is the most severe form of concept drift, because it directly affects the performance of a classifier, as the distribution of the features, with respect to the class, has changed.

The different forms of concept drift can be further classified into two different types, either *real concept drift*, or *virtual concept drift* [78,72]. In virtual concept drift, while the distribution of instances may change (corresponding to a change in the class priors or the distribution of the classes), the underlying concept (i.e., the posterior distribution) does not. This may cause problems for the learner, as such changes in the probability distribution may change the error of the learned model, even if the concept didn't change. Additionally, while previously portions of the target concept may have gone unseen by the learner, due to a change in the distribution such instances may become more prevalent. Since the learner was never presented with such data, it could not learn the concept and therefore must be retrained. This type of virtual drift is especially relevant when the data stream exhibits class imbalance, which we discuss in the next section.

Alternatively, real concept drift is defined as a change in the class boundary (or, more formally, a change in the posterior distribution). While such a change in the posterior distribution indicates a more fundamental change in the generating function, we do distinguish between the two forms of drift in practice. This is due to the fact that while real concept drift does require a change in the model, virtual concept drift does as well. Since the result is the same, regardless of what type of concept drift is detected, no distinction is made between the two forms in this paper.

Finally, while we have given a brief overview of concept drift in particular, Moreno-Torres et. al. [59] provide a more focused overview of different forms of "dataset drift", as well as their causes.

In addition to classifying the different types of concept drift as being real or virtual, it is often common to classify drift based on its speed. In the next section, we discuss the various speeds with which concept drift can occur, and discuss its relative effects.

2.1.1 Speed of Drift

When detecting concept drift, one must be conscientious of the various rates at which concept drift which may be present. In particular, the speed can occur in two main ways: sudden and gradual. For this section we assume that f generates the original concept, and g generates the new concept.

In sudden concept drift (depicted in Figure 3), there is a definite time period t at which f ceases to be used to generate the concepts; at this point g is used instead. This is the simplest case of concept drift. Since sudden concept drift — also referred to as concept change — is defined as having a sharp boundary between generating functions, it is often the easiest to detect, as future data no longer resembles the past data.

In contrast to sudden drift, gradual concept drift (depicted in Figure 3(b)) occurs when there is a (relatively) smooth transition from sampling from f to sampling from g. The smoother the transition from f to g, the more gradual the concept drift. The difficulty of detecting this type of concept drift is further exacerbated by the fact that f and g can be different in minor (but important) ways. In such cases, the existence of a very gradual shift may go unnoticed, increasing the likelihood of it being missed by the classifier.

As concepts change over time, there may be instances where a concept will *reoccur* (shown in Figure 3(c)). A concept can reoccur either suddenly, or gradually. A concept also need not reoccur at regular intervals or in the same manner, instead reoccurring at (seemingly) random times in (seemingly) random ways. Such reoccurring concepts can be exploited by learning algorithm to improve the performance with limited data, as the classifier can retain knowledge of the previous concept, e.g., by keeping models trained on the old data.

2.2 Class Imbalance

Class imbalance is a common problem faced in the data mining community with a rich history [16, 18, 17]. Class imbalance arises when one of the classes (typically the more important, or positive, class) is severely underrepresented in the dataset. Unlike the concept drift problem, the class imbalance problem can also appear in static datasets. In addition to instilling bias in the learning algorithm towards the majority class, class imbalance also causes challenges with interpretation, as the standard figure of merit (i.e., the percentage of correct classification) becomes meaningless. After all, under such a metric, indiscriminately choosing the majority class becomes the optimal decision (e.g., with a class ratio of 99:1, 99% accuracy is achievable by always predicting the majority class).

The problem of class imbalance is further exacerbated when learning from data streams, as the duration between consecutive positive class examples can become arbitrarily large, which in turn may seriously impair the learner's ability to learn the positive class. Consider, for example, the case where a sensor is polled once a day in a dataset which has a class ratio of 100:1. In such instances, it is likely that there will be no positive class instances seen for months at a time. The paucity of data makes the positive class boundary very hard to learn in practice.

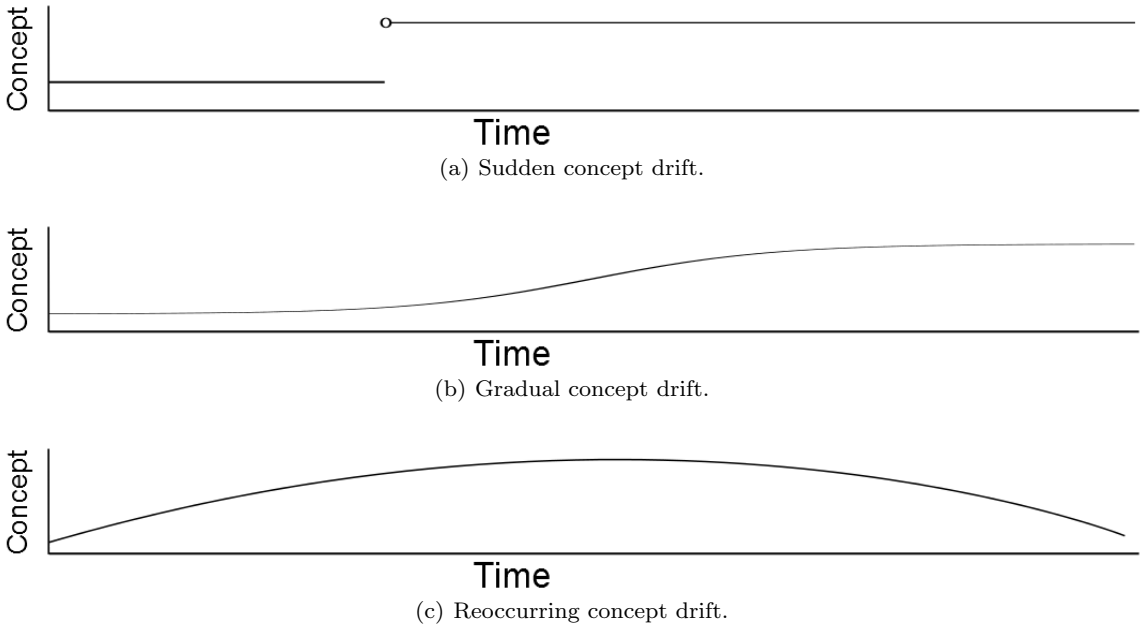
2.3 Concept Drifting Data Streams with Class Imbalance

Combining class imbalance with concept drift, we see that the two problems together provide confounding effects. Namely, in an imbalanced data stream undergoing concept drift, the time until the concept drift is detected can be arbitrarily long. This is due to so few positive examples appearing in the stream, which in turn makes it difficult to infer the source of the error for the positive class. In some cases, the misclassified positive class instance can merely be a result of noise in the data stream. In other cases, however, such a misclassification can signify a drift in concept that must be handled by the algorithm.

In addition to being a challenging problem, there is also a distinct lack of research in such scenarios. Therefore, we recommend further research into the field, as it provides a challenging frontier combining two of the difficult challenges present in data mining.

3 Overcoming Concept Drift

In order to learn in the presence of concept drift, algorithm designers must deal with two main problems. The first problem is detecting concept drift present in the stream. This is also referred to as *c*hange detection or *a*nomaly detection in related literature. Once concept drift has been detected, one must then determine how to best proceed to



(a) Sudden concept drift.

(b) Gradual concept drift.

(c) Reoccurring concept drift.

Fig. 3 Graphical representation of the various speeds of concept drift

make the most appropriate predictions on the new data.

Techniques developed to overcome concept drift can be broken down into three main categories:

- adaptive base learners
- learners which modify the training set
- ensemble techniques

In the following sections we discuss each of the categories, and classifiers in the categories, in more detail.

4 Adaptive Base Learners

Adaptive base learners are the conceptually simplest way of addressing concept drift. Such learners are able to dynamically *adapt* to new training data that contradicts a learned concept. Depending on the base learner employed, this adaptation can take on many forms, but usually relies on restricting (or expanding) the data that the classifier uses to predict new instances in some region of the feature space.

One base learner that has been heavily studied in the context of concept drift is the decision tree; of which the most common variant is C4.5 [68]. The original extension to decision tree learning algorithm, called Very Fast Decision Tree (VFDT) proposed by Domingos and Hulten [24] dealt with building decision trees from streaming data. In VFDT, Hoeffding bounds [38,56] are used to grow decision trees in streaming data. The authors show that in the case of streaming data, applying Hoeffding bounds to a subset of the data can, with high confidence, choose the same split attribute as using all of the data. This observation allows for trees to be grown from streaming data that are nearly equivalent to those built on all of the data.

Many modifications of this work were made for streams which undergo concept drift. The first of which is due to Hulten, Spencer, and Domingos [40], who adapted the learning method to create the Concept-Adapting Very Fast Decision Tree (CVFDT). In CVFDT, a sliding window of instances is retained in short term memory. After a fixed number of new instances arrive, the relevant statistics at each of the nodes are updated (old instances removed, new instances added), and the Hoeffding bounds are recomputed. If a better splitting attribute is found, the (sub)tree determines concept drift may have occurred and a new (sub)tree is learned. The algorithm then waits for more instances and if the new instances confirm that the new (sub)tree learned is of higher quality than the original, the original is replaced.

Since the development of CVFDT, a number of modifications have been proposed [39,7]. Hoeglinger and Pears proposed an alternative to CVFDT which is based on a concept-based window, as opposed to the fixed window in CVFDT. In their approach, instead of updating with respect

to time, the window is updated with respect to concepts. To accomplish this, trees are only grown from the leaves. Instances are then retained until the window becomes full, when underused leaves become "recombined" with their parent nodes, and all instances associated with the leaf are discarded. Leaves are chosen to be recombined by minimizing the overall loss of information for the tree. With the reclaimed window space, new instances can be accepted and the tree can continue to grow.

More recently, Bifet and Gavaldà [7] proposed two new methods: the Hoeffding Window Tree (HWT), and the Hoeffding Adaptive Tree (HAT). HWT is similar to CVFDT with two major differences. First, HWT creates alternative (sub)trees immediately without waiting for a fixed number of instances, which enables the algorithm to more quickly respond to concept drift. Second, HWT does not wait for a fixed number of instances to update a (sub)tree, instead preferring to update as soon as there is evidence to support the improvements of the new (sub)tree. Not having to wait a fixed number of instances gives HWT a distinct advantage over CVFDT, as concept drift can be more automatically detected, with fewer user-defined parameters. Bifet and Gavaldà also propose HAT, which aims to fix the other major deficiency of CVFDT, namely the fixed-size sliding window. Thus, instead of using a fixed window to detect change, HAT use an adaptive window at each internal tree node. The adaptive window allows HAT to more quickly and accurately respond to concept drift, as it is no longer bound to a parameterized window size. Another improvement that HAT brings over CVFDT is that of a concrete performance guarantee. Namely, HAT, under appropriate assumptions and after concept drift has been detected, guarantee to converge to the tree that VFDT would have built seeing only the instances in the new concept.

As an extension to standard decision trees, Buntine [12] introduced option trees, which were further explored by Kohavi and Kunz [46]. In standard decision trees, there is only one possible path from the root to a leaf node, where predictions are made. In option trees, however, a new type of node — known as an option node — is added to the tree, which splits the path along multiple split nodes. Pfahringer, Holmes, and Kirby combined the concept of Hoeffding trees and option trees to create Hoeffding Option Trees (HOTs) [64]. Pfahringer et. al. combine these two methods by altering a standard Hoeffding tree so that, as data arrives, if a new split is found to be better than the current split at a point in the tree, an option node is added and both splits are kept. Bifet et. al. extend HOTs further in Adaptive Hoeffding Option Trees (AHOTs) [8]. In AHOTs, each leaf is provided with an exponential weighted moving average estimator, where the decay is fixed at 0.2. The weight of each leaf is then proportional to the square of the inverse of the error.

An alternative to CVFDT is the CD3 algorithm proposed by Black and Hickey [9], where the authors propose learning decision trees with an additional time-stamp feature for each instance. Once learned, the algorithm can detect concept drift by following each of the paths to the leaves, and determining if the time-stamp was an important attribute in the newly built tree. If the time-stamp was important, and denoted the most recent time period, the rule is said to be good and the data is kept. If, however, the time-stamp is found to be from a previous time period, the rule is said to be invalid and the old instances are removed. Finally, if the time-stamp feature was not used, the rule is kept as valid.

Another heavily studied learning algorithm that has been adapted for concept drift is the k-Nearest Neighbors (kNN) algorithm. Alippi and Roveri [2,3] demonstrate how to modify the kNN algorithm for use in the streaming case. First, they demonstrate how to appropriately choose k in a data stream which does not exhibit concept drift based on theoretical results from Fukunga [31]. With this framework, they describe how to update the kNN classifier when no concept drift is detected (add new instances to the knowledge base), and when concept drift is detected (remove obsolete examples from the knowledge base).

Finally, another popular technique for learning under concept drift is fuzzy ARTMAP by Carpenter et. al. [13], an extension of ARTMAP (Adaptive Resonance Theory Map) due to Carpenter, Grossberg, and Reynolds [14]. ARTMAP attempts to generate a new "cluster" for each pattern that it finds in the dataset, and then maps the cluster to a class. If a new pattern is found that is sufficiently different (defined via a vigilance parameter), then the new pattern is added with its corresponding class. Fuzzy ARTMAP extends this by adding fuzzy logic to ARTMAP. This is accomplished by incorporating two fuzzy ART modules, namely fuzzy ART_a and fuzzy ART_b, connected via an inter-ART module. The inter-ART module F^{ab}, called a *map field*, associates categories

in ART_a to categories in ART_b. If F^{ab} detects a mismatch in categories, the vigilance parameter is increased by the minimum amount needed such that the system searches for, or if necessary creates, a new category such that the predictions once again match. Note that ARTMAP (and by extension fuzzy ARTMAP) are incremental learners who trivially adapt to concept drift through their ability to dynamically create new concepts on the fly.

One extension to fuzzy ARTMAP is due to Andrés-Andrés, Gómez-Sánchez, and Bote-Lorenzo [4], who propose an incremental rule pruning strategy for fuzzy ARTMAP. They accomplish this by extending the work of Carpenter and Tan [15], who propose a pruning strategy based on the confidence, usage, and accuracy of a given rule. The drawback of the proposal from Carpenter and Tan, however, is that it requires remembering all instances in order to update the relevant statistics. Andrés-Andrés, Gómez-Sánchez, and Bote-Lorenzo modify this strategy to enable instances to be forgotten once they have been used to update the model. Instances can be forgotten by slightly modifying the confidence, usage, and accuracy equations such that instances only contribute to these factors when they are learned, and therefore the equations are not modified with old instances when rules are added or removed.

5 Modifying the Training Set

Another popular approach of addressing concept drift is by modifying the training set seen by the classification algorithm. The most common approaches employed are windowing (i.e., where only a subset of previously seen instances are used), and instance weighting. One of the strengths of the modification approaches over the adaptive base learners approach is that the modification strategies are often classifier agnostic. Indeed, much of the research into modifying the training set deals not with building an entire classification algorithm, but merely with how to select or weight instances which are used to build a classifier. We now discuss training set modification strategies, making note of those which are full learning algorithms, and which are merely detection strategies.

5.1 Windowing Techniques

When modifying the training set by way of windowing, the naïve algorithm is to merely keep a fixed number of the newest instances (i.e., a "window" over the newest instances in the data stream) [57]. This naïve approach suffers from many drawbacks, the most important of which is that it is impossible to, a priori, determine the appropriate window size for any given problem.

In order to overcome these shortcomings, many alternative approaches have been presented. One of the original windowing methods is due to Kubat and Widmer [78, 79] in FLORA3. FLORA3 is an extension of FLORA [50], and FLORA2 by [77]. FLORA originated as learning algorithms aimed at learning from streaming data. FLORA is built from sets of disjunctive normal form (DNF) expressions representing the positive examples in the window (ADES), the negative examples in the window (NDES) and potential DNF which covers both positive and negative examples (PDES). FLORA keeps the ADES and NDES sets *maximally general*, while the PDES set is kept *maximally specific*. FLORA3 introduces an adaptive window which attempts to vary its size to fit the current concept. That is, based on the coverage of the ADES set and the performance of the learning algorithm, the window is either grown or shrunk. Specifically, for low coverages and/or poor predictive performance, the window is aggressively shrunk (by 20%). For extremely high coverage, the window is conservatively shrunk (by size 1). If the coverage is high, but the predictive accuracy is good, the window size remains the same. Otherwise the window is grown by 1.

Subsequently, a plethora of windowing algorithms have been proposed. Klinkenberg and Joachims propose a method based on Support Vector Machines (SVMs) [45]. They proposed the use of a $\xi\alpha$-estimator (discussed more formally in [41]) to compute a bound on the error rate of the SVM. Specifically, assuming t batches, they use the $\xi\alpha$-estimator to compute t error bounds. The first error bound corresponds to learning the classifier on just the newest batch, the second bound corresponds to learning on the newest 2 batches, etc. They then choose the window size corresponding to the minimum estimated error.

Gama et. al. [32] proposed a method based on the learner's error rate over time. Assume that each new instance represents a random Bernoulli trial. They then compute the probability of observing a "false" for instance i (p_i), and the standard deviation ($s_i = \sqrt{p_i(1-p_i)/i}$), arguing that a significant increase in the error rate denotes a concept drift. Their algorithm issues a *warning* if

$p_i + s_i \geq p_{min} + 2s_{min}$, and drift is detected if $p_i + s_i \geq p_{min} + 3s_{min}$.

In contrast, Bifet and Gevaldà [6] proposed two methods that determine the window size by ensuring that all sub-windows of the current window represent the same distribution. Their first method, named Adaptive Window (ADWIN), compares the means of two "large enough" sub-windows, and if they are "distinct enough", concept drift is said to have occurred. The definition of "large enough" and "distinct enough" are given by the statistical test chosen. The main drawback is its computational and spacial inefficiency, specifically it: 1) requires keeping a large number of instances (to find two large enough sub-windows); and 2) requires testing all pairs of "large enough" sub-windows. To combat these issues, the authors propose ADWIN2. ADWIN2 introduces a memory-efficient data structure that is able to store a sliding window of length w in logarithmic memory and process the window in logarithmic update time. Using this data structure, the algorithm is able to efficiently update and store a large number of windows. Then, instead of checking all possible sub-windows, evaluates a subset of the possible windows to check for concept drift.

Lazarescu, Venkatesh, and Bui propose using multiple windows to handle concept drift [54]. Instead of a single window, they propose using two windows (named small and medium), where the small window has fixed size S, and the medium window has a minimum size of $2S$. The algorithm then constantly updates the small window with new instances as they arrive. The medium window, on the other hand, is updated depending on the detection of concept drift. Once concept drift is detected over a sufficient number of samples, the medium window is set to size $2S$ in order to capture the new concept. Once the concept has stabilized, the medium window can then grow to a maximum size M.

A final strategy, based on FLORA2, is due to Last [53]. In this strategy, the performance of the learning algorithm is evaluated on the training and a validation set. If a statistically significant difference is detected in the performance of the algorithm over the two datasets, concept drift is considered to be detected, and the window size is updated based on an update rule.

5.2 Weighting Techniques

In addition to windowing techniques, weighting techniques are also commonly used where the relative importance of each instance is used during classification. Instance weighting provides a benefit over windowing techniques, as it allows the learning algorithm to have more precise control over how instances are incorporated into the model than simply "present" or "not present".

Alippi and Roveri's work extending their kNN based method to work under slow, gradual drift in adaptive weighted kNN [1] is a notable example of weighting approaches. In their previous work [2, 3], they recommended the removal of all instances that belong to the "old" knowledge base, and keeping all of the instances in the "new" knowledge base. Implicitly, this means they recommend giving a weight of 0 to instances from the old concept and a weight of 1 to instances in the new concept. In adaptive weighted kNN, the authors recommend to weight all instances based on how likely they are to be from the current concept. Slow concept drift can then be tracked initially as the old examples are still present in the training data with a high weight. As the concept continues to drift, however, these examples are given smaller weights, thereby contributing less to the new model.

6 Ensemble Techniques

The use of ensemble methods is popular in the data mining community due in part to their empirical effectiveness. This effectiveness is derived from combining multiple (usually weak) classifiers trained on similar datasets to provide accurate and robust predictions for future instances. The use of slightly different datasets and/or base learners is important to ensemble methods to avoid overfitting, ensure that the ensemble is diverse, and the learners in the ensemble are not all similarly biased when making predictions. Some examples of traditional ensemble methods are bagging [10], AdaBoost [29], random forest [11], and random subspaces [37]. Overviews of ensemble based decision making can be found in the literature [21, 65, 66].

Ensemble based approaches have been the most popular structure for concept drift algorithms. One important advantage of ensemble techniques in streaming data is their ability to deal with reoccurring concepts. Since ensembles (often) contain models built from past data, such models can be reused to classify new instances if they are drawn

from a reoccurring concept. This is an important advantage over the previous techniques, as other approaches often discard historical data in order to learn the new concepts.

One of the earliest ensemble based approaches for concept drift is Streaming Ensemble Algorithm (SEA) due to Street and Kim [71]. In SEA, the stream is broken into a series of consecutive, non-overlapping windows. For each new window, a new model is learned on all of the instances. If the current ensemble is not full (i.e., there are not more than some predetermined number of classifiers in the ensemble), the new model is added to the ensemble. Otherwise, the model is tested against all other models currently in the ensemble, and the "worst" one is pruned. In order to determine which classifier to prune, Street and Kim recommend a classifier replacement strategy based on instances that were "nearly undecidable". Specifically, if all (or most) classifiers agree on the label of a test instance, that instance did not significantly effect the importance of the classifier in the ensemble. If the votes of the ensemble members are split relatively evenly among class labels, then such an instance would have a much higher impact on the retention (or removal) of this classifier. This approach rewards classifiers that perform well on the "hard" instances (i.e., those correctly classified by only half of the classifiers), while simultaneously ignoring the classifier's performance on "impossible" instances (i.e., those misclassified by all or most of the ensemble members), thereby making the ensemble more robust to noise.

While choosing a pruning strategy is one important consideration in ensemble learning, Wang et. al. [75] demonstrate the effectiveness of also weighting each classifier in the ensemble when performing voting. They prove that an ensemble in which each classifier is weighted inversely proportional to its expected error will always perform at least as well as a single classifier learned on the same data. Given this, and the inability to exactly compute a classifier's expected error, they propose a weight estimation procedure based on the classifier's performance on the previous batch. Two other approaches to weighting are due to Kolter and Maloof [47,48,49] and Becker and Arias [5]. In their weighting schemes, classifiers have their weights updated based on a constant multiplicative factor.

A particularly important ensemble method, called Dynamic Weighted Majority (DWM) proposed by Kolter and Maloof [47,48,49], is the current state of the art method in the literature. In DWM a weighted ensemble of classifiers (whose weights are initially set to 1) is built such that both the overall ensemble performance, combined with the performance of each of the individual classifiers, are combined to overcome concept drift. Specifically, if the ensemble (collectively) misclassifies an instance, a new base learner is added to the ensemble with weight 1. Additionally, if a member of the ensemble misclassifies an instance, then its weight is reduced. If, over time, a base learner's weight drops below a threshold, the classifier is removed from the ensemble entirely. Since DWM only has a method for reducing weights, after updating each classifier's weights, the classifier weights are normalized such that the maximum weight among all classifiers is 1.

Another popular ensemble based technique is due to Polikar et. al. [67] called Learn^{++}. Learn^{++} was developed as an incremental learning algorithm for learning neural network classifiers in streaming data and is loosely based on AdaBoost [29]. The underlying principle of Learn^{++} is that weak learners are generated on the current batch of instances, and then voted together using a weighted average according to the current normalized error of the classifier.

In addition to Learn^{++}, other methods based on boosting are due to Chu and Zaniolo [20] and Scholz and Klinkenberg [69]. Chu and Zaniolo propose an Adaptive Boosting Ensemble (ABE) which performs boosting given only a single pass through the data. It then uses very simple base models (depth-limited decision trees) to exploit the performance characteristics of boosting. In order to handle concept drift, ABE employs a concept drift detection algorithm that notifies the ensemble when concept drift has occurred. When detected, ABE discards the current ensemble, and relearns from scratch.

Elwell, Muhlbaier, and Polikar [27,60, 61] propose Learn^{++}.NSE as an extension of Learn^{++} which is applicable to drifting environments (further experimental verification of the effectiveness of Learn^{++}.NSE are due to Karnick, Muhlbaier, and Polikar [42,43]). The novelty of Learn^{++}.NSE is in determining its voting weights, based on each classifier's time-adjusted accuracy on current and past environments, which gives a higher voting weight to those classifiers — new or old — that perform well in the current environment. Time adjustment comes from weighting performances with respect to time

using a sigmoidal weighting function. Thus, any classifier containing relevant knowledge about the current environment, regardless of its age, can receive a high voting weight. Classifier age itself has no direct effect on voting weight, but rather it is the classifier's performance on recent environments that determine its "time adjusted" voting weight. Such a weighting strategy allows ensemble members to contribute to the ensemble decision if a former concept becomes relevant after long periods (e.e., reoccurring concepts).

Note that the Learn^{++}.NSE does not ever explicitly discard any classifiers in order to ensure that the algorithm can recall reoccurring concepts. In order to investigate the impact of retaining all ensemble members, Elwell and Polikar investigate the effects of pruning the ensemble under various forms of concept drift, various pruning strategies [25], varying rates of concept drift, and the ability of the algorithm to deal with new classes/removal of classes [26]. Based on their experimentation, they note that error based pruning is always preferable to age-based pruning, even in the presence of sudden concept drift. They also note, however, that neither pruning strategy effectively dealt with situations of reoccurring concepts. This is obvious, as one cannot, a priori, determine which concepts are going to reoccur, and thus which classifiers to retain. Additionally, the authors also determine that Learn^{++}.NSE is robust even in such cases where classes are added/removed from the data stream. In light of this, the authors recommend retaining as many models as possible if reoccurring concepts are likely.

In contrast to these boosting techniques, Bifet et. al. propose a technique based on bagging [8] called ADWIN bagging, that employs adaptive classifiers in the ensemble (similar to Committee of Decision Trees (CDC) [70]). Unlike CDC, however, each tree in the ensemble can only grow to a predefined maximum height to ensure the diversity of the ensemble is maintained.

While the previous strategies focused on global concept drift, Tsymbal et. al. propose a strategy based on local concept drift [73,74]. They argue that many real-world scenarios of concept drift are in fact local phenomenon, relegated to a specific region of the feature space. As such, they recommend a dynamic integration of the classifiers in the ensemble based on the local accuracy of each classifier. The authors demonstrate the effectiveness of weighting the classifiers in the ensemble based on the accuracy in the neighborhood (as defined by a relevant distance metric) of the given test instance. Of potential concern in this approach, however, is that the necessity of determining the weights for each classifier may be cost-prohibitive if prediction is time sensitive.

Wang et. al. have proposed a similar strategy, where they assume the feature space has been partitioned [76]. Using these partitions and a forgetting parameter, they are then able to classify a new instance by assigning a weight to each instance in its partition (neighborhood). By controlling how fast old instances are forgotten, this strategy enables the algorithm to trivially handle concept drift, as old concepts are forgotten along with the old instances.

Nishda, Yamauchi, and Omori [63] propose an alternate hybrid approach, called Adaptive Classifier Ensemble (ACE) that mimics the short and long term memory capabilities of the brain. In ACE, the authors combine batch learners (for long term memory) with an online learner (for short term memory), and a drift detection mechanism. By combining the drift detection method with the online learner, ACE is able to rapidly react to suddenly drifting concepts. Since ACE also employs an ensemble, however, it is also robust to stationary and slowly drifting concepts, making it a more robust technique than those previously mentioned.

While the previously mentioned approaches have all assumed that using all past data is advantageous, Fan [28] argues that indiscriminately using old data when building models is only helpful if the concept is constant. When experiencing concept drift, however, instance selection becomes an important problem which must be carefully addressed. Therefore, Fan suggests an approach whereby one uses cross-validation to build a multitude of models, and then "let the data speak for themselves" to choose the models that result in the best performance.

An important drawback of ensemble methods is that each base model is typically learned on a fixed window of the previous data. This may cause very poor models to be built if the drift in the concepts is not well matched with the window size. Therefore various ensembles have been developed that build their base models based on varying window sizes rather than fixed ones. For example, Stanley proposed a method of building an ensemble based on incremental learners called Concept Drift Committee (CDC) [70]. In CDC, a series of n incremental classifiers is learned, such that, initially, classifier i sees all instances j where $j \geq i$. A classifier is

said to be "mature" if it has seen at least as many instances as there are classifiers in the ensemble; an "immature" classifier is not used for classification of new instances. Once a classifier has matured, it begins classifying new instances. If its performance falls below some threshold t, the classifier is removed from the ensemble and a new classifier is learned starting with the current instance.

7 Overcoming Class Imbalance in Concept Drifting Data Streams

In the previous sections we focused on strategies for overcoming concept drift in balanced class distributions. While this research is valuable, a large number of concept drifting data sources also suffer from class imbalance (e.g., credit card fraud, network intrusion detection, etc.). In this section we outline various methods which seek to overcome both issues simultaneously, and note the relative paucity of research into such methods.

In addition to being the most commonly applied technique when dealing only with concept drift, ensemble methods have also been the de facto standard for combating class imbalance. Gao et. al. [34,33] proposed a framework based on collecting positive class examples. In their ensemble algorithm, they break each incoming chunk into a set of positive (P) and negative (Q) class instances. One then selects all seen positive class instances (AP), and a subset of the negative class instances (O) which is determined randomly based on a distribution ratio. These two sets are then combined to form a complete dataset to train the new ensemble classifier C_i. By accumulating all positive class instances, this approach implicitly assumes, however, that the minority class is not drifting.

Building on this concept, Chen and He propose SERA [19], which is similar to the proposal of Gao et. al., however instead of using all past instances, the algorithm selects the "best" n minority class instances as defined by the Mahalanobis distance. Given these instances, the algorithm then uses all majority class instances and uses bagging to build an ensemble of classifiers. Thus SERA suffers from a similar, albeit less severe, concern as the method proposed by Gao et. al., as the algorithm may not be able to track drift in minority instances depending on the parameter n.

Similarly, Lichtenwalter and Chawla [55] propose an extension of Gao et. al.'s work where instead of propagating all minority class examples, they also propagate misclassified majority class instances. In this way, they seek to better define the boundary between the classes, thereby increasing the performance of the ensemble members. Additionally, they propose to use a combination of Hellinger distance and information gain to measure the similarity of the current batch to the batch that each ensemble member was built on. The more similar the batches, the more likely that they describe the same concept. Thus each ensemble member's probability estimate is weighted by the similarity measure in order to obtain a more accurate prediction.

Finally, Ditzler and Polikar outline a method for extending their Learn^{++}.NSE algorithm for the case of class imbalance [22,23]. In these papers, the authors propose Learn^{++}.NIE (for learning in non-stationary and imbalanced environments). In Learn^{++}.NIE, the authors apply the logic of the Learn^{++}.NSE algorithm, with an additional step of using bagging instead of a single base classifier. In this way, the authors claim they can both reduce error via bagging, and, more importantly, learn on a less imbalanced dataset by under-sampling the majority class when creating each bag.

8 Concluding Remarks and Future Work

In this paper, we discuss the leading areas of research in mining data streams that exhibit concept drift and class imbalance. We structured our discussion in terms of the three main areas of research in concept drift: adaptive base learners, modifying the training set, and ensemble techniques.

Despite the growing number of efforts, there is still much work to be done in data streams that exhibit both concept drift and class imbalance. We note that two of the main methods of overcoming of concept drift — adaptive base learners and modifying the training set — have not yet been applied to the class imbalance problem. Additionally, while there has been preliminary work on the use of ensembles to solve the problem of concept drift and class imbalance, the work is sparse, and has not been thoroughly evaluated o n large scale, real-world problems. Hence we propose future work be directed towards overcoming many of the shortcomings of the current body of research by also rigorously evaluating these approaches on large scale, real-world applications.

Acknowledgements

Work is supported in part by the NSF Grant ECCS-0926170, NSF Grant ECCS-092159, and the Notebaert Premier Fellowship.

References

1. Alippi, C., Boracchi, G., Roveri, M.: Just in time classifiers: Managing the slow drift case. In: IJCNN, pp. 114 –120. IEEE (2009). DOI 10.1109/IJCNN.2009.5178799
2. Alippi, C., Roveri, M.: Just-in-time adaptive classifiers in non-stationary conditions. In: IJCNN, pp. 1014–1019. IEEE (2007)
3. Alippi, C., Roveri, M.: Just-in-time adaptive classifierspart ii: Designing the classifier. TNN **19**(12), 2053–2064 (2008)
4. Andres-Andres, A., Gomez-Sanchez, E., Bote-Lorenzo, M.: Incremental rule pruning for fuzzy artmap neural network. ICANN pp. 655–660 (2005)
5. Becker, H., Arias, M.: Real-time ranking with concept drift using expert advice. In: KDD, pp. 86–94. ACM (2007)
6. Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: SDM, pp. 443–448. Citeseer (2007)
7. Bifet, A., Gavalda, R.: Adaptive learning from evolving data streams. IDA pp. 249–260 (2009)
8. Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavaldà, R.: New ensemble methods for evolving data streams. In: KDD, pp. 139–148. ACM (2009)
9. Black, M., Hickey, R.: Learning classification rules for telecom customer call data under concept drift. Soft Computing-A Fusion of Foundations, Methodologies and Applications **8**(2), 102–108 (2003)
10. Breiman, L.: Bagging predictors. Machine Learning **24**(2), 123–140 (1996). DOI 10.1023/A:1018054314350. URL `http://dx.doi.org/10.1023/A:1018054314350`
11. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001). DOI 10.1023/A:1010933404324. URL `http://dx.doi.org/10.1023/A:1010933404324`
12. Buntine, W.: Learning classification trees. Statistics and Computing **2**(2), 63–73 (1992)
13. Carpenter, G., Grossberg, S., Markuzon, N., Reynolds, J., Rosen, D.: Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. TNN **3**(5), 698–713 (1992)
14. Carpenter, G., Grossberg, S., Reynolds, J.: Artmap: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. Neural Networks **4**(5), 565–588 (1991)
15. Carpenter, G., Tan, A.: Rule extraction: From neural architecture to symbolic representation. Connection Science **7**(1), 3–27 (1995)
16. Chawla, N., Japkowicz, N., Kotcz, A.: Editorial: special issue on learning from imbalanced data sets. ACM SIGKDD Explorations Newsletter **6**(1), 1–6 (2004)
17. Chawla, N.V.: Data mining for imbalanced datasets: An overview. In: O. Maimon, L. Rokach (eds.) Data Mining and Knowledge Discovery Handbook, pp. 875–886. Springer (2010)
18. Chawla, N.V., Cieslak, D.A., Hall, L.O., Joshi, A.: Automatically Countering Imbalance and Its Empirical Relationship to Cost. Data Mining and Knowledge Discovery **17**(2), 225 – 252 (2008)
19. Chen, S., He, H.: Sera: Selectively recursive approach towards nonstationary imbalanced stream data mining. In: IJCNN, pp. 522–529. IEEE (2009)
20. Chu, F., Zaniolo, C.: Fast and light boosting for adaptive mining of data streams. Advances in Knowledge Discovery and Data Mining pp. 282–292 (2004)
21. Dietterich, T.: Ensemble methods in machine learning. MCS pp. 1–15 (2000)
22. Ditzler, G., Polikar, R.: An incremental learning framework for concept drift and class imbalance. In: IJCNN. IEEE (2010)
23. Ditzler, G., Polikar, R., Chawla, N.V.: An incremental learning algorithm for nonstationary environments and class imbalance. In: ICPR. IEEE (2010)
24. Domingos, P., Hulten, G.: Mining high-speed data streams. In: KDD, pp. 71–80. ACM (2000)
25. Elwell, R., Polikar, R.: Incremental learning in nonstationary environments with controlled forgetting. In: IJCNN, pp. 771–778. IEEE (2009)
26. Elwell, R., Polikar, R.: Incremental learning of variable rate concept drift. MCS pp. 142–151 (2009)
27. Elwell, R., Polikar, R.: Incremental learning of concept drift in nonstationary environments. TNN **22**(10), 1517–1531 (2011)
28. Fan, W.: Systematic data selection to mine concept-drifting data streams. In: KDD, pp. 128–137. ACM (2004)
29. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: ICML (1996). DOI 10.1007/3-540-59119-2_166
30. Fu, L.: Incremental knowledge acquisition in supervised learning networks. Systems, Man and Cybernetics, Part A: Systems and Humans **26**(6), 801–809 (2002)
31. Fukunaga, K., Hostetler, L.: Optimization of k nearest neighbor density estimates. Information Theory **19**(3), 320–326 (2002)
32. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. AAI pp. 66–112 (2004)
33. Gao, J., Ding, B., Fan, W., Han, J., Yu, P.: Classifying data streams with skewed class distributions and concept drifts. Internet Computing **12**(6), 37–49 (2008)
34. Gao, J., Fan, W., Han, J., Yu, P.: A general framework for mining concept-drifting data streams with skewed distributions. In: SDM, pp. 3–14. Citeseer (2007)
35. Giraud-Carrier, C.: A note on the utility of incremental learning. AI Communications **13**(4), 215–223 (2000)
36. Grossberg, S.: Nonlinear neural networks: Principles, mechanisms, and architectures. Neural Networks **1**(1), 17–61 (1988)
37. Ho, T.: The random subspace method for constructing decision forests. PAMI **20**(8), 832–844 (1998)
38. Hoeffding, W.: Probability inequalities for sums of bounded random variables. JASA **58**(301), 13–30 (1963)
39. Hoeglinger, S., Pears, R.: Use of hoeffding trees in concept based data stream mining. In: ICIAFS, pp. 57 –62 (2007). DOI 10.1109/ICIAFS.2007.4544780

40. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: KDD, pp. 97–106. ACM (2001)
41. Joachims, T.: Estimating the generalization performance of an svm efficiently. In: ICML, p. 431. Morgan Kaufmann (2000)
42. Karnick, M., Ahiskali, M., Muhlbaier, M., Polikar, R.: Learning concept drift in nonstationary environments using an ensemble of classifiers based approach. In: IJCNN, pp. 3455–3462. IEEE (2008)
43. Karnick, M., Muhlbaier, M., Polikar, R.: Incremental learning in non-stationary environments with concept drift using a multiple classifier based approach. In: ICPR, pp. 1–4. IEEE (2009)
44. Kelly, M., Hand, D., Adams, N.: The impact of changing populations on classifier performance. In: KDD, pp. 367–371. ACM (1999)
45. Klinkenberg, R., Joachims, T.: Detecting concept drift with support vector machines. In: ICML. Citeseer (2000)
46. Kohavi, R., Kunz, C.: Option decision trees with majority votes. In: ICML, pp. 161–169. Morgan Kaufmann (1997)
47. Kolter, J., Maloof, M.: Dynamic weighted majority: A new ensemble method for tracking concept drift. In: ICDM, pp. 123–130. IEEE (2003)
48. Kolter, J., Maloof, M.: Using additive expert ensembles to cope with concept drift. In: ICML, pp. 449–456. ACM (2005)
49. Kolter, J., Maloof, M.: Dynamic weighted majority: An ensemble method for drifting concepts. JMLR **8**, 2755–2790 (2007)
50. Kubat, M.: Floating approximation in time-varying knowledge bases. PRL **10**(4), 223–227 (1989)
51. Lange, S., Grieser, G.: On the power of incremental learning. TCS **288**(2), 277–307 (2002)
52. Lange, S., Zilles, S.: Formal models of incremental learning and their analysis. In: IJCNN, vol. 4, pp. 2691–2696. IEEE (2003)
53. Last, M.: Online classification of nonstationary data streams. IDA **6**(2), 129–147 (2002)
54. Lazarescu, M., Venkatesh, S., Bui, H.: Using multiple windows to track concept drift. IDA **8**(1), 29–59 (2004)
55. Lichtenwalter, R., Chawla, N.V.: Adaptive methods for classification in arbitrarily imbalanced and drifting data streams. In: New Frontiers in Applied Data Mining, *Lecture Notes in Computer Science*, vol. 5669, pp. 53–75. Springer Berlin / Heidelberg (2010)
56. Maron, O., Moore, A.W.: Hoeffding races: Accelerating model selection search for classification and function approximation. In: NIPS, pp. 59–66 (1993)
57. Mitchell, T., Caruana, R., Freitag, D., McDermott, J., Zabowski, D.: Experience with a learning personal assistant. Communications of the ACM **37**(7), 80–91 (1994)
58. Moreno-Torres, J., Herrera, F.: A preliminary study on overlapping and data fracture in imbalanced domains by means of genetic programming-based feature extraction. In: Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on, pp. 501 –506 (2010). DOI 10.1109/ISDA.2010.5687214
59. Moreno-Torres, J., Raeder, T., Alaiz-Rodríguez, R., Chawla, N.V., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recognition (2011)
60. Muhlbaier, M., Polikar, R.: An ensemble approach for incremental learning in nonstationary environments. MCS pp. 490–500 (2007)
61. Muhlbaier, M., Polikar, R.: Multiple classifiers based incremental learning algorithm for learning in nonstationary environments. In: ICMLC, vol. 6, pp. 3618–3623. IEEE (2007)
62. Muhlbaier, M., Topalis, A., Polikar, R.: Learn^{++}.nc: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes. TNN **20**(1), 152 –168 (2009). DOI 10.1109/TNN.2008.2008326
63. Nishida, K., Yamauchi, K., Omori, T.: Ace: Adaptive classifiers-ensemble system for concept-drifting environments. MCS pp. 176–185 (2005)
64. Pfahringer, B., Holmes, G., Kirkby, R.: New options for hoeffding trees. AAI pp. 90–99 (2007)
65. Polikar, R.: Ensemble based systems in decision making. Circuits and Systems Magazine **6**(3), 21–45 (2006)
66. Polikar, R.: Bootstrap-inspired techniques in computation intelligence. Signal Processing Magazine **24**(4), 59–72 (2007)
67. Polikar, R., Upda, L., Upda, S.S., Honavar, V.: Learn^{++}: an incremental learning algorithm for supervised neural networks. IEEE Transactions on Systems, Man, and Cybernetics, Part C pp. 497–508 (2001)
68. Quinlan, J.: C4.5: programs for machine learning. Morgan Kaufmann (1993)
69. Scholz, M., Klinkenberg, R.: Boosting classifiers for drifting concepts. IDA **11**(1), 3–28 (2007)
70. Stanley, K.: Learning concept drift with a committee of decision trees. Tech. Rep. AI-03-302, Computer Science Department, University of Texas-Austin (2003)
71. Street, W., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In: KDD, pp. 377–382. ACM (2001)
72. Tsymbal, A.: The problem of concept drift: definitions and related work. Tech. Rep. TCD-CS-2004-15, Departament of Computer Science, Trinity College (2004). URL `https://www.cs.tcd.ie/publications/techreports/reports`
73. Tsymbal, A., Pechenizkiy, M., Cunningham, P., Puuronen, S.: Handling local concept drift with dynamic integration of classifiers: Domain of antibiotic resistance in nosocomial infections. In: Computer-Based Medical Systems, 2006. CBMS 2006. 19th IEEE International Symposium on, pp. 679 –684 (2006). DOI 10.1109/CBMS.2006.94
74. Tsymbal, A., Pechenizkiy, M., Cunningham, P., Puuronen, S.: Dynamic integration of classifiers for handling concept drift. Information Fusion **9**(1), 56–68 (2008)
75. Wang, H., Fan, W., Yu, P., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: KDD, pp. 226–235. ACM (2003)
76. Wang, H., Yin, J., Pei, J., Yu, P., Yu, J.: Suppressing model overfitting in mining concept-drifting data streams. In: KDD, pp. 736–741. ACM (2006)
77. Widmer, G., Kubat, M.: Learning flexible concepts from streams of examples: Flora2. In: ECAI, p. 467. John Wiley & Sons, Inc. (1992)
78. Widmer, G., Kubat, M.: Effective learning in dynamic environments by explicit context tracking. In: ECML, pp. 227–243. Springer (1993)

79. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine learning **23**(1), 69–101 (1996)