2017

Ensemble learning for data stream analysis: A survey

Bartosz Krawczyk
Virginia Commonwealth University, bkrawczyk@vcu.edu

Leandro L. Minku
University of Leicester

João Gama
University of Porto

Jerzy Stefanowski
Poznań University of Technology

Michał Wozniak
Wrocław University of Science and Technology

Information Fusion 37 (2017) 132–156

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Ensemble learning for data stream analysis: A survey

Bartosz Krawczyk[a,*], Leandro L. Minku[b], João Gama[c], Jerzy Stefanowski[d], Michał Woźniak[e]

[a] Department of Computer Science, Virginia Commonwealth University, Richmond, VA 23284, USA
[b] Department of Computer Science, University of Leicester, Leicester, UK
[c] Laboratory of Artificial Intelligence and Decision Support, University of Porto, Porto, Portugal
[d] Institute of Computing Science, Poznań University of Technology, 60-965 Poznań, Poland
[e] Department of Systems and Computer Networks, Wrocław University of Science and Technology, Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland

ARTICLE INFO

Article history:
Received 15 December 2016
Revised 31 January 2017
Accepted 1 February 2017
Available online 3 February 2017

Keywords:
Ensemble learning
Data streams
Concept drift
Online learning
Non-stationary environments

ABSTRACT

In many applications of information systems learning algorithms have to act in dynamic environments where data are collected in the form of transient data streams. Compared to static data mining, processing streams imposes new computational requirements for algorithms to incrementally process incoming examples while using limited memory and time. Furthermore, due to the non-stationary characteristics of streaming data, prediction models are often also required to adapt to concept drifts. Out of several new proposed stream algorithms, ensembles play an important role, in particular for non-stationary environments. This paper surveys research on ensembles for data stream classification as well as regression tasks. Besides presenting a comprehensive spectrum of ensemble approaches for data streams, we also discuss advanced learning concepts such as imbalanced data streams, novelty detection, active and semi-supervised learning, complex data representations and structured outputs. The paper concludes with a discussion of open research problems and lines of future research.

Published by Elsevier B.V.

1. Introduction

The analysis of huge volumes of data is recently the focus of intense research, because such methods could give a competitive advantage for a given company. For contemporary enterprises, the possibility of making appropriate business decisions on the basis of knowledge hidden in stored data is one of the critical success factors. Similar interests in exploring new types of data are present in many other areas of human activity.

In many of these applications, one should also take into consideration that data usually comes continuously in the form of *data streams*. Representative examples include network analysis, financial data prediction, traffic control, sensor measurement processing, ubiquitous computing, GPS and mobile device tracking, user's click log mining, sentiment analysis, and many others [19,59,60,203,208].

Data streams pose new challenges for machine learning and data mining as the traditional methods have been designed for static datasets and are not capable of efficiently analyzing fast growing amounts of data and taking into consideration characteristics such as:

- Limited computational resources as memory and time, as well as tight needs to make predictions in reasonable time.
- The phenomenon called *concept drift*, i.e., changes in distribution of data which occur in the stream over time. This could dramatically deteriorate performance of the used model.
- Data may come so quickly in some applications that labeling all items may be delayed or sometimes even impossible.

Out of several tasks studied in data streams [60], *supervised classification* has received the most research attention. It is often applied to solve many real life problems such as discovering client preference changes, spam filtering, fraud detection, and medical diagnosis to enumerate only a few. The aforementioned speed, size and evolving nature of data streams pose the need for developing new algorithmic solutions. In particular, classifiers dedicated to data streams have to present adaptation abilities, because the distribution of the data in motion can change. To tackle these challenges, several new algorithms, such as VFDT [44], specialized sliding windows, sampling methods, drift detectors and adaptive ensembles have been introduced in the last decade.

In our opinion, *ensemble methods* are one of the most promising research directions [188]. An ensemble, also called a multiple classifier or committee, is a set of individual component classi-

* Corresponding author.
E-mail addresses: bkrawczyk@vcu.edu (B. Krawczyk), leandro.minku@leicester.ac.uk (L.L. Minku), jgama@fep.up.pt (J. Gama), jerzy.stefanowski@cs.put.poznan.pl (J. Stefanowski), michal.wozniak@pwr.edu.pl (M. Woźniak).

http://dx.doi.org/10.1016/j.inffus.2017.02.004
1566-2535/Published by Elsevier B.V.

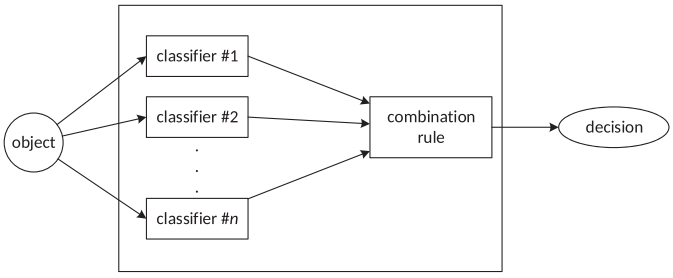

Fig. 1. A diagram of the classifier ensemble.

fiers whose predictions are combined to predict new incoming instances. Ensembles have been shown to be an efficient way of improving predictive accuracy or/and decomposing a complex, difficult learning problem into easier sub-problems.

The main motivation for using classifier ensembles is the *no free lunch* theorem formulated by Wolpert [185]. According to it, there is not a single classifier that is appropriate for all the tasks, since each algorithm has its own domain of competence. Usually, we have a pool of classifiers at our disposal to solve a given problem. Turner [176] showed that averaging outputs of an infinite number of unbiased and independent classifiers may lead to the same response as the optimal Bayes classifier [48]. Ho [75] underlined that a decision combination function must receive useful representation of each individual decision. Specifically, they considered several methods based on decision ranks, such as Borda count.

We also have to mention another of Ho's work [74], who distinguished two main approaches to design a classifier ensemble:

- *Coverage optimization* focuses on the generation of a set of mutually complementary classifiers, which may be combined to achieve optimal accuracy using a fixed decision combination function.
- *Decision optimization* concentrates on designing and training an appropriate decision combination function, while a set of individual models is given in advance [151].

Other important issues that have be taken into consideration when building classifier ensembles are the following:

- Proposing interconnections among individual classifiers in the ensemble.
- Selecting a pool of diverse and complementary individual classifiers for the ensemble.
- Proposing a combination rule, responsible for the final decision of the ensemble, which should exploit the strengths of the component classifiers.

The general diagram of a classifier ensemble is depicted in Fig. 1.

The selection of classifiers for the ensemble is a key factor. An ideal ensemble includes mutually complementary individual classifiers which are characterized by high diversity and accuracy [106]. It is generally agreed that not only the accuracy, but also the diversity of the classifiers is a key ingredient for increasing the ensemble's accuracy [195]. Classifiers must be selected to obtain positive results from their combination. Sharkley et al. [159] proposed four levels of diversity based on the majority vote rule, coincident error, and the possibility of at least one correct answer of ensemble members. Brown et al. [24] reflected that it is inappropriate for the case where diversity of an ensemble is different in various subspaces of the feature space. For comprehensive reviews on ensemble methods developed for static datasets see, e.g., [108].

Classifier ensembles are an attractive approach to construct data stream classifiers, because they facilitate adaptation to changes in the data distribution. Their adaptation could be done by changing the line-up of the ensemble, e.g., by adding components classifiers trained on the most recent data and/or removing the outdated classifiers, or by retraining the ensemble components.

There are several interesting books or surveys on the data stream analysis and classification, but most of them focus on general methods of data stream analysis, not dedicating too much space to ensemble approaches [43,60,64,114,131], and some have been written several years ago [59,107,109]. Therefore, there is still a gap in this literature with respect to present the development in learning ensembles from data streams. This survey aims to fill this gap.

It is also worth mentioning the work [105,207], where data stream mining challenges have been discussed. We will discuss open research problems and lines of future research in the specific area of ensemble approaches for data streams.

We will pay the most attention to classifier ensembles, given that most existing literature is in this area. However, we will also discuss research on regression (or prediction model) ensembles. Furthermore, we will review recent ensemble approaches dedicated to various more complex data representations in streams.

This survey is organized as follows. Section 2 focuses on the main characteristics of data streams and methods dedicated to their analysis, as well as on the type of data streams and drift detection methods. Section 3 presents methods for evaluating classifiers over streaming data. In Section 4, a comprehensive survey on ensemble techniques for classification and regression problems is presented. Section 5 enumerates advanced problems for data stream mining, such as imbalanced data, novelty detection, one-class classification, and active learning, as well as focuses on non-standard and complex data representations or class structures. The final section draws open challenges in this field for future research.

2. Data stream characteristics

In this section we will provide a general overview of the data stream domain, discussing different types of streaming data, learning frameworks used for its analysis, and the issue of changes in the data stream distribution, known as concept drift.

2.1. General issues

A data stream is a potentially unbounded, ordered sequence of data items which arrive over time. The time intervals between the arrival of each data item may vary. These data items can be simple attribute-value pairs like relational database tuples, or more complex structures such as graphs.

The main differences between data streams and conventional static datasets include [11,60,169]:

- data items in the stream appear sequentially over time,
- there is no control over the order in which data items arrive and the processing system should be ready to react at any time,
- the size of the data may be huge (streams are possibly of infinite length); it is usually impossible to store all the data from the data stream in memory,
- usually only one scan of items from a data stream is possible; when the item is processed it is discarded or sometimes stored if necessary, or aggregated statistics or synopses are calculated,
- the data items arrival rate is rapid (relatively high with respect to the processing power of the system),
- data streams are susceptible to change (data distributions generating examples may change on the fly),
- the data labeling may be very costly (or even impossible in some cases), and may not be immediate.

These data stream characteristics pose the need for other algorithms than ones previously developed for *batch learning*, where

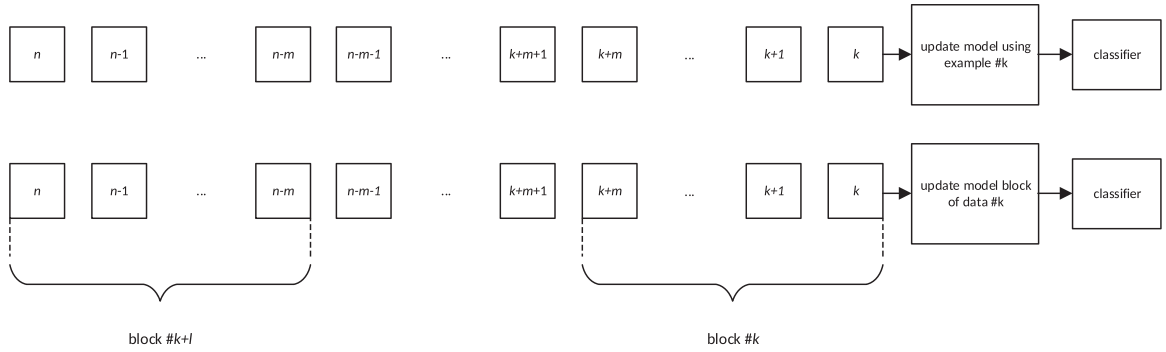

Fig. 2. Difference between incremental and block base classifier updating.

data are stored in finite, persistent data repositories. Typical batch learning algorithms are not capable of fulfilling all of the data stream requirements such as constraints of memory usage, restricted processing time, and one scan of incoming examples [25]. Note that some algorithms, like Naïve Bayes, instance based learning or neural networks are naturally *incremental* ones. However, simple incremental learning is typically insufficient, as it does not meet tight computational demands and does not tackle evolving nature of data sources [60].

Constraints on memory and time have resulted in the development of different kinds of windowing techniques, sampling (e.g. reservoir sampling) and other summarization approaches. However, the distribution in the data source generating the stream data items may change over time. Thus, in case of non-stationary data streams, data from the past can become irrelevant or even harmful for the current situation, deteriorating predictions of the classifiers. Data management approaches can play the role of a forgetting mechanism where old data instances are discarded.

2.2. Types of data streams and learning frameworks

If a completely *supervised learning* framework is considered, it is assumed that after some time the true target output value y^t of the example is available. Thus, data stream S is a sequence of labeled examples $\mathbf{z}^t = (\mathbf{x}^t, y^t)$ for $t = 1, 2, \ldots, T$. Usually, $\mathbf{x}$ is a vector of attribute values, and y is either a discrete class label ($y \in \{K_1, \ldots, K_l\}$) for classification problems or numeric output (independent) values for regression problems. The general task is to learn from the past data (a training set of examples) the relationship between the set of attributes and the target output. In the case of classification, this relationship corresponds to discovered classification knowledge and it is often used as classifier C to determine the class label for the new coming example $\mathbf{x}^{t'}$. In the case of regression, the learned model is used to predict a numeric value. Note that the classifier or the regression model is supposed to provide its prediction at *any time* based on what it has learned from the data items $\{\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^t\}$ seen so far. This prediction $\hat{y}^t$ and true target value y^t can be used by the learning algorithm as additional learning information.

As most of the current research on data stream ensembles concerns classification, we will present the remaining of this section using the classification terminology. However, nearly all of these issues are also valid for regression cases.

The majority of proposed algorithms for learning stream classifiers follow the supervised framework (i.e. with a complete and immediate access to class labels for all processed examples). However, in some applications the assumption of a complete labeling of learning examples may be unrealistic or impractical, as the class labels of newly coming examples in data streams are not immediately available. For instance, in the financial fraud detection, information on fraud transactions is usually known after a long delay (e.g. when an account holder receives the monthly report [52]), while for a credit approval problem the true label is often available after 2–3 years. Moreover, the acquiring of labels from experts is costly and needs substantial efforts [204]. Therefore some researchers consider other frameworks such as:

- learning with delayed labeling when an access to true class labels is available much later than it is expected; the classifier may adapt to the stream earlier without knowing it [104],
- semi-supervised learning where labels are not available for all incoming examples; They are provided in limited portions from time to time; alternatively, the system employs an active learning technique, which selects unlabeled examples for acquiring their labels [52,97,110,204],
- unsupervised framework or learning from initially labeled examples; An initial classifier is learned from a limited number of labeled training examples, and then it processes the upcoming stream of unlabeled examples without any access to their labels [49].

We will come to these issues in Section 5.3.

Examples from the data stream are provided either *online*, i.e., instance by instance, or in the form of *data chunks* (portions, blocks). In the first approach, algorithms process single examples appearing one by one in consecutive moments in time, while in the other approach, examples are available only in larger sets called *data blocks* (or *data chunks*) $S = B_1 \cup B_2 \cup \ldots \cup B_n$. Blocks are usually of equal size and the construction, evaluation, or updating of classifiers is done when all examples from a new block are available. This distinction may be connected with supervised or semi-supervised frameworks. For instance, in some problems data items are more naturally accumulated for some time and labeled in blocks while an access to class labels in an online setup is more demanding. Moreover, these types of processing examples also influence the evaluation of classifiers. Both discussed modes are depicted in Fig. 2.

2.3. Stationary and non-stationary (drifting) data streams

Two basic models of data streams are considered: *stationary*, where examples are drawn from a fixed, albeit unknown, probability distribution, and *non-stationary*, where data can evolve over time. In the second case, target concepts (classes of examples) and/or attribute distributions change. In other words, the concept from which the data stream is generated shifts after a minimum stability period [60]. This phenomenon is called *concept drift*, a.k.a, covariant shift. Concept drifts are reflected in the incoming instances and deteriorate the accuracy of classifiers/regression models learned from past training instances. Typical real life streams affected by concept drift could include [200]:

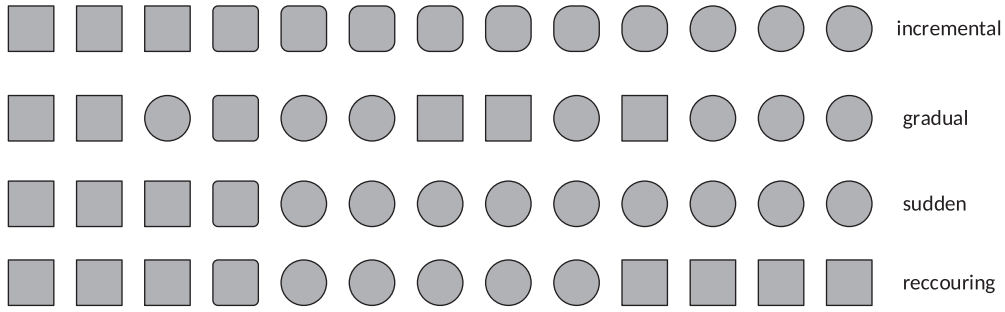

Fig. 3. Type of drifts.

- computer or telecommunication systems, where attackers look for new ways of overcoming security systems,
- traffic monitoring, where traffic patterns may change over time,
- Weather predictions, where climate changes and natural anomalies may influence the forecast,
- system following personal interests, like personal advertisement, where users may change their preferences, and
- medical decision aiding, where disease progression may be influenced and changed in response to applied drugs or natural resistance of the patients.

Other examples of real life concept drifts include spam categorization, object positioning, industrial monitoring systems, financial fraud detection, and robotics; and they are reviewed in the recent survey [208].

Concept drift can be defined from the perspective of hidden data contexts, which are unknown to the learning algorithm. Zliobaite also calls it an unforeseen change as the change is unexpected with respect to the current domain knowledge or previous learning examples [200]. However, a more probabilistic view on this matter is usually presented, e.g. [60,183].

In each point in time t, every example is generated by a source with a joint probability distribution $P^t(\mathbf{x}, y)$. Concepts in data are *stable or stationary* if all examples are generated by the same distribution. If, for two distinct points in time t and $t + \Delta$, there exits $\mathbf{x}$ such that $P^t(\mathbf{x}, y) \neq P^{t+\Delta}(\mathbf{x}, y)$, then concept drift has occurred.

Different components of $P^t(\mathbf{x}, y)$ may change [60]. In particular, when concept drift occurs, either one or both of the following changes:

- prior probabilities of classes $P(y)$,
- class conditional probabilities $P(\mathbf{x}|y)$.

As a result, posterior probabilities of the classes $P(y|\mathbf{x})$ may (or may not) change.

Based on the cause and effect of these changes, two types of drift are distinguished: *real drift* and *virtual drift*.

A real drift is defined as a change in $P(y|\mathbf{x})$. It is worth noting that such changes can occur with or without changes in $P(\mathbf{x})$. Therefore, they may or may not be visible from the data distribution without knowing the true class labels. Such a distinction is crucial, as some methods attempt to detect concept drifts using solely input attribute values. Real drift has also been referred to as concept shift and conditional change [64].

A virtual drift is usually defined as a change in the attribute-value $P(\mathbf{x})$, or class distributions $P(y)$ that does not affect decision boundaries. In some work virtual drift is defined as a change that does not affect the *posterior* probabilities, but it is hard to imagine that $P(\mathbf{x})$ is changed without changing $P(y|\mathbf{x}) = \frac{P(y)P(\mathbf{x}|y)}{P(\mathbf{x})}$ in real world applications. However, the source and therefore the interpretation of such changes differs among authors. Widmer and Kubat [184] attributed virtual drift to incomplete data representation rather than to true changes in concepts. Tsymbal [175] on the other hand defined virtual drift as changes in the data distribution that do not modify the decision boundary, while Delany [40] described it as a drift that does not affect the target concept. Furthermore, virtual drifts have also been called temporary drifts, sampling shifts or feature changes [25].

Most current research on learning classifiers from evolving streams concentrates on real drifts. However, it is worth mentioning that even if the true class boundaries do not change in virtual drifts, this type of drift may still result in the learnt class boundaries to become inadequate. Therefore, techniques for handling real drifts may still work for certain types of virtual drifts. If posterior probabilities do not change, it is worthless to rebuild the model, because the decision boundaries are still the same. Virtual drift detection is also important, because even though it does not effect the decision boundaries of the classifier, its wrong interpretation (i.e., detecting and classifying as real drift) could provide wrong decision about classifier retraining.

Apart from differences in the cause and effect of concept changes, researchers distinguish between several ways of how such changes occur. Concept drifts can be further characterized, for example, by their permanence, severity, predictability, and frequency. The reader is also referred to the recent paper by Hyde et al. [183], which is the first attempt to provide the more formal framework for comparing different types of drifts and their main properties. These authors also proposed a new, quite comprehensive taxonomy of concept drift types.

The most popular categorizations include *sudden* (abrupt) and *gradual* drifts [175]. The first type of drift occurs when, at a moment in time t, the source distribution in S^t is suddenly replaced by a different distribution in S^{t+1}. Gradual drifts are not so radical and are connected with a slower rate of changes, which can be noticed while observing a data stream for a longer period of time. Additionally, some authors distinguish two types of gradual drift [126]. The first type of gradual drift refers to the transition phase where the probability of sampling from the first distribution P^j decreases while the probability of getting examples from the next distribution P^{j+1} increases. The other type, called *incremental* (stepwise) drift, consists of a sequence of small (i.e., not severe) changes. As each change is small, the drift may be noticed only after a long period of time, even if each small change occurs suddenly.

In some domains, situations when previous concepts reappear after some time are separately treated and analyzed as *recurrent* drifts. This re-occurrence of drifts could be cyclic (concepts reoccur in a specific order) or not [175]. Moreover, data streams may contain blips (rare events/outliers) and noise, but these are not considered as concept drifts and data stream classifiers should be robust to them. The differences among the drifts are depicted in Fig. 3.

Some other drift characteristics are also considered in the literature. Typically, real concept drift concerns changes for all examples but it could be also a sub-concept change where drift is lim-

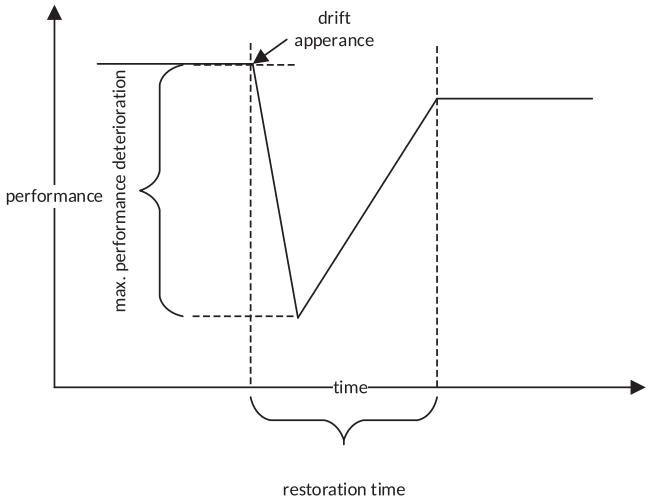

Fig. 4. The idea of the model restoration time.

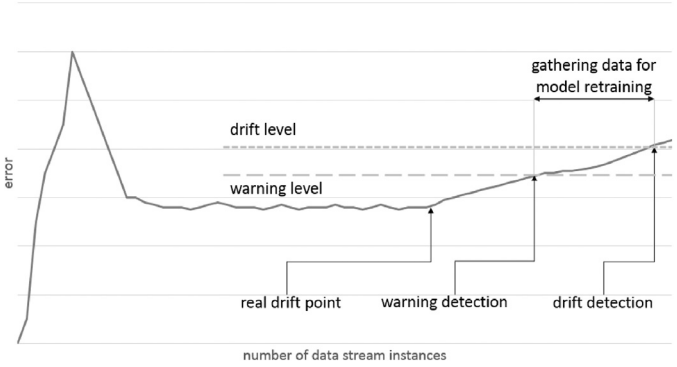

Fig. 5. The idea of drift detection based on tracking classifier errors.

ited to a subspace of a domain – see discussions on the drift severity in [126]. Moreover, in real life situations, concept drifts may be a complex combination of many types of basic drifts.

For more information on these and other changes in underlying data distributions, the reader is referred to [60,64,114,175,183]. These studies, and more application oriented papers, such as [208], demonstrate that the problem of concept drift has also been recognized and addressed in multiple application areas. This shows the strong requirement for streaming classifiers to be capable of predicting, detecting, and adapting to concept drifts.

2.4. Drift detection methods

Concept drift detectors are methods, which on the basis of information about classifier's performance or the incoming data items themselves, can signal that data stream distributions are changing. Such signals usually trigger updating/retraining of the model, or substituting the outdated model by the new one. Our aim is on the one hand to reduce the maximum performance deterioration and on the other hand to minimize so-called restoration time (see Fig. 4).

The detectors may return not only signals about drift detection, but also warning signals, which are usually treated as a moment when a change is suspected and a new training set representing the new concept should start being gathered. The idea of drift detection is presented in Fig. 5.

Drift detection is not a trivial task, because on the one hand we require sufficiently fast drift detection to quickly replace outdated model and to reduce the restoration time. On the other hand we do not want too many false alarms [69]. Therefore, to assess a concept drift detector's performance, the following metrics are usually considered:

- number of true positive drift detections,
- number of false alarms, i.e., false positive drift detections,
- drift detection delay, i.e., time between real drift appearance and its detection.

One difficulty arises because there is typically a trade-off between different metrics. For instance, a drift detector can typically be tuned to decrease the detection delay, but this may lead to a higher number of false alarms. In view of that, Alippi et al. [7] have recently used the following procedure to evaluate their drift detection method when using artificial data streams. They generates a stream that contains enough instances after a drift so that drifts are always detected by all drift detection methods being evaluated. They then plotted the number of false alarms versus the drift detection delay for all drift detectors, using several different parameter configurations. This lead to a curve that resembles the Receiver Operating Characteristics curve, but used to evaluate drift detection methods rather than classifiers.

In a few papers aggregated measures, which take into consideration the aforementioned metrics, are also proposed. It is worth mentioning the work of Pesaranghader and Victor [141], where the *acceptable delay length* was defined to determine how far the detected drift could be from the true location of drift, for being considered as a true positive. A recent experimental framework for the drift detection evaluation can be found in [89].

The authors of [64] propose to categorize the drift detectors into the following four main groups:

1. Detectors based on Statistical Process Control.
2. Detectors based on the sequential analysis.
3. Methods monitoring distributions of two different time windows.
4. Contextual approaches.

In the next paragraphs, we briefly describe a few drift detection methods.

DDM (*Drift Detection Method*) [62] is the most well known representative of the first category. It estimates classifier error (and its standard deviation), which (assuming the convergence of the classifier training method) has to decrease as more training examples are received [147]. If the classifier error is increasing with the number of training examples, then this suggests a concept drift, and the current model should be rebuilt. More technically, DDM generates a warning signal if the estimated error plus twice its deviation reaches a warning level. If the warning level is reached, new incoming examples are remembered in a special window. If afterwards the error falls below the warning threshold, this warning is treated as a false alarm and this special window is dropped. However, it the error increases with time and reaches the drift level, the current classifier is discarded and a new one is learned from the recent labeled examples stored in the window. Note that this detection idea may be also used to estimate time interval between the warning and drift detection, where shorter times indicate a higher rate of changes.

EDDM (*Early Drift Detection Method*) is a modification of DDM to improve the detection of gradual drifts [10]. The same idea of warning and drift levels is realized with a new proposal of comparing distances of error rates. Yet another detector ECDD employs the idea of observing changes in the exponentially weighted moving average [152].

The sequential probability ratio tests, such as the Wald test, are the basis for detectors belonging to the second category. The cumulative sum approach (CUSUM) [138] detects a change of a given parameter value of a probability distribution and indicates

when the change is significant. As the parameter the expected value of the classification error could be considered, which may be estimated on the basis of labels of incoming examples from data stream. A comprehensive analysis of the relationship between CUSUM's parameters and its performance was presented in [64].

PageHinkley is modification of the CUSUM algorithm, where the cumulative difference between observed classifier error and its average is taken into consideration [156].

Yet other drift detectors based on non-parametric estimation of classifier error employing Hoeffding's and McDiarmid's inequalities were proposed in [22].

ADWIN is the best known representative of methods comparing two sliding windows. In this algorithm [14] a window of incoming examples grows until identifying a change in the average value inside the window. When the algorithm succeeds at finding two distinct sub-windows, their split point is considered as an indication of concept drift.

Besides the use of parametric tests for concept drift detection, some non-parametric tests have also been investigated, such as the computational intelligence cumulative sum test [8] and the intersection of confidence intervals-based change detection test [6].

Alippi presents an interesting comparison of different triggering mechanisms for concept drift detection [5]. It is worth noting that drift detectors frequently rely on continuous access to class labels, which usually cannot be granted from the practical point of view. Therefore, during constructing the concept drift detectors we have to take into consideration the cost of data labeling, which is usually passed over. A very interesting way to design detectors is to employ the *active learning* paradigm [68] or unlabeled examples only.

Unsupervised detection of virtual concept drift is most often performed with statistical tests [120], which check whether a current data portion comes from the same distribution as the reference data. Obviously, not all statistical tests are suited for this task, e.g., two-sample parametric tests such as a T2 statistic [79] assume a specific distribution, which might not be a correct approach in the real data case. Also, the distributions may not be similar to any standard distribution, what moreover suggests non-parametric tests for the task of unsupervised concept drift detection. Examples of such tests include [164]:

- CNF Density Estimation test introduced in [45], describes the data by vectors of binary features, assigned by discretizing attributes into sets of bins. Then, it creates a set of Boolean attributes, which covers all of the examples in the reference dataset, meaning that each true feature in attribute set is the same as in at least one of the vectors describing the data points in the reference set. Next, another set of data is drawn from the same distribution as the data in the reference set, represented as binary vectors, and compared to the attribute set by applying a Matt–Whitney test. If the difference is insignificant, all data is considered to come from the same distribution, otherwise a difference in distributions is detected.
- The multivariate version of the Wald–Wolfowitz test [57] constructs a complete graph, with examples as vertices and distances between them as edges. This graph is then transformed into a forest and a test statistic is computed basing on the amount of trees.

Furthermore, non-parametric univariate statistical tests are often used for detecting concept drift in data distribution [160]:

- Two-sample Kolmogorov–Smirnov test,
- Wilcoxon rank sum test,
- Two-sample *t*-test.

Unfortunately, it is easy to show that without access to class labels the real drift could be undetected [163] if they are not associated to changes in $P(x)$.

As yet not so many papers deal with combined drift detectors. Bifet et al. [21] proposed the simple combination rules based on the appearance of drift once ignoring signals about warning level.

It is worth mentioning *Drift Detection Ensemble* [119], where a small ensemble of detectors is used to make a decision about the drift and *Selective Detector Ensemble* [46] based on a selective detector ensemble to detect both abrupt and gradual drifts. Some experimental studies showed that simple detector ensembles do not perform better than simple drift detection methods [191].

3. Evaluation in data stream analysis

Proper evaluation of classifiers or regression models is a key issue in machine learning. Many evaluation measures, techniques for their experimental estimation and approaches to compare algorithms have already been proposed for static data scenarios. A comprehensive review is presented in [88].

In the context of data stream mining, especially in non-stationary environments, new solutions are needed. While evaluating predictive ability, it is necessary to consider both incremental processing as well as evolving data characteristics and the classifier reactions to changes. New classes may appear, feature space changes and decision rules lose relevance over time. Moreover, one should take into account computational aspects such as processing time, recovery of the model after the change, and memory usage. Fast updating of a learning model and gradual recovery is often more reasonable than gathering data for a longer period of time and trying to rebuild the model in a single time consuming step. Instead of examining point or average prediction measures of the classifier, one is usually more interested in tracking its working characteristics over the course of stream progression.

The authors of several papers often present graphical plots for a given dataset presenting the algorithms' functioning in terms of the chosen evaluation measure, such as e.g. training time, testing time, memory usage, and classification accuracy over time. By presenting the measures calculated after each data chunk or single example on the y-axis and the number of processed training examples on the x-axis, one can examine the dynamics of a given classifier, in particular, its reactions to concept drift. Such plots also nicely support a comparative analysis of several algorithms.

Additionally, one must also consider the availability of information regarding the true target values of incoming examples. The majority of current measures and evaluation techniques assume immediate or not too much delayed access to these labels. However, in some real life problems, this assumption is unrealistic.

It is also worth mentioning that a thorough evaluation of predictive models in non-stationary environments typically requires the use of not only real world data streams, but also data streams with artificially generated concept drifts. Real world data streams enable us to evaluate how helpful a predictive model is in real world situations. However, they usually do not allow us to know when exactly a drift occurs, or even if there are really drifts. This makes it difficult to provide an in depth understanding of the behaviour of predictive models or drift detection methods. Data streams with artificially induced drifts enable a more detailed analysis. Therefore, both real world data streams and data streams with artificially induced drifts are important when evaluating predictive models and concept drift detectors in non-stationary environments.

The comparison of algorithms proposed in the literature is not an easy task, as authors do not always follow the same recommendations, experimental evaluation procedures and/or datasets.

Below, we discuss the most popular evaluation measures and then their experimental estimation procedures.

3.1. Evaluation measures

The predictive ability of classifiers or regression models is usually evaluated with the same measure as proposed for static, non-online learning which are also the least computationally demanding ones. Below we list the most popular ones:

- **Accuracy** : the proportion of all correct predictions to the total number of examples, or its corresponding measure **classification error**, are the most commonly used for classification.
- **Mean square error** or absolute error is a typical measure for regression.
- **Sensitivity** of the class of interest (also called Recall or True Positive Rate) is accuracy of a given class.
- **G-Mean** : the geometric mean of sensitivity and specificity is often applied on class-imbalanced data streams to avoid the bias of the overall accuracy.
- **Kappa Statistic** : $K = \frac{p_0 - p_c}{1 - p_c}$, where p_0 is accuracy of the classifier and p_c is the probability of a random classifier making a correct prediction.
- **Generalized Kappa Statistics** such as Kappa M proposed in [20], which should be more appropriate than the standard Kappa Statistics for dealing with imbalanced data streams.

Furthermore, in the case of static data the area under the Receiver Operating Characteristics curve, or simply **AUC**, is a popular measure for evaluating classifiers both on balanced and imbalanced class distributions [54]. However, in order to calculate AUC one needs to sort scores of the classifiers on a given dataset and iterate through each example. This means that the traditional version of AUC cannot be directly computed on large data streams. The current use of AUC for data streams has been limited only to estimations on periodical holdout sets [76] or entire streams of a limited length [42]. A quite recent study [30] introduces an efficient algorithm for calculating *Prequential AUC*, suitable for assessing classifiers on evolving data streams. Its statistical properties and comparison against simpler point measures, such as G-mean or Kappa statistics, has been examined in [33].

When analyzing the performance of classifiers dedicated to drifted data, we should also take into consideration their adaptation abilities, i.e., evaluating the maximum performance deterioration and restoration time, as mentioned in Section 2.4.

Apart from the predictive accuracy or error, the following performance metrics should be monitored and taken into account during properly executed evaluation of streaming algorithms:

- **Memory consumption**: it is necessary to monitor not only the average memory requirements of each algorithm, but also their change over time with respect to actions being taken.
- **Update time**: here one is interested in the amount of time that an algorithm requires to update its structure and accommodate new data from the stream. In an ideal situation, the update time should be lower than the arrival time of a new example (or chunk of data).
- **Decision time**: amount of time that a model needs to make a decision regarding new instances from the stream. This phase usually comes before the updating procedure takes place. So, any decision latency may result in creating a bottleneck in the stream processing. This is especially crucial for algorithms that cannot update and make predictions regarding new instances at the same time.

Nevertheless, in order to calculate reaction times and other adaptability measures, usually a human expert needs to determine moments when a drift starts and when a classifier recovers from it. Alternately, such evaluations are carried out with synthetic data generators.

More complex measures have also been proposed to evaluate other properties of algorithms. Shaker and Hüllermeier [158] proposed a complete framework for evaluating the recovery rate of the algorithm once a change has occurred in the stream. They consider not only how well the model reduced its error in the new decision space, but also what was the time necessary to achieve this. Zliobaite et al. [207] introduced the notion of cost-sensitive update in order to evaluate the potential gain from the cost (understood as time and computational resources) put into adapting the model to the current change. The authors argue that this allows to check if the actual update of the model was a worthwhile investment. Hassani et al. [71] proposed a new measure for evaluating clustering algorithms for drifting data streams, with special attention being paid to the behavior of micro-clusters.

3.2. Estimation techniques

In the context of static and batch learning the most often used scenario for estimating prediction measures is cross validation. However, in the context of online learning with computationally strict requirements and concept drifts, it is not directly applicable. Other techniques are considered. Two main approaches are used depending whether the stream is stationary or not, as shown below.

- **Holdout evaluation**: In this case two sub-sets of data are need: the training dataset (to learn the model) and the independent holdout set to test it. It is arranged that, at any given moment of time when we want to conduct model evaluation, we have at our disposal a holdout set not previously used by our model. By testing the learning model on such a continuously updated set (it must be changed after each usage to ensure that it represents the current concept well), we obtain an unbiased estimator of the model error. When conducted in a given time or instance interval, it allows us to monitor the progress of the model.
- **Prequential evaluation** is a sequential analysis [177] where the sample size is not fixed in advance. Instead, data are evaluated as they are collected. Predictive sequential evaluation, or prequential, also referred to as interleave train and test, follows the online learning protocol. Whenever an example is observed, the current model makes a prediction; when the system receives feedback from the environment, we can compute the loss function.

  Prequential measures can be calculated only for selected instances, thus allowing to accommodate the assumption of limited label availability. On the other hand, simply calculating a cumulative measure over the entire stream may lead to strongly biased results. One may easily imagine a situation in which the overall cumulative evaluation is strongly influenced by a certain time period, when, e.g., access to training data was limited, the decision problem was much more simple, or drift was not present. Thus, to make the error estimation more robust to such cases, a proper forgetting mechanism must be implemented – sliding windows or fading factors. With this, an emphasis is put on error calculation from the most recent examples. Indeed the term prequential (combination of words predictive and sequential) stems from online learning and is used in the literature to denote algorithms that base their functioning only on the most recent data. Prequential accuracy [63] is popularly used with supervised learning, but also a prequential version of AUC metric was proposed by Brzezinski and Stefanowski [30], being suitable for streams with skewed distributions. This issue was

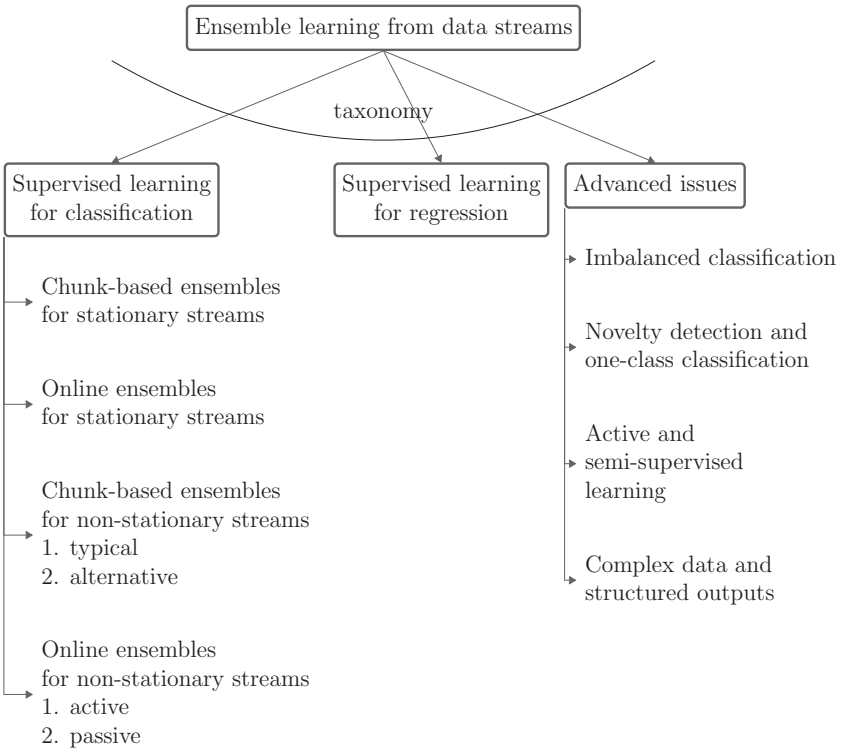

Fig. 6. The taxonomy of ensemble learning methods for data streams discussed thorough this survey.

also addressed by Bifet and Frank [12], who also proposed a prequential modification of kappa statistic suitable for streams.

A more elaborated approach to evaluate and compare algorithms in streaming scenarios have been introduced recently. Shaker and Hüllermeier [158] proposed an approach, called *recovery analysis*, which uses synthetic datasets to calculate classifier reaction times. The authors proposed to divide a dataset with a single drift into two sets without drifts. Afterwards, they propose to plot the accuracy of the tested classifier on each of these datasets separately. The combination of these two plots is called the optimal performance curve and serves as a reference that can be compared with the accuracy plot of the classifier on the original dataset. Zliobaite proposed to use modify a real stream by *controlled permutations* to better study the reaction of classifiers to drifts [201]. Recently Bifet at al. considered a prequential and parallel evaluation strategy inspired by cross-validation, which switches new incoming examples between copies of classifiers – some of them use it for updating while others for testing [12].

Statistical tests have gained a significant popularity in the machine learning community [66]. In the area of data streams there were few approaches to using these tools [20]. However, they usually concentrated on applying standard tests over the averaged results or by using sliding window technique. One may be critical to such approaches, as they either try to transform a dynamic problem into a static one, or take under consideration only local characteristics. So far, there has been no unified statistical testing framework proposed for data streams that would seem fully appropriate.

4. Ensemble learning from data streams

This section discusses supervised data stream ensemble learning approaches for classification and regression problems. To organize the subjects discussed in this survey and to offer a navigation tool for the reader, we summarize the proposed taxonomy of ensemble learning approaches for data streams in Fig. 6. Content presented there will be discussed in detail in Sections 4 and 5, with in-depth presentation of advances in the respective areas. Here, we would like to explain a disproportion in the subcategories between supervised learning in classification and regression problems. Theoretically, the same taxonomy used for the classification ensembles could be used for the regression ones. However, as there are still very few methods developed in this area, we have opted for not proposing a separate taxonomy for the streaming regression ensembles yet.

4.1. Supervised learning for classification problems

Ensembles are the most often studied new classifiers in the data stream community, see e.g. lists of methods in [43,60]. The proposed stream classifiers can be categorized with respect to different points of view. The most common categorizations are the following:

- stationary vs. non-stationary stream classifiers,
- active vs. passive approaches,
- chunk based vs. on-line learning modes,
- distinguishing different techniques for updating component classifiers and aggregating their predictions.

Approaches for stationary environments do not contain any mechanism to accelerate adaptation when concept drift occurs. Approaches for non-stationary environments are approaches specifically designed to tackle potential concept drifts.

When studying approaches to tackle concept drift, researchers usually distinguish between *active* vs. *passive* (also called *trigger* vs. *adaptive*) approaches, see e.g. a discussion in [43,169,200]. Active algorithms use special techniques to detect concept drift which trigger changes or adaptations in classifiers (e.g., rebuilding it from the recent examples) – see the discussion in earlier Section 2.4. Passive approaches do not contain any drift detector and continuously update the classifier every time that a new data item is presented (regardless whether real drift is present in the data stream

or not). The majority of current ensembles follow a passive schema of adaptation, while triggers are usually used mainly with single online classifiers. A few rare cases of integrating them with ensembles, such as ACE [133], BWE [38] or DDD [127], will be further discussed.

Then, with respect to the way of processing examples, the classifiers can be categorized into chunk-based approaches and online learning approaches. *Chunk-based approaches* process incoming data in chunks, where each chunk contains a fixed number of training examples. The learning algorithm may iterate over the training examples in each chunk several times. It allows to exploit batch algorithms to learn component classifiers. *Online learning approaches*, on the other hand, process each training examples separately, upon arrival. This type of approach is intended for applications with strict time and memory constraints, or applications where we cannot afford processing each training example more than once, e.g., applications where the amount of incoming data is very large.

It is worth noting that the above categorization does not mean that chunk-based approaches must be used only for situations where new training examples arrive in chunks. They can also be used to learn training examples that arrive separately, because each new training example can be stored in a buffer until the size of this buffer reaches the size of the chunk. Then, chunk-based approaches may process all these examples stored in the buffer. Similarly, this categorization does not mean that online learning approaches must be used only for situations where new training examples arrive separately, one-by-one. Online learning approaches can process each training example of a chunk separately. They can be used for applications where training examples arrive in chunks.

Finally, considering different strategies for re-constructing ensemble component classifiers and aggregating their predictions, one can recall Kuncheva's categorization [107], where she has distinguished the following four basic strategies:

- Dynamic combiners – component classifiers are learnt in advance and are not further updated; the ensemble adapts by changing the combination phase (usually by tuning the classifier weights inside the voting rule, e.g., the level of contribution to the final decision is directly proportional to the relevance [86,117]). The drawback of this approach is that all contexts must be available in advance; emergence of new unknown contexts may result in a lack of experts.
- Updating training data – recent training examples are used to online-update component classifiers (e.g. in on-line bagging [137] or its further generalizations [16,180]).
- Updating ensemble members – updating online or retraining in batch mode (using chunks) [15,55,100,136,150].
- Structural changes of the ensemble – replacing the worst performing classifiers in the ensemble and adding a new component, e.g., individual classifiers are evaluated dynamically and the worst one is replaced by a new one trained on the most recent data [84,98]

In this paper, the main criterion used to categorize classification ensemble approaches is the data processing method, i.e., whether examples are processed in chunks or one-by-one. Then, as the second criterion we use information on whether the approaches are designed to deal with stationary or non-stationary data streams. We consider these two criteria first because approaches within each of these categories tackle different types of data stream applications. Within each of these categories, we will then use further criteria to distinguish among existing approaches.

Section 4.1.1 presents chunk-based ensemble approaches for stationary environments, Section 4.1.2 presents online learning approaches for stationary environments, Section 4.1.3 presents chunk-based ensemble approaches for non-stationary environments, and Section 4.1.4 presents online learning approaches for non-stationary environments.

Table 1
Chunk-based ensembles for stationary data streams.

Algorithm	Description
Learn^{++} [143]	Incremental neural network ensemble
Ada.Boost RAN-LTM [92]	Combination of AdaBoost.M1 and RAN-LTM classifier
Growing NCL [124]	Incremental version of the Negative Correlation Learning
Bagging^{++} [197]	Training classifiers with Bagging from incoming chunks of data

4.1.1. Chunk-based ensembles for stationary streams

Chunk-based ensembles for stationary data streams are not so well developed as online versions and did not receive so significant attention from the research community. They are also related to the issue of batch processing of larger sets of data, and often do not explicitly refer to this as stream mining. This section reviews the most popular methods in this area. They are summarized in Table 1.

Learn^{++} is one of the most well recognized approaches to stationary streams [143]. This ensemble constructs new neural network models on each incoming chunk of data, and then combines their outputs using majority voting. This allows to accommodate new incoming instances into the ensemble. This approach however retains all previously learned classifiers, thus being inefficient for handling massive datasets as the size of the ensemble continuously grows.

Kidera et al. [92] proposed a combination of AdaBoost.M1 and Resource Allocating Network with Long-Term Memory, a stable neural network classifier for incremental learning. They used a predetermined number of base classifiers for the entire stream processing and incrementally updated them with new chunks. They suppressed the forgetting factor in these classifiers in order to allow an efficient weight approximation for weighted voting combination. This however limits the usability of this approach for potentially unbounded streams.

Minku et al. [124] introduced an incremental version of Negative Correlation Learning that aimed at co-training an ensemble of mutually diverse and individually accurate neural networks. At the same time their proposed learning scheme allowed to maintain a trade-off between the forgetting rate and adapting to new incoming data. Two models were discussed: fixed size and growing size, differing in their approach to maintaining the ensemble set-up. Experimental results showed that the fixed size approach has better generalization ability, while the growing size may easily overcome the impact of too strong forgetting.

Bagging^{++} [197] was developed as an improvement over Learn^{++} by utilizing Bagging to construct new models from incoming chunks of data. Additionally, the ensemble consisted of heterogeneous classifiers selected from a set of four different base classifiers. Authors showed that their approach gives comparable results to Learn^{++} and Negative Correlation Learning, while being significantly faster.

4.1.2. Online ensembles for stationary streams

Online ensembles for stationary data streams have gained significantly more attention than their chunk-based counterparts. This was caused by a general popularity of online learning and its application to various real-life scenarios, not only limited to streaming data. Let us review the most representative proposals in this area. They are summarized in Table 2.

Oza and Russel [137] introduced Online Bagging, which alleviates the limitations of standard Bagging of requiring the entire

Table 2
Online ensembles for stationary data streams.

Algorithm	Description
Bagging-based	
OzaBag [137]	Online Bagging
ASHT [17]	Ensemble of adaptive-size Hoeffding trees
LevBag [16]	Leveraging Bagging with increased resampling and output detection codes
ORF [41,153]	Online Random Forest
MF [111]	Online Mondrian Forest
Boosting-based	
OzaBoost [137]	Online Boosting
Others	
UFFT [61]	Ultra fast forest of binary trees
HOT [60]	Hoeffding Option Trees
EOS-ELM [112]	Ensemble of online extreme learning machines

training set available beforehand for learning. They assumed that, in online learning, each new incoming instance may be replicated zero, one or many times during the update process of each base classifier. Thus each classifier in the ensemble is updated with k copies of the newly arrived instance. The value of k is selected on the basis of Poisson distribution, where $k \sim Poisson(1)$. This comes from the fact that for potentially unbounded data streams the binominal distribution of k in standard Bagging tends to this specific Poisson distribution. Theoretical foundations of this approach were further developed by Lee and Clyde [113]. They proposed a Bayesian Online Bagging that was equivalent to the batch Bayesian version. By combining it with a lossless learning algorithm, they obtained a lossless online bagging approach.

Bifet et al. introduced two modifications of Oza's algorithm called Adaptive-Size Hoeffding Trees (ASHT) [17] and Leveraging Bagging [16], which aim at adding more randomization to the input and output of the base classifiers. ASHT synchronously grows trees of different sizes, whereas Leveraging Bagging increases resampling from $Poisson(1)$ to $Poisson(\lambda)$ (where λ is a user-defined parameter) and uses output detection codes [16].

Another online ensemble developed by Oza and Russel is Online Boosting [137]. This ensemble maintains a fixed size set of classifiers trained on the examples received so far. Each new example is used to update each of the classifiers in a sequential manner. Examples misclassified by the former classifiers in the sequence have their weights updated so as to be emphasized by the latter classifiers. This is done in the following way. For each new incoming example, one initially assigns the highest possible weight $\lambda = 1$ to it. The first classifier in the pool is updated with this example $k = Poisson(\lambda)$ times. After the update, this classifier is used to predict this example, and the weighted overall fraction ϵ of examples that it misclassified is updated. If the example is correctly classifies the example, the example's weight λ is multiplied by $\frac{1}{2(1-\epsilon)}$. If this classifier misclassified the example, we multiply the weight associated to this example by $\frac{1}{2\epsilon}$. This procedure is then repeated for the next classifier in the pool, but using the new weight λ.

Several researchers developed ensembles based on a combination of decision trees. Hoeffding Option Trees (HOT) can be seen as an extension of Kirkby's Option Tree [142]. It allows each training example to update a set of option nodes rather than just a single leaf. It provides a compact structure that works like a set of weighted classifiers, and just like regular Hoeffding Trees, they are built in an incremental way – for a more detailed algorithm refer to its description in [25,60].

Ultra Fast Forest of Trees, developed by Gama and Medas [61], uses an ensemble of Hoeffding trees for online learning. Their split criterion is applicable only to binary classification tasks. To handle multi-class problems, a binary decomposition is applied. A binary tree is constructed for each possible pair of classes. When a new instance arrives, each classifier is updated only if the true class label for this instance is used by the binary base classifier.

Ensemble of Online Extreme Learning Machines [112] was developed by Lan et al. It is a simple combination of online randomized neural networks, where initial diversity of the pool is achieved by a randomized training procedure. Base models are combined using averaging of individual outputs. Each base model is updated with the incoming instances, but no discussion of verification of how the diversity in the ensemble is maintained during the course of stream processing was given.

Some other researchers focused their work on proposing online versions of the popular Random Forest algorithm [41,153]. They introduced online Random Trees that generate test functions and thresholds at random and select the best one according to a quality measure. Their online update methodology is based on the idea of generating a new tree having only one root node with a set of randomly selected tests. Two statistics are calculated online: minimum number of instances before split and minimum gain to be achieved. When a split occurs statistics regarding the instances that will fall into left and right node splits are propagated into children nodes, thus they start already with the knowledge of their parent node. Although the authors acknowledge the existence of the Hoeffding bound, they argue that using online updated gain is closer to the real idea behind decision trees. Additionally, a forgetting mechanism via temporal knowledge weighting is applied to reduce the influence of old instances. This is realized as pruning random trees, where a classifier is discarded from the ensemble based on its out-of-bag error and the time its age (time spend in the ensemble).

This idea was further developed by Lakshminarayanan et al. into online Mondrian Forest algorithm [111]. They used Mondrian processes for their tree induction scheme, which are a family of random binary partitions. As they were originally introduced as infinite structures, the authors modified them into finite Mondrian trees. The main differences between this approach and standard decision trees are the independence of splits from class labels, usage of split time at every node, introduction of parameter controlling dynamically the number of nodes and that the slit is bounded by the training data and is not generalized over the entire feature space. The ensemble is constructed identically as in standard Random Forest, but another difference lies in online update procedure. Mondrian trees can accommodate new instances by creating a new split that will be on higher level of tree hierarchy than existing ones, extending the existing split, or splitting the existing leaf into children nodes. Please note that standard online Random Forest can only update their structure using the third of mentioned methods. This makes Mondrian Forests much more adaptable to streaming data, allowing for more in-depth modifications in ensemble structure. The authors report that their method outperforms existing online Random Forests, achieves accuracy similar to batch versions and is at least an order of magnitude faster than reference ensembles.

4.1.3. Chunk-based ensembles for non-stationary streams

Chunk-based approaches for non-stationary environments usually adapt to concept drifts by creating new component (a.k.a. base) classifiers from new chunks (blocks or batches) of training examples. In general, component classifiers of the ensemble are constructed from chunks which correspond to different parts of the stream. Therefore, the ensemble may represent a mixture of different distributions (concepts) that have been present in the data stream. Learning a new component from the most recent chunk is also a natural way of adaptating to drifts [200]. Additionally, some chunk-based ensembles maintain an additional buffer for storing old classifiers that can be reused when needed, offering a potential to handle recurring concepts.

Table 3
Chunk-based ensembles for non-stationary data streams.

Algorithm	Description
Typical approaches	
SEA [170]	Streaming Ensemble Algorithm
AWE [178]	Accuracy Weighted Ensemble
Aboost[36]	Adaptive, fast and light Boosting
Learn^{++}.NSE [50]	Learn^{++} for non-stationary environments
Alternative approaches	
KBS [154]	Boosting-like method using knowledge-based sampling
AUE [31]	Accuracy Updated Ensemble
WAE [189]	Weighted Aging Ensemble
BWE [38]	Batch Weighted Ensemble
ET [146]	Ensemble tracking for recurring concepts

Learning component classifiers from complete chunks enables applying standard, batch learning algorithms. Forgetting of old classification knowledge can be done by eliminating too poorly performing components. This offers a way to limit the amount of memory required to store the ensemble, even though it impedes the ensemble of recovering deleted classifiers if and when their corresponding concept reoccurs.

Most of the chunk-based ensembles periodically evaluate their components with the newest chunk. The results of this evaluation are used to update weights associated to each component classifier. These weights can be used to emphasise the classifiers that best reflect the most recent data distribution when making an ensemble prediction, or to decide which unhelpful classifiers should be discarded.

One of the main features to distinguish between different chunk-based ensembles for non-stationary environments is whether or not they always create new classifiers for each new chunk of data in order to deal with concept drift. So, we discuss these approaches under this perspective below. Presented algorithms are summarized in Table 3.

Typical Chunk-based Approaches. Typically, chunk-based ensembles are constructed according to the following schema:

1. For each new chunk $B_i \in S$, evaluate component classifiers C_j in the ensemble with respect to a given evaluation measure $Q(C_j)$;
2. Learn a new candidate classifier C_c using B_i;
3. Add C_c to the ensemble if the ensemble size is not exceeded; otherwise replace one of the existing components of the ensemble.

Each of these approaches implements a different strategy to restrict the ensemble size and to weight different classifiers in the ensemble.

As a new classifier is always created to learn each new data chunk, the size of the chunk plays a particularly important role. A too large chunk size would result in slow adaptation to drifts. On the other hand, a too small chunk size would not be enough to learn an entire stable concept well, would increase computational costs, and may result in poor classification performance [178].

One of the earliest well known approaches in this category is the Streaming Ensemble Algorithm (SEA), proposed by Street and Kim [170]. This approach creates a new classifier to learn each new chunk of training data. If the maximum ensemble size has not been reached yet, this new classifier is simply added to the ensemble. Otherwise, the quality of the new classifier is first evaluated based on the next incoming training chunk. Then, the new classifier replaces an existing classifier whose quality is worse than the quality of the new classifier on this training chunk. One of the key features for the success of this approach is its quality measure. It favours the classifiers which correctly classify examples that are nearly undecided by the ensemble. In this way, this approach can avoid overfitting and maintain diversity. The predictions given by the ensemble are based on the majority voting. This approach has been shown to recover faster from concept drift than single classifiers. One of its potential problems is that old classifiers can outweigh the new classifier, potentially slowing down adaptation to new concepts. How fast the ensemble can recover from drifts depends not only on the chunk size, but also on the ensemble size.

A similar way of restructuring an ensemble was proposed by Wang et al. as the algorithm called Accuracy Weighted Ensemble (AWE) [178]. The key idea of AWE is to assign weights to each classifier of the ensemble based on their prediction error on the newest training chunk. A special variant of the mean square error (which allows to deal with probabilities of a component classifier predictions) is used for that purpose. The assumption made by this approach is that the newest training chunk is likely to represent the current test examples better. Classifiers that have equal or worse performance than a random classifier (in terms of their mean square errors) are discarded. Pruning can also be applied to maintain only the K classifiers with the highest weights. In this way, it is possible to remove classifiers that would hinder the predictions and include new classifiers that can learn the new concepts. For cost-sensitive applications, it is also possible to use instance-based dynamic ensemble pruning [51]. This approach was shown to be successful in achieving better accuracy than single classifiers when the ensemble size becomes large enough (i.e., after enough data chunks are received). However, as noticed in [27], the AWE's pruning strategy may sometimes delete too many component classifiers in the case of certain sudden drifts and decrease too much of AWE's classification accuracy. Another problem concerns the evaluation of the new candidate classifier – it requires k-fold cross-validation inside the latest chunk, which increases computational time.

Chu and Zaniolo [36] proposed a chunk-based approach inspired by the boosting framework. When a training chunk is received, the ensemble error is calculated. After that, a mechanism based on statistical tests is used to detect concept drifts. If a concept drift is detected, all the classifiers composing the ensemble are deleted. After the concept drift detection mechanism is applied (and the possible deletion of ensemble members), a new classifier is created to learn the training chunk. The training examples of the chunk are associated to weights determined in an AdaBoost way based on the ensemble error. If the ensemble error on the current chunk is e and the example i is misclassified, then this example's weight is set to $w_i = (1 - e)/e$. If the example was correctly classified, its weight is maintained as 1. If the inclusion of the new classifier makes the ensemble exceed the maximum size M, the oldest ensemble member is eliminated. The classification is done by averaging the probabilities predicted by the classifiers and selecting the class with the highest probability. This approach was shown to be able to improve predictive performance in comparison to previous approaches such as SEA [170] and Wang et al.'s [178] in the presence of concept drift. A potential problem of this approach is that it resets the whole ensemble upon drift detection. This strategy can be sensitive to false alarms (false positive drift detections) and is unable to deal with recurring concepts.

Another approach inspired by the boosting framework is Elwell and Polikar's generalization of Learn++ for Non-Stationary Environments (called Learn++.NSE) [50]. This approach also sets the weights of the training examples from a new data chunk based on the ensemble error on this chunk. If an example i is misclassified, its weight is set to $w_i = 1/e$. Otherwise, it is set to 1. One of the main differences between this approach and Chu and Zaniolo's [36] is that it does not use a concept drift detection mechanism. Instead, reaction to drifts is based on weights associated to each base classifier. These weights are higher when the corresponding base classifier is able to correctly classify examples that were mis-

classified by the ensemble. Weights are lower if the corresponding base classifier misclassifies examples that were correctly classified by the ensemble. Weights are also set to give more importance to the misclassifications on more recent data chunks, which are believed to represent the current concept better. The predictions given by the ensemble are based on weighted majority voting. Therefore, base classifiers that were poorly performing for some period of time can be automatically re-emphasised through their weights once they become useful. The fact that base classifiers are not deleted can help dealing with recurrent drifts. However, as the ensemble size is unlimited and a new base classifier is added for every new data chunk, the number of base classifiers may become high.

Alternative Chunk-Based Approaches. Chunk-based ensembles are typically quite sensitive to a proper tuning of the size of the data chunk. In particular, a too large chunk size may delay reaction to drifts, while a too small chunk size may lead to poorly performing base classifiers. Moreover, learning every new data chunk may introduce a learning overhead that could be unnecessary when existing classifiers are considered good enough for the current concept. Some researchers proposed approaches that deviate from the typical chunk-based learning schema in an attempt to overcome some of these issues. We discuss some representative approaches in this section.

Scholz and Klinkenberg's approach [154,155] decides, for each new training chunk, whether to train a new classifier or update the newest existing classifier with it. This decision is based on the accuracy resulting from training the most recent classifier with the new chunk in comparison with the accuracy obtained by training a new classifier on the new chunk. Only the best between these two classifiers is kept. This strategy may reduce the problem of creating poor base classifiers due to small chunk sizes, because existing classifiers can be trained with more than one chunk. Besides assigning weights to the examples within a training chunk in a boosting-like style, each classifier itself also has a weight, which is assigned depending on its performance on the new training chunk. These weights are not only used to speed up reaction to concept drifts, but also to prune unhelpful classifiers. This approach has been shown to perform well in comparison to previous approaches such as adaptive window size [95] and batch selection [94,96]. However, it did not perform so well when the drift consisted of an abrupt concept drift quickly followed by a change back to the previous concept.

Deckert [38] proposed an ensemble approach that uses a concept drift detection method to decide whether a new classifier should be created to learn a new data chunk, or whether the new data chunk should be discarded without further training.

Another alternative chunk-based approach is the Accuracy Updated Ensemble (AUE) [27,31]. In this ensemble, all component classifiers are incrementally updated with a portion of the examples from the new chunk. This may help reducing the problems associated to creating poor base classifiers due to small chunk sizes. Another novelty includes weighting classifiers with non-linear error functions, which better promotes more accurate components. Moreover, the newest candidate classifier always receives the highest weight, as it should reflect the most recent data distribution better. AUE also contains other techniques for improving pruning of ensembles and achieving better computational costs. The experimental studies [31] showed that AUE constructed with Hoeffding Trees obtained higher classification accuracy than other chunk ensembles in scenarios with various types of drifts as well as in stable streams.

Yet another approach to rebuilding a chunk-based ensemble was presented by Wozniak et al. Weighted Aging Ensemble (WAE) modifies the classifier ensemble line-up on the basis of their diversity. The ensemble prediction is made according to the weighted majority voting, where the weight of a given classifier depends on its accuracy and time spent inside an ensemble [189].

Table 4
Online ensembles for non-stationary data streams.

Algorithm	Description
Passive approaches	
DWM [100]	Dynamic Weighted Majority
AddExp [99]	Addictive expert ensembles for classification
HRE [107]	Horse racing ensembles
CDC [168]	Concept Drift Committee
OAUE [29]	Online Accuracy Updated Ensemble
WWH [194]	Ensemble of classifiers using overlapping windows
ADACC [83]	Anticipative Dynamic Adaptation to Concept Changes
Active approaches	
ACE [133]	Adaptive Classifiers-Ensemble
Todi [132]	Two Online Classifiers For Learning And Detecting Concept Drift
DDD [127]	Diversity for Dealing with Drifts
ADWINBagging [18]	Online Bagging with ADWIN drift detector

A number of approaches have been discussed in the literature to specifically tackle recurring concepts in data streams. Ramamurthy and Bhatnagar [146] proposed an ensemble tracking approach that tries to deal with recurring concepts explicitly. It maintains a *global set* of classifiers representing different concepts. Whenever a new training chunk is available, the error of each classifier on it is determined. *MaxMSE* is defined as the classification error of a classifier that predicts randomly. If at least one classifier has error lower than a pre-defined value τ, or if the error of the weighted ensemble formed by all classifiers *with error lower than AcceptanceFactor*MaxMSE* is lower than τ, no new classifier is created. This reduces the overhead associated to learning every new data chunk. If neither a single classifier nor the above mentioned ensemble have error lower than τ, a new classifier is created and trained with the new data chunk, which is assumed to represent a new concept. One of the problems of this approach is that it has no strategy to limit the size of the global set of classifiers.

Another approach for storing the special definitions of previous concepts has been considered by Katakis et al. in their ensemble with conceptual clusters calculated and compared for each data chunk [91]. Jackowski [84] described an evolutionary approach for selecting and weighting classifiers for the ensemble in the presence of recurrent drifts, while Sobolewski and Wozniak used the idea of the recurring concepts to generate a pool of artificial models and select the best fitted in the case of concept drift [165].

4.1.4. Online ensembles for non-stationary streams

Online ensembles learn each incoming training example separately, rather than in chunks, and then discard it. By doing so, these approaches are able to learn the data stream in one pass, potentially being faster and requiring less memory than chunk-based approaches. These approaches also avoid the need for selecting an appropriate chunk size. This may reduce the problems associated with poor base models resulting from small chunk sizes, even though these approaches would still normally have other parameters affecting the speed of reaction to drifts (e.g., parameters related to sliding windows and fading factors).

One of the main features to distinguish between different online ensemble learning approaches for non-stationary environments is the use of concept drift detection methods. So, they are divided into passive or active categories. Presented algorithms are summarized in Table 4.

Passive Approaches. Passive approaches are approaches which do not use explicit concept drift detection methods. Different passive online ensembles have different strategies to assign weights to classifiers, as well as to decide when to add or remove classifiers from the ensemble in order to react to potential concept drifts. Most of these approaches present mechanisms to continuously adapt to concept drifts that may occur in the stream. How fast adaptation is achieved and how sensitive this adaptation is to noise usually depends on parameters.

One of the most well known approaches under this category is Dynamic Weighted Majority (DWM) [100], proposed by Kolter and Maloof. In this approach, each classifier has a weight that is reduced by a multiplicative constant β ($0 \leq \beta < 1$) when it makes a wrong prediction, similar to Littlestone and Warmuth's Weighted Majority Algorithm [117]. This allows the ensemble to emphasize the classifiers that are likely to be most accurate at a given point in time. All classifiers are incrementally trained on the incoming training examples. In addition, in order to accelerate reaction to concept drift, it is possible to add a new classifier or remove existing classifiers. New classifiers are added when the ensemble misclassifies a given training example. They can learn potentially new concepts from scratch, avoiding the need for existing classifiers to forget their old knowledge when there is concept drift. Classifiers whose weights are too low are classifiers that have been unhelpful for a long period of time. They can be deleted to avoid the ensemble becoming too large. The weight updates and the addition and removal of classifiers are performed only at every p time steps, where p is a pre-defined value. Larger values of p are likely to be more robust against noise. However, too large p values can result in slow adaptation to concept drift. At every p training examples, the weights of all ensemble members are also normalized, so that the new member to be included does not dominate the decision-making of all the others. DWM has demonstrated to achieve good performance in the presence of concept drifts [100], usually achieving similar performance to an approach with perfect forgetting. However, it may not perform so well as Littlestone and Warmuth's Weighted Majority Algorithm [117] under stationary conditions.

Addictive Expert Ensembles (AddExp) is a method similar to DWM [99]. The main motivation for this method is the fact that it allows the definition of mistake and loss bounds. In this method, the parameter p is eliminated, so that weight updates happen whenever a base classifier misclassifies a new training example. A new classifier is always added when the prediction of the ensemble as a whole is wrong. When combined with a strategy to prune the oldest classifiers once a maximum pre-defined ensemble size if reached, the bounds are defined in the same way as when no pruning of classifiers is performed. However, eliminating the oldest classifiers may not be a good strategy to deal with non-stationary environments, as old classifiers may still be very useful. The alternative strategy of pruning the lowest weight classifiers is more practical, but offers no theoretical guarantees.

Other approaches to combine online classifiers are also considered in Hedge β or Winnow algorithm [117]. Kuncheva called them "horse racing" ensembles [107]. For instance, Hedge β works in a similar way to the Weighted Majority Algorithm, but instead of using an aggregating rule it selects one component classifier based on the probability distribution obtained by normalized weights to represent the final ensemble prediction. Winnow also follows the main schema of Weighted Majority Algorithm, but uses different updating and calculating weights ideas.

Another example of passive online learning ensemble approach for non-stationary environments is Stanley's Concept Drift Committee (CDC) [168]. As with DWM and AddExp, all classifiers that compose the ensemble are trained on the incoming training examples. Instead of multiplying the weights of the classifiers by a constant β upon misclassifications, CDC uses weights that are proportional to the classifier's accuracy on the last n training examples. A new classifier is added whenever a new training example becomes available, rather than only when the ensemble misclassifies the current training example. When a maximum pre-defined ensemble size is reached, a new classifier is added only if an existing one can be eliminated. A classifier can be deleted if its weight is below a pre-defined threshold t and its age (number of time steps since its creation) is higher than a pre-defined maturity age. Imature classifiers do not contribute to the ensemble's prediction. This gives them a chance to learn the concept without hindering the ensemble's generalization. This approach was shown to achieve comparable or better performance than previous approaches such as FLORA4 [184] and instance-based learning 3 (IB3) [3] in the presence of concept drifts, but sometimes presented worse performance than FLORA4 before the drifts.

Yet another idea has been used in Online Accuracy Updated Ensemble (OAUE) [29]. It inherits some positive solutions coming from its hybrid preceder AUE, like incremental updating of component classifiers and learning new classifiers at some time steps. However, to more efficiently process incoming single examples and weight component classifiers, the new proposal of a cost-effective function was introduced. It achieves a good trade-off between predictive accuracy, memory usage and processing time.

The WWH algorithm from Yoshida et al. [194] builds different component classifiers on overlapping windows to select the best learning examples and aggregates component predictions similarly to the Weighted Majority Algorithm. Therefore, WWH can be seen as a combination of an instance selection windowing technique with an adaptive ensemble.

Quite recently, Jaber proposed the Anticipative Dynamic Adaptation to Concept Changes (ADACC) ensemble, which attempts to optimize control over the online classifiers by recognizing concepts in incoming examples [83].

Active Approaches. Even though active online ensemble approaches are not so common as passive ones, there are a few approaches in this category. One of the advantages of using explicit drift detection methods is the possibility to inform practitioners of the existence of concept drifts. The use of concept drift detectors can also help approaches to swiftly react to concept drifts once they are discovered. However, if concept drift detectors fail to detect drifts, these approaches will be unable to react to drifts. Concept drift detectors may also present false alarms, i.e., false positive drift detections. Therefore, it is important for active ensemble approaches to implement mechanisms to achieve robustness against false alarms.

An example of active online ensemble is the Adaptive Classifiers-Ensemble (ACE) [133]. This approach uses both an online classifier to learn new training examples and batch classifiers trained on old examples stored in a buffer. The batch classifiers are used not only to make predictions, but also to detect concept drifts. ACE considers that there is a concept drift if the accuracy of the most accurate batch classifier over the last W examples is outside the confidence interval formed by its accuracy over the W examples preceding the last W examples. Whenever a concept drift is detected or the maximum number of training examples to be stored in the buffer is attained, a new batch classifier is trained with the stored examples and both the online classifier and the buffer are reset. A pruning method is used to limit the number of batch classifiers used. This pruning method removes older classifiers first, unless they present the highest predictive accuracy over a long period of time. In that way, the approach can use old knowledge when there are recurring concepts. The classification is done by weighted majority vote. The weight is based on the accuracy on

the most recent W training examples, and it is zero if this accuracy is equal to or lower than the lower endpoint of the accuracy confidence interval. As the size of the buffer of stored examples is independent of the size of the sliding window W, ACE can respond to sudden changes even if the buffer is large. However, determining the size W of the sliding window may not be easy. ACE also requires storage of examples in an incremental way to create the batch classifiers, but this issue can be easily overcome by replacing the buffer by an online learning algorithm. A comparative experiment of ACE against other ensembles has been presented in [39].

Two Online Classifiers For Learning And Detecting Concept Drift (Todi) [132] uses two online classifiers to detect concept drift. One of them (H_0) is rebuilt every time a drift is detected. The other one (H_1) is not rebuilt when a drift is detected, but can be replaced by the current H_0 if a detected drift is confirmed. Todi detects concept drift by performing a statistical test of equal proportions to compare H_0's accuracies on the most recent W training examples and on all the training examples presented so far excluding the last W training examples. After the detection of a concept drift, a statistical test of equal proportions with significance level β is done to compare the number of correctly classified training examples by H_0 and H_1 since the beginning of the training of H_0. If statistical significant difference is detected, this means that H_0 was successful to handle concept drift, and the drift is confirmed. H_0 then replaces H_1 and is rebuilt. The classification is done by selecting the output of the most accurate classifier considering the W most recent training examples. This strategy makes the approach more robust to false alarms than approaches that reset the learning system upon drift detection [62,134]. However, no strategy is adopted to accelerate the learning of a new concept, as the new concept has to be learnt from scratch.

Another example of active online ensemble learning approach in this category is Diversity for Dealing with Drifts (DDD) [127]. DDD is based on the observation that very highly diverse ensembles (whose base classifiers produce very different predictions from each other) are likely to have poor predictive performance under stationary conditions, but may become useful when there are concept drifts. So, in the mode prior to drift detection, DDD maintains both a low diversity ensemble and a high diversity ensemble. The low diversity ensemble is used for learning and for making predictions. The high diversity ensemble is used for learning and is only activated for predictions upon drift detection. This is because this ensemble is unlikely to perform well under stationary conditions. Concept drifts can be detected by using existing methods from the literature. Once a concept drift is detected, the approach shifts to the mode after drift detection, where it activates both the low and high diversity ensembles and creates new low and high diversity ensembles to start learning the new concept from scratch. The prediction given by DDD is then set to the weighted majority vote of the predictions given by its ensembles, except for the new high diversity ensemble. The weight of each ensemble is proportional to its prequential accuracy since drift detection. This approach manages to achieve robustness to different types of drift and to false alarms, because the different ensembles are most adequate for different situations. However, the use of more than one ensemble can make this approach heavier for applications with very strict time constraints.

Modifications of the architecture of tree ensembles with drift detectors have also been considered by Bifet at al.[13]. The ADWIN change detector has been used to reset ensemble members when their predictive accuracy degrades significantly. This makes it possible to better deal with evolving data streams. The same ADWIN method may also be integrated with online bagging ensemble – see ADWINBagging [18].

Table 5
Ensembles for regression from data streams.

Algorithm	Description
OzaBag [137]	Online Bagging for regression
OzaBoost [137]	Online Boosting for regression
AddExp [99]	Addictive expert ensembles for regression
ILLSA [90]	Incremental local learning soft sensing algorithm
eFIMT-DD [81]	Ensembles of any-time model trees
AMRules [47]	Ensemble of randomized adaptive model rules
iSOUP-Tree-MTR [135]	Ensembles of global and local trees
DCL [125]	Dynamic cross-company learning
Dycom [128]	Dynamic cross-company mapped model learning
LGPC [192]	Lazy Gaussian Process committee
OWE [162]	Online weighted ensemble of regressor models
DOER [161]	Dynamic and on-line ensemble regression

4.2. Supervised learning for regression problems

Regression analysis is a technique for estimating a functional relationship between a numeric dependent variable and a set of independent variables. It has been widely studied in statistics, pattern recognition, machine learning and data mining. Many ensemble methods can be found in the literature for solving classification tasks on streams, but only a few exist for regression tasks. Discussed algorithms are summarized in Table 5.

Oza and Russel's online bagging algorithm for stationary data streams [137] described in Section 4.1.2 is an example of method that is inherently applicable both to classification and regression.

Kolter and Maloof's Addictive Expert Ensembles (AddExp) for non-stationary data streams also contains another version for continuous dependent variables [99]. As in the AddExp for classification problems, a weight is associated to each base learner. For classification, AddExp makes predictions by using weighted majority vote, while for regression, weighted average is used. In the version for classification, the weight associated to a base classifier is multiplied by β, $0 \leq \beta < 1$, whenever it misclassifies a training example. In the version for regression, the weight of a base learner is always multiplied by $\beta^{|\hat{y}-y|}$, where $\hat{y}$ is the prediction given by the base learner is y is the actual value of the dependent variable.

Kadlec and Gabrys developed an incremental local learning soft sensing algorithm (ILLSA) [90], operating in two phases. During the initial phase a number of base models is being trained, each using different concepts (subsets) of the training data. During the online data stream mining phase, weights assigned to models are recalculated instance-by-instance using their proposed Bayesian framework working on output posterior probabilities.

The most in depth study on learning ensembles of model trees from data streams appears in [80,81]. These research include two different methods for online learning of tree-based ensembles for regression from data streams. Both methods are implemented on the top of single model trees induced using the FIMT-DD algorithm (a special incremental algorithm for learning any-time model trees from evolving data streams). Then, the ensembles of model trees are induced by the online bagging algorithm and consist of model trees learned with the original FIMT-DD algorithm and a randomized version named R-FIMT-DD. Authors explore the idea of randomizing the learning process through diversification of the input space and the search trajectory and examine the validity of the statistical reasoning behind the idea for aggregating multiple predictions. It is expected that this would bring the resulting model closer to the optimal or *best* hypothesis, instead of relying only on the success of a greedy search strategy in a constrained hypothesis space. The authors also perform a comparison with respect to the improvements that an option tree brings to the learning process.

In [82], the authors observe that the use of options acts as a kind of backtrack past selection decisions. Their empirical compar-

ison has shown that the best tree found within the option tree has a better accuracy (on most of the problems) than the single tree learned by FIMT-DD. The increased predictive performance and stability comes at the cost of a small increase of the processing time per example and a controllable increase in the allocation of memory. The increase in the computational complexity is due to the increased number of internal nodes being evaluated at any given point in time. The option tree incurs an additional increase in computational complexity when computing the aggregate of the multiple predictions for a single testing example, as it has to examine all of the options on the path from the root to the corresponding leaf node.

Adaptive Model Rules [47] is the first streaming rule learning algorithm for regression problems. It extends AMRules algorithm by using random rules from data streams. Several sets of rules are being generated. Each rule set is associated with a set of N_{att} attributes. These attributes are selected randomly from the full set of attributes of the dataset. The new algorithm improves the performance of the previous version.

Osojnik et al. [135] investigated ensembles of local and global trees for multi-target regression from data streams. Authors argued that predicting all target at once is more beneficial to mining streams than using local models. A novel global method was proposed, named incremental Structured Output Prediction Tree for Multi-target Regression (iSOUP-Tree-MTR). For improving the predictive power, the authors used it as a base learner for Oza's Online Bagging.

An approach called Dynamic Cross-company Learning (DCL) [125] has been proposed to perform transfer learning for data streams in non-stationary environments. The approach aims at making predictions in the context of a given target company or organization. A data stream containing training examples from this company or organization is available, but produces few examples over time. This can happen, for example, when it is expensive to collect labeled examples in the context of a given company. Therefore, this approach maintains not only a base learner to learn such examples, but also other base learners to learn examples obtained from other companies or organizations. A weight is associated to each base learner. This weight is multiplied by β, $0 \leq \beta < 1$, whenever this base learner is not the one that provided the best prediction to a new target company/organization training example. So, these weights can be used to emphasize the base learners that currently best reflect the present concept of the target company/organization. The prediction given by the ensemble is the weighted average of the predictions given by the base learners.

Another approach called Dynamic Cross-company Mapped Model Learning (Dycom) [128] extends DCL to learn linear functions to map the base learners created with data from other companies or organizations to the current concept of the target company or organization. These mapping functions are trained based on a simple algorithm that uses training examples from the target company/organization data stream and the predictions given to these examples by the base learners representing other companies/organizations. This algorithm operates in an online manner and gives more importance to more recent training examples, so that the mapping functions represent the current concept of the companies/organizations. It is expected to enable a reduction in the number of training examples required from the target company while keeping a similar predictive performance to DCL. This is because it can benefit from all base learners by mapping them to the concept of the target company, rather than benefiting only from base learners that currently best represent the concept of the target company.

Xiao and Eckert [192] proposed an approximation of Gaussian processes for online regression tasks. They combined several base models, each being initialized with random parameters. Each incoming instance is used to update a selected subset of base models that are being chosen using a reedy subset selection, realizing an optimization of a submodular function. The authors showed that their method displays favorable results in terms of error reduction and computational complexity, however used only methods based on Gaussian processes as a reference.

On-line Weighted Ensemble (OWE) of regressor models was discussed by Soares and Araujo [162]. It was designed to handle various types of concept drift, including recurrent ones. The ensemble model is based on a sliding window that allows to incorporate new samples and remove redundant ones. A boosting-like solution is used for weight calculation of ensemble models, by measuring their error on the current window. Additionally, contribution of old windows can be taken into consideration during weight calculation, thus allowing for switching between recurring and non-recurring environments. Finally, OWE can expand its structure by adding new model when the ensemble error is increasing and pruning models characterized by highest loss of accuracy.

This concept was further developed by the same authors in their dynamic and on-line ensemble regression (DOER) [161]. Here, the selection and pruning of models within the ensemble is being done dynamically, instance after instance, to offer improved adaptation capabilities. Additional novelty lies in ability of each base model to update its parameters during the stream mining procedure.

An evolutionary-based ensemble that can adapt the competence areas and weights assigned to base models for regression tasks was also discussed by Jackowski in [85].

5. Advanced issues in data stream analysis

The previous sections have discussed typical representations of examples and output values (as attribute - value pairs) and learning problems which are the commonly encountered in data stream analysis. However, in several new studied problems one can meet more complex representations or learning issues. We will now discuss ensemble solutions to these problems, including learning from imbalanced data, novelty detection, lack of counterexamples, active learning and non-standard data structures.

5.1. Imbalanced classification

Non-stationary data streams may be affected by additional data complexity factors besides concept drifts and computational requirements. In particular, it concerns class imbalance, i.e., situations when one of the target classes is represented by much less instances than other classes. Class imbalance is an obstacle even for learning from static data, as classifiers are biased toward the majority classes and tend to misclassify minority class examples. Dealing with unequal cardinalities of different classes is one of the contemporary challenges in batch learning from static data. It has been more studied in this static framework and many new algorithms have already been introduced, for their comprehensive review see the recent monograph [73] or surveys [72,101,172].

Out of these new solutions ensembles are one of the most promising directions. However, class imbalance has still received less attention in non-stationary learning [77]. Note that imbalanced data streams may not be characterized only by an approximately fixed class imbalance ratio over time. The relationships between classes may also be no longer permanent in evolving imbalanced streams. A more complex scenario is possible where the imbalance ratio and the notion of a minority class may change over time. It becomes even more complex when multi-class problems are being considered [181]. Below we discuss main ensemble-based proposals for mining imbalanced evolving streams. They are summarized in Table 6.

Table 6
Ensembles for imbalanced data streams.

Algorithm	Description
Chunk-based approaches	
SE [65]	Ensemble with majority class sampling
SERA [34]	Selectively recursive approach for sampling minority class
REA [35]	SERA with *k*-NN for chunk similarity analysis
BD [116]	Boundary definition ensemble
Learn^{++}.CDC [42]	Learn^{++} with concept drift and SMOTE
Online approaches	
EONN [67]	Ensemble of online cost-sensitive neural networks
ESOS-ELM [129]	Ensemble of subset online sequential extreme learning machines
OOB [180]	Oversampling-based online Bagging
UOB [180]	Undersampling-based online Bagging
MOOB [181]	Multi-class oversampling-based online Bagging
MUOB [181]	Multi-class undersampling-based online Bagging

Many of these proposals adapt an idea of re-sampling the data in incoming data to obtain more balanced class distributions. In general re-sampling methods transform the distribution of examples in the original data towards more balanced classes. Undersampling removes some examples from the majority classes while oversampling adds minority class examples (either by random replicating or generating synthetic new ones).

The first proposal by Gao et al. [65] was an ensemble approach that divided examples from the incoming data chunk into positive (the minority class) and negative (other classes) subsets. To build a new base classifier one takes all positive instances gathered so far and randomly selects a subset of the negative instances of the new data chunk. The size of this subset is calculated basing on a parameter referring to the class distribution ratio. Then, this new classifier is added to the ensemble. Predictions of base classifiers are combined using a simple voting technique. In order to accommodate this idea for a potentially infinite stream authors propose to sample examples from only a limited number of the most recent chunks, using either fixed (each chunk contributes equally) or fading (the more recent chunks contribute more instances) strategy. However, as all positive examples are used to learn each classifier, this method is limited to situations with a stable definition of the minority class.

Selectively recursive approach (SERA) [34] is another ensemble method proposed by Chen and He that extends the Gao et al. concept by using selective sampling of the minority class. Mahalanobis distance is used to select a subset of most relevant minority instances (from the previous chunks) for the current chunk of the stream and combine them with bagging method applied on examples from the majority class. This approach alleviates the drawbacks of the previous method regarding drifts on minority class, but at the same time makes SERA very sensitive to proper selection of the number of minority samples taken under consideration.

Chen and He proposed yet another ensemble, called REA [35], which changes SERA properties by adopting the k-nearest neighbor principle to estimate similarity between previous minority examples with ones in the most recent chunk. The predictions of base classifiers are weighted on the basis of their classification of the recent chunk.

Lichtenwalter and Chawla [116] proposed weighted ensembles in which both classified minority and majority instances are being propagated between chunks. This allows to better capture the potentially changing boundary between classes. A combination of information gain and Hellinger's distance (a skew-insensitive metric) is used to measure similarities between two data chunks and thus to implicitly check if a concept drift has taken place. This information is then used to weight ensemble members during the combination of their predictions, with a linear function being inverse of the actual closeness of chunks. The authors acknowledge the potential limitations of this approach (like small differences in weights or reduced variance) but leave a more precise examination of different combination functions for future studies.

Ditzler and Polikar [42] proposed an extension of their Learn^{++} ensemble for incremental learning from imbalanced data. This combines their previous approach to learning in non-stationary scenarios with idea of bagging, where undersampling is performed in each bag. Classifiers are weighted based on their performance on both minority and majority classes, thus preventing significant loss of accuracy on negative cases. However, one must point out that this approach assumes well-defined minority class and cannot handle dynamically changing properties of classes. The authors also studied a variant called Learn^{++}.CDC (Concept Drift with SMOTE), which employs oversampling of the minority class.

Ghazikhani et al. [67] introduced an ensemble of online neural networks to handle drifting and imbalanced streams. They embedded a cost-sensitive learning into the process of neural network training in order to tackle the skewed class distribution. A number of cost-sensitive neural networks is trained at the beginning of the stream using different initial random weights. Then, the ensemble is updated with new instances without set-up modifications. A cost matrix is predefined, with penalty for errors on minority class being twice the remaining costs. The usage of the fixed cost matrix limits the adaptability to evolving streams. Classifiers are combined using weighted voting, and individual weights are calculated with a modified Winnow strategy.

An ensemble of online sequential extreme learning machine (ESOS-ELM) was developed by Mirza et al. [129]. It maintains randomized neural networks that are trained on balanced subsets of stream. Short and long term memories were implemented to store the ensemble and the progress of the stream. Two different learning schemes were proposed for moderate and high imbalance ratios (the difference being the way of processing majority class instances). However, the algorithm replicates the limitations of some of the previous methods, assuming no drift on the minority class taking place.

Another approach to imbalanced and drifting streams was proposed by Wang et al. [180]. These authors are the only researchers which currently consider also dynamic changes of class cardinalities. They proposed a number of online bagging-based solutions that are able to cope with dynamically changing imbalance ratio and switching of class properties (e.g. majority becoming minority over time). They considered a dedicated concept drift detector for imbalanced streams, whose output directly influences the oversampling or undersampling ratios, allowing to accommodate evolving data skewness. A further modification, called WEOB, uses a combination of both under and oversampling in order to choose the better strategy for the current state of the stream. An adaptive weighting combination scheme was proposed to accommodate this hybrid solution, where the weights of the sampling strategies are either computed as their G-mean values or are binary (meaning only one of them will be used at a time). A multi-class extension of this method was discussed in [181], where concepts of multi-minority and multi-majority classes are used to model complex relations among classes.

Finally, recently some researchers have started to discuss the need for new evaluation measures to address complexity of imbalanced data streams, see , e.g., [20,30,33].

5.2. Novelty detection and one-class classification

Due to the evolving nature of data streams the learning algorithm has to be prepared to handle new, unseen data that do not

Table 7
Ensembles for novelty detection and one-class classification.

Algorithm	Description
OCLS [198]	One-class learning and summarization ensemble
UOCL [118]	Extended ensemble for one-class learning and summarization
IncOCBagg [102]	Incremental one-class Bagging
OLP [37]	One-class ensemble based on prototypes
Learn^{++}.NC [130]	Learn^{++} ensemble for novel class detection
ECSMiner [122]	Ensemble for novelty detection with time constraints
MCM [121]	Ensemble for novelty detection and drifting feature space
AnyNovel [1]	Two-step clustering ensemble for novelty detection
CBCE [171]	Class-based ensemble for class evolution
CLAM [4]	Class-based micro classifier ensemble
SCARN [4]	Stream Classifier and novel and recurring class detector

follow the previously seen distributions. Such examples may be caused by noise in the stream or may actually originate from a novel concept that started emerging. Such a novelty may be caused by some abnormality (like zero-day-attach in networks or anomaly in the system) or may be a new instance from a concept that was previously not seen. In the latter case a completely new class may appear in the decision space, existing classes may merge or one of the classes may star to disappear. This may happen in the context of two possible scenarios: binary and multi-class. In the former case we may treat it as a task of recognizing a target (correct) concept and a set of potential outliers [115], while in the latter we must deal at the same time with a recognition problem among a number of classes and detection of possible new emerging classes [53]. For the binary case we often must face the fact that it is difficult or even impossible to gather sufficient representatives of the novel class, or that they may not even form a class. Therefore, one-class classification (known as learning in the absence of counterexamples) is being utilized as it allows to model the target concept without making any assumptions regarding the properties of the novelty observations to appear.

Let us discuss now main ensemble-based methods suitable for these scenarios. They are summarized in Table 7.

Zhu et al. [198] proposed an one-class ensemble approach to mining data streams with a concept summarization approach by providing labels not for single instances but for chunks of instances. They introduced a vague one-class learning module, based on one-class Support Vector Machines. Each base classifier utilized weights assigned to instances from given chunk, reflecting their level of relevance (in the discussed application the relevance was based on user's interests in given information). This was done in a two-step procedure, utilizing local and global weighting. Local weighting calculated instance weight values using examples in the given data chunk. Global weighting was used to calculate a weight value for both positive and unlabeled instances in given chunk, utilizing information coming from classifiers trained on previous data chunks. This weight information was directly embedded in the process of classifier training. A weighted classifier combination scheme was used to make a final ensemble decision, where the weights of each classifier were calculated as an agreement measure between it and the most recent classifier in the pool. One must notice that this approach used static one-class classifiers and thus adaptability was achieved only by adding new members to the ensemble.

This idea was further developed by Liu et al. [118]. They also proposed a chunk-based ensemble of one-class classifiers for simultaneous learning from uncertain data streams and concept summarization. They proposed a different scheme for calculating instance weights by using a local kernel-density approach. It allowed to generate a bound score for each example based on its local nearest neighbors in a kernel feature space. Thus, instance weight was calculated only once and directly embedded in the process of one-class Support Vector Machine training. A combination of classifiers was done using a weighted aggregation, where a weight for each base classifier was determined by its mean square error. Similar to the previous work, classifiers used here were static ones.

An ensemble of adaptive one-class classifiers for drifting data streams was proposed by Krawczyk and Woźniak [102]. Here, classifiers were trained with the usage of Bagging. The set-up of the ensemble remains unchanged during the stream processing, but base classifiers are updated with random subsets of examples from incoming data chunks. As a base classifier they used an incremental weighted one-class Support Vector Machine [103]. It incorporates new examples by re-weighting support vectors and adding/removing them according to the stream progress. New instances can be weighted according to two different strategies (highest priority to newest examples or weights based on the distance from the hypersphere center). The forgetting mechanism was implemented as a gradual decrease of weights assigned to vectors, realized as a time-dependent function (the longer time given instance spent in the stream, the higher the forgetting ratio). This approach allowed the method to adapt to concept drift without a need for an external drift detector, as old concepts were gradually removed from the ensemble memory. Additionally, a parallel implementation was proposed in order to achieve a computational speed-up. However, authors focused their works only with chunk-based processing of data streams.

Czarnowski and Jedrzejowicz [37] proposed yet another chunk-based ensemble of one-class classifiers for handling binary and multi-class data streams. Here a single one-class classifiers (decision tree) was responsible for tackling a single class. Each class-based data chunk utilized for training classifiers consisted of class prototypes and information about whether the class predictions of these instances, carried-out at earlier steps, has been correct. When a new chunk of data becomes available, an instance selection algorithm is applied to select the most valuable examples. Classifiers are combined using a weighted voting scheme.

Muhlbaier et al. [130] introduced an extension of Learn^{++} for the cases with novel class appearance in streams. The main change over the previous version of the ensemble is an extension of the classifier combination phase. A dynamically weighted consult and vote was proposed, where individual classifiers interchange their information regarding novel instances and select the most competent ones by assigning them highest weights. This allows to prevent cases when a new classifier trained with a novel class is outvoted by older ones who did not have access to new instances. However, this solution is suitable only to scenarios in which classes emerge in a transient manner.

Masud et al. [122] introduced an ensemble classifier for simultaneous classification and novelty detection in drifting data streams with embedded time constraints. It worked under an assumption that each example must be evaluated within a given time window not to create a bottleneck for rapidly incoming instances. This is of crucial importance to the novelty detection module that is usually characterized by the highest computational complexity in the entire classification system. Additionally, authors took into account the possible delay with which a true class label may become available to the system. These two constraints allowed to create a computationally efficient ensemble for high-speed and evolving data streams. As a base component authors proposed Enhanced Classifier for Data Streams with novel class Miner (ECSMiner), an ensemble system with three buffers: for po-

tentially novel instances, for instances waiting for class labels, and for labeled instances to be used in training new classifiers.

In their follow-up work Masud et al. [121] proposed a new ensemble method that take into account not only concept drift and novel class appearance, but also the possibility of evolving feature space. They assumed that new features may appear over time, which is being justified by specific domain-based applications (e.g., new phrases in text stream mining). Each model in the ensemble was built using feature space homogenization using lossless conversion, to avoid differences between training and testing sets. However, there are several different modifications of their methods in this work. The outlier detection module has been enhanced with an adaptive threshold for changing definitions of novel instances. The novelty detection module was constructed with the usage of Gini coefficient to simultaneously measure the difference among new instance and existing classes, as well as its similarity to other novel instances stored in a buffer. Finally, the proposed classification system allowed for detecting multiple novel classes at the same time using a graph analysis.

Abdallah et al. [1] proposed an adaptive ensemble approach for multi-class novelty detection. The proposed method was based on a two-step cluster formation. Firstly a supervised learning method was applied to divide the initial data into class-based clusters. Then, an unsupervised learning was applied to detect sub-concepts within each cluster and thus to create more local models. Authors showed that their algorithm can efficiently distinguish between actual novel concept appearance, drift present in one of the existing sub-concepts or singular outliers appearance. This was done by defining novel concept as residing outside all existing cluster-based models and consistently moving away from all existing concepts. A forgetting mechanism was implemented to detect concepts that no longer appear in the incoming stream and mark them as irrelevant. To evaluate the model within the stream progress, authors proposed an active learning strategy to reduce labeling costs.

Sun et al. [171] introduced Class-Based ensemble for Class Evolution (CBCE). They considered three possible scenarios: class emergence, disappearance and re-occurrence. CBCE constructs its ensemble by storing in a memory an online classifier for every single class that has appeared during the course of data stream processing. This is done via one-vs-all binary decomposition. Additionally, a dynamic undersampling technique to deal with class imbalance is applied to each base classifier to counter the evolving disproportions between instances in classes. However, CBCE requires its base classifiers to provide predictions in the form of a score, which limits the number of possible models to be used. When a novel class emerges, then its prior probability is being estimated and a new classifier is being trained. Classifiers may be deactivated when a concept disappears and reactivated when its re-occurrence has been detected.

Two other ensemble-based approaches to novel class detection were proposed by Al-Khateeb et al. [4], namely Class Based Micro Classifier Ensemble (CLAM) and Stream Classifier And Novel and Recurring class detector (SCARN). CLAM uses an ensemble of micro-classifiers, where each base micro-classifier has been trained using only positive instances from a given class. This is done via a clustering approach. When a new instance becomes available, the ensemble of micro-classifiers decides whether this is instance belongs to any of existing classes or it is a novel one. After a certain number of instances has been tagged as representatives of a novel concept, a new classifier is trained on them and added to the ensemble. The novelty detection is conducted using a proposed neighborhood-based distance score. SCARN approach uses two ensemble models: primary ensemble and auxiliary ensemble. The primary ensemble is responsible for distinction between known classes and potential outliers. If the outlier has been detected by the primary ensemble, it is then delegated to the auxiliary ensemble. Its role is to decide whether this is a reoccurring concept from previously known class or a completely new case.

Table 8
Active and semi-supervised ensembles.

Algorithm	Description
MV [199]	Optimal Weight Classifier Ensemble with active learning
ReaSC [123]	Ensemble of semi-supervised micro-clusters
ECU [196]	Semi-supervised ensemble integrating classifiers and clusters
COMPOSE [49]	Ensemble for initially labeled data streams
SPASC [78]	Ensemble of semi-supervised clustering algorithms

5.3. Active and semi-supervised learning

Fast availability of information about true target value (class) of incoming examples is another issue which should be taken into account. As mentioned in Section 3 most of used frameworks assume immediate or not too much delayed access to target values. In some situations it is possible to obtain true example state at minimal or no cost. An example would be weather prognosis, where our prediction will be evaluated in future. This is however connected with the problem of label latency - even if we will have access to such an information it does not become available right after the arrival of a new instance. However, in many practical situation this assumption may not be realistic, mainly due to potentially high speed of incoming examples and costs of human labeling. Note that while cooperating with human experts one has to take into account their limited abilities, responsiveness, and threshold on amount of data labeled in a certain amount of time. When all examples cannot be quickly labeled, it may be still possible to obtain true target values for a limited number of these examples at reasonable costs – see a discussion in Section 2.2. This can be exploited with active learning [58] or semi-supervised (including self-labeling) learning [174].

Active learning techniques must take into account the possible drifts in data and adapt their sampling rules to it [205]. One cannot use standard static uncertainty-based methods, as they are not robust to situations where drift occurs in a region of high classifier certainty. In recent years, one could see an increased number of studies dealing with this problem that propose various mechanisms for adaptive active learning over non-stationary streams [23,93,187,190]. Ensemble-inspired approaches have been already applied to select examples in static, non-stream data frameworks. However, existing work on using ensemble-based approaches for active learning in data stream mining is scarce and this direction seems worthwhile for future exploitation. We present the ensemble solutions for active and semi-supervised learning over data streams below. Discussed algorithms are summarized in Table 8.

It is worth mentioning one of the key concepts of active learning called *Query by Committee* [56], where active learning sampling is controlled by an ensemble of classifiers. The most popular methods from this domain include *Query by Bagging* [2] and *Query by Boosting* [2]. They have been proven to offer increased stability and improved instance selection for labeling compared to queries based on a single classifier decision. However, work on using ensemble-based approaches for active learning in data stream mining is scarce and this direction also seems worthwhile for future exploitation.

Zhu et al. [199] proposed to use active learning for controlling the adaptation progress of an ensemble over drifting data streams. Authors argued that variance of an ensemble has a direct relationship with its error rate and thus one should select such instances for labeling that contribute towards the minimization of the variance. Authors used bias-variance decomposition of ensemble error as a basis for their minimum-variance instance selection method.

Additionally, these authors derived an optimal weight calculation scheme for combining components. These two elements work in an active learning loop – weights from the previous iteration are used to guide the active learning procedure, after which a set of labeled examples is used for the weight update step.

Masud et al. [123] proposed an approach where micro-clusters are generated using semi-supervised clustering and a combination of these models are used to handle unlabeled data incoming from the stream. A label propagation technique is used to assign each micro-cluster to a class. Then, inductive label propagation is used to classify a new instance. New micro-clusters can be added to an ensemble with the progress and changes in the stream. Additionally, an ensemble pruning technique is utilized, deleting any micro-cluster with accuracy dropping below the given threshold (70%).

Learning with delayed labels has often been studied with a mechanism to propagate available labels through the next steps when only unlabeled data is available. For instance, Zhang et al. considered a hybrid ensemble integrating classifiers and clusters, where labeled example are used to learn classifiers while clusters are formed from unlabeled data [196]. New incoming instance receives a label resulting from voting both classifiers and clusters. Another interesting statistical approach to represent each class in the stream by a mixture of sub-population was considered by Krempl and Hofer [104]. However this approach is restricted to track only limited gradual drifts in unlabeled data.

COMPOSE (*COMPacted Object Sample Extraction*) ensemble [49] was proposed for streams where labeled instances are available only during the initial training of classifiers. After this phase, all incoming instances are assumed to be non-labeled. COMPOSE works in three steps. First, initial labels are combined with new unlabeled data to train a semi-supervised classifier and use it to label these instances. Then, each class gets assigned a geometric descriptor to construct an enclosing boundary and provide the current distribution of this class. Finally, instances called core supports are extracted to serve as class representatives. This allows to track concept drift in a semi-supervised manner and adapt models accordingly.

Hosseini et al.[78] proposed an ensemble of semi-supervised clustering algorithms, where each class is described by a single model. Each new incoming chunk obtains a pre-defined number of labeled instances, which are used to update classifiers in the ensemble. Chunks are assigned based on a semi-supervised Bayesian approach. Authors claim that their approach is able to automatically recognize recurrent concepts within the data stream.

5.4. Complex data representations and structured outputs

Non-standard data and class structures have gained increasing attention in recent years from the machine learning community. Due to the advent of big data and the necessity to mine unstructured, heterogeneous and complex information, we require learning methods that can efficiently accommodate such instances. Although most of the current research concerns static, non-streaming frameworks, some research has been undertaken in the case of data streams. The most important streaming ensemble solutions are discussed below and are summarized in Table 9.

Multi-label and multi-instance learning is still a largely unexplored area in data stream mining. In case of multi-label algorithm a proper experimental and evaluation framework was proposed by Read et al. [149], but there is not an abundance of work that follow it, especially from the ensemble point of view. Qu et al. [145] proposed a dynamic classifier ensemble for multi-label data streams, where a binary relevance scheme was extended by using feature weighting and keeping a subset of the most recent classifiers in the pool, instead of all possible pairwise combinations. Classifiers are weighted dynamically for each incoming example from the stream.

Table 9
Ensembles for streaming complex data representations.

Algorithm	Description
Multi-label data streams	
DI [145]	Dynamic ensemble with improved binary relevance
MW [193]	Multiple-window ensemble for multi-label streams
MLDE [166]	Multi-voting dynamic ensemble with clustering
FCM-BR [173]	Binary relevance with fuzzy confusion matrix
Multi-instance data streams	
MILTrack [9]	Multi-instance online Boosting
OMILBoost [144]	Online Boosting based on image patches
Semi-WMIL [182]	Semi-supervised ensemble of weak online classifiers
Other data structures	
AdaTreeMiner [15]	XML stream mining using closed tree algorithms
XSC [26]	Ensemble of maximal frequent subtrees for each class
gSLU [140]	Ensemble based framework to partition graph streams
gEboost [139]	Boosting for imbalanced and noisy graph streams

Xioufis et al. [193] introduced an ensemble using a binary relevance model and maintaining two separate windows – one for positive and one for negative examples. An efficient implementation of k-NN classifier is used due to its natural incremental nature, while each base classifier is trained on an undersampled label set to tackle possible label imbalance.

The problems related with labeling costs for multi-label data streams were discussed by Wang et al. [179]. A theoretical loss function for their proposed ensemble classifier and an active learning function to select examples minimizing this function were derived. This allowed for using less labeled instances for training and detecting concept drift on the basis of labeling the most uncertain examples.

Multi-Label Dynamic Ensemble (MLDE) was developed in [166]. It used adaptive cluster-based classifiers that were combined by a voting method utilizing two separate weights based on accuracy on the given data chunk and similarity among chunks.

Trajdos and Kurzynski [173] proposed a stream-based extension of binary relevance model utilizing a fuzzy confusion matrix to correct the decisions of base classifiers in the ensemble. The correction model was updated as the stream progressed, thus adapting to its current state. However, no explicit drift detection technique was used.

Multi-instance learning is an even less exploited area in the stream mining context. Most work in this domain concentrates on image analysis applications and is used in online video processing. However, one may see a video as a stream of images. Babenko et al. [9] proposed a modification of online boosting for learning from bags of examples. They assumed that once a bag is labeled as a positive one, then all examples within are also positive and hence used for training. However, this drawback was reduced by choosing weak classifiers on the basis of a bag likelihood loss function. The ensemble could be updated with new models with the progress of the stream similar to standard online Boosting. A similar approach was proposed by Qi et al. [144], using however a different classifier selection approach based on selecting correct image patch around the labeled target. Wang et al. [182] proposed a semi-supervised ensemble of weak online classifiers for object tracking. The final ensemble was constructed by selecting weak classifiers obtained by maximizing the log-likelihood function but minimizing the inconsistency function.

Mining XML data is well-studied in static scenarios. However, modern computing environments require online and efficient document processing within time and memory constraints. Bifet and Gavaldà [15] proposed compression of XML trees into vectors that are possible for processing by standard classifiers, creating closed frequent pattern data structures. These are later feed into a number of stream classifiers based on variants of Bagging and Boosting

for online analysis. However, the main contribution of the paper lied in new data structures, whereas their ensembles were standard ones from the literature.

Brzezinski and Piernik [26] developed XML Stream Classifier (XSC) ensemble. It creates a set of maximal frequent subtrees for each class independently. Label prediction is done using association between new documents incoming from the stream and the closest maximal frequent subtree (and thus the class to which it is associated). The base classifiers are updated in sequential manner, but as each class has its own classifier the update rates or size of the update chunks may vary. This makes XCS suitable for processing imbalance and locally drifting data streams.

Streams of graphs are also a frequent challenge for learning algorithms, as they become more and more prevalent with the constant growth of social networks. Pan et al. [140] proposed an ensemble approach for mining graph streams, where a stream is partitioned into a number of chunks, each of which contains both labeled and unlabeled graphs. A minimum-redundancy feature selection is applied independently in each chunk to reduce its dimensionality. A sliding window solution with instance weighting is used to accommodate the possibility of drift presence and forget outdated examples. Each chunk serves as a training set to build a classifier, and then form them into an ensemble. Nearly the same authors have recently extended this idea by proposing a Boosting approach called gEboost for imbalanced and noisy graph streams [139]. It maintains the graph partitioning approach (including a special feature selection from subgraphs), but for each chunk a Boosting classifier was constructed and learned with a variant of margin maximization. Instance weighting was incorporated directly into this scheme to put more emphasis on the most difficult examples for the imbalance problem.

6. Future research directions

In this paper, we have discussed the challenging issues of learning ensembles from data streams. We have considered both classification and regression ensembles, even though classifier ensembles are typically the most often applied approaches in data stream analysis.

In the first sections of the paper, we have presented characteristics which distinguish data streams from the standard static data repositories. New requirements to using computationally effective algorithms, which should usually also be able to adapt to concept drift in non-stationary data streams, have been discussed. Different types of concept drift, their characteristics, and methods for their detection in different stream scenarios have been reviewed. Moreover, difficulties in evaluating stream classifiers in presence of concept drift have been shown. The main part of our paper includes a detailed survey of ensembles, which are categorized with respect to different criteria (stationary or not data, chunk or online processing modes, passive or active reactions to drifts). Furthermore, we have extended this study to more complex stream situations such as class-imbalanced learning, novelty detection, active and semi-supervised learning, and dealing with more complex data structures.

Despite many interesting developments in the field of mining data streams, there is still a number of open research problems and challenges awaiting to be properly addressed. We briefly present our views on potential directions that seem worthwhile to be further explored below:

- **Better handling delayed information and extending current techniques within semi-supervised learning**: these approaches are still limited to few ensemble proposals and definitely need more attention. In particular, in fast evolving streams, the relationship between attributes and target values may be only locally valid due to concept drift [105]. Many of the discussed approaches employ a kind of transfer learning, where predictions from models learned from labeled examples are transferred to next unlabeled portions of the data. In general, they are more useful for limited gradual drifts, while more complex scenarios are still open problems. Developing new approaches to deal with delayed information, including ensembles, that would work in the presence of different types of drift is a non-trivial research task. It would be particularly useful for many real life automated systems, where an interaction with human experts is quite limited. Finally, delayed information may not refer to target values only, but may concern also incomplete attribute descriptions. The problem of incomplete data is more intensively studied in static, off-line data mining, where different imputation techniques have been developed. In the streaming context, there is not too much research on such techniques or other approaches which could learn classifiers with omitting such incomplete descriptions and then update the classifier structure.
- **New frameworks for evaluating data stream classifiers**: several interesting issues on evaluating classifiers have been studied for static, off-line data. For a comprehensive overview, we refer the reader to [88]. Although new measures [20,30,63,158] have been recently introduced, the nature of complex evolving data streams still poses requirements for novel theoretical and algorithmic solutions. This is particularly needed for more complex stream scenarios with verification latency, changing class imbalance, censored even data streams [157], multiple data streams [167], and changes of misclassification costs [105]. As researchers have considered many different kinds of measures (e.g. predictive performance, time or memory costs, reaction time and many others), a multi-criteria analysis may be more appropriate than aggregating several measures into a single coefficient [28]. Another open issue is rethinking frameworks for testing stream algorithms. Tuning parameters of streaming ensembles is more difficult than in the static case, where special validation sets or internal cross-validation are usually employed. Their equivalents for evolving streams are yet to be invented. How to access ground truth in unsupervised streams also needs to be elaborated. Finally, statistical analysis of significance of difference between several algorithms with respect to time changes should be developed, similarly to recent recommendations to use appropriate non-parametric tests for static offline setup.
- **Benchmark datasets**: the number of real-world publicly available datasets for testing stream classifiers is still too small. It limits comparative studies of different streaming algorithms. Moreover, some popular data used in the literature is questioned to represent sufficiently real drifts, see e.g. discussions on electricity data [202]. This is a more difficult situation compared to the state of available static datasets such as the UCI Machine Learning Repository.
- **Dedicated diversity measures for data stream classifier ensembles**: recall that ensemble diversity is one of the important characteristics of ensembles in the standard, static data context [24,108,159]. As discussed in Section 1, several researchers studied the relationship between high ensemble predictive performance and the diversity of its components. Others used specialized diversity measures [108] to visually analyzing ensemble classification accuracy. These measures have also been used to tune the combination rule for aggregating component classifier predictions or to prune too large pool of components inside the ensemble. However, such research is not much visible in case of streaming ensembles. On the one hand, one can say that as component classifiers are learnt from different parts of the stream, they are already different and diverse ones. On the

other hand, our literature survey shows that only few authors directly consider promoting diversity while constructing an ensemble or rebuilding them in the moment of detecting drifts, see e.g. DDD ensemble [126] or other generalizations of online bagging such as [16]. However, nearly nobody directly measures the diversity of component classifiers in streams. Rare studies are based on taking into consideration the diversity measures developed for static, off-line solutions. The most recent study [32] provides a wider experimental study of using six of the most popular diversity measures [108], where a few online and chunk-based ensembles were evaluated in several scenarios of drifts. The first observation from these experiments is that diversity of ensembles is rather low. Some diversity measures, e.g., κ inter-agreement measure, change values over the stream with relation to occurring drifts – it is more visible for sudden changes rather than for gradual drifts. So, these results may indicate further research lines on combining selected diversity measures, perhaps also with more typical drift detectors to better monitor changes in the evolving stream and to more precisely identify moments of drifts. This could also lead to new solutions for monitoring changes in unlabeled streams. Nevertheless, more research on new diversity measures specialized for evolving stream should be undertaken.

- **Dealing with multiple streams and more complex representations**: nearly all streaming ensembles have been proposed to processing a single stream only. However, some applications, see e.g. studies on internet messages or censored data in the variant of survival analysis [157], may provide several parallel streams. In such multiple streams, the same data events (objects identified in the data sources) may appear in different time moments in each stream and may have different descriptions. This poses several interesting and new challenges, e.g., how to aggregate the information about the same event available in different streams, how to predict the moment of an event appearing in one of the streams, given knowledge on other streams, and whether to develop a new ensemble dedicated to work over such multiple streams. These aspects should be particularly important in the context of integrating different (also heterogeneous) data repositories in Big Data Analysis [87]. Note that data streams are becoming more and more complex in some new applications, such as social media or electronic health records, which require to deal with many heterogeneous data representations at the same moment. Such mixed representations include both structured, semi-structured and completely unstructured data fields, quite often referring to static images, video sequences, or other signals. To fully comprehend the dynamic and phenomenon of these data sources, we need to find interactions among such complex and varying data. As ensembles naturally integrate diverse models, they seem to be a highly promising solution for this challenge.
- **Considering more complex class distributions in imbalanced streams**: working with class-imbalanced and evolving streams is still in early stages. Among very few existing ensemble proposals, most researchers consider the simplest problem of the imbalanced class ratio, without changes of imbalance ratio [180] over time. Note that in the static data framework, other data difficulty factors such as decomposition of the minority class into rare sub-concepts, overlapping with other classes, and presence of very rare minority cases in the majority class regions are also considered as more influential than the global imbalance between classes. Considering them in drifting scenarios, where sub-concepts or rare cases appear over time and overlapping regions change, is an open research problem. Similar new challenges may refer to studying changing multiple minority classes [181]. Finally, new evaluation measures and more rigorous evaluation procedures are needed for evaluating algorithms in such complex imbalanced streams – see a discussion in [105].
- **More studies on the nature of some drift types**: although a lot of research has been done on adaptating ensembles to different concept drifts, several more detailed characteristics of drifts have not yet been consistently examined in literature. In particular, gradual drifts are more difficult to be detected and tracked than sudden changes or reoccurring concepts. The current drift detectors work better with sudden drifts, while the identification of characteristic moments of developing gradual or incremental drifts in real streams are still not sufficiently developed. Furthermore, a more formal definition of different kinds of gradual drifts should be proposed. The authors of [183] showed that the progress of changes inside gradual drifts may be realized in many different ways and needs more specialized solutions. The work of [127] also considers different types of gradual drifts, besides considering that drifts may occur in a sequence of several abrupt and non-severe drifts. The paper [43] postulates that the idea of the so called limited gradual drift is used rather in an intuitive way in most work. Although the work of [183] has attempted to provide more formal definitions of drift characteristics and introduces a new taxonomy of different types of drift, more research should be undertaken to better understand the nature of some drifts, how they develop in real streams, how to measure drift magnitude (e.g. small, medium or high), and which forms of drift could be better handled by specific categories of ensembles.
- **Considering background knowledge or context while classifying data streams**: some researchers argue for including more additional information than basic descriptions of instances when constructing predictions from streams. One of the options is to add background knowledge into drift adaptation techniques [208]. For instance, taking into account seasonal effects while analyzing the electricity benchmark data set nicely illustrates the usefulness of this postulate [206]. Another possibility is classifying data streams taking context into consideration, i.e., usually Markov chains are used to analyze the data stream when there are inter-dependencies between the successive labels, e.g., medical diagnosis – the state of the patient depends not only on the recent observation but also his/her history is taken into consideration. The same in the case of character recognition, when we know that the text is, e.g., written in English, where we can recognize the current letter on the basis of its characteristic, but also take into consideration what was the previous letter (some combinations are not possible and some of them are almost impossible). There are several studies on classification with context, e.g., [70,148,186].
- **Self-tuning ensembles**: most online and chunk-based approaches use models with parameters being either individually tuned or using some preset values – fixed for the complete analysis process. However, with the changes within the stream the previously set parameters may no longer be the sufficiently good (especially in case of parameter-sensitive methods, like support vector machines or neural networks). Therefore, proposing a new methodology for self-tuning streaming ensemble systems may lead to improved predictive power. Additionally, tuning parameters for single classifiers should take into account that they are components within the ensemble. Thus, more global update methods that can lead to obtain more complementary models seems to be worth exploring.
- **Ensemble pruning**: although many ensembles for data streams apply pruning procedures, they are usually based on prediction performance or time that the model has spent within the ensemble. However, as data stream mining is a complex task, these factors may not be sufficient to capture the full dynamics of changes. More advanced pruning techniques could also ex-

ploit a multiple criteria analysis, including not only current predictive ability, but also computational efficiency of base models, memory usage or other resources, current diversity of the ensemble, available information on class labels, etc. At the same time, these pruning techniques should impose minimal computational overhead. Such compound, yet lightweight approaches, should lead to maintaining better ensemble setup and improve adaptation abilities to various types of changes.

- **Other requirements to processing Big Data and privacy issues**: when dealing with massive data streams, algorithms should be able to handle not only changing data, but also big volumes of instances arriving rapidly. At the same time, an ensemble for such data must still work under strict time and memory constraints. This can be handled in two ways – by proposing algorithms with improved scalability or by using special performance computing environments, like SPARK, Hadoop or GPU clusters. Although some attempts to extend the most often used software, like MOA, have already been undertaken, there is still a need for efficient implementations of existing methods within these specialized frameworks for Big Data, as well as developing new solutions natively for them. Another aspect of analyzing Big Data concerns the requirements for privacy protection, especially in complex systems where streams are a sub-part of a more complex analytical workflow [87]. Here, often not only no information can be leaked outside, but also the teams participating within the analysis may not be willing to directly share their data. It raises the need for data stream ensemble algorithms able to work in such scenarios without the possibility of reverse-engineering the underlying data from their decisions and models.

Acknowledgments

This work was supported by the Polish National Science Center under the grants no. DEC-2013/09/B/ST6/02264 and no. DEC-2013/11/B/ST6/00963. J. Gama acknowledges project MAESTRA - ICT-750 2013-612944.

References

[1] Z.S. Abdallah, M.M. Gaber, B. Srinivasan, S. Krishnaswamy, Anynovel: detection of novel concepts in evolving data streams, Evolving Syst. 7 (2) (2016) 73–93.
[2] N. Abe, H. Mamitsuka, Query learning strategies using boosting and bagging, in: Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24–27, 1998, 1998, pp. 1–9.
[3] D. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1991) 37–66.
[4] T. Al-Khateeb, M.M. Masud, K. Al-Naami, S.E. Seker, A.M. Mustafa, L. Khan, Z. Trabelsi, C.C. Aggarwal, J. Han, Recurring and novel class detection using class-based ensemble for evolving data stream, IEEE Trans. Knowl. Data Eng. 28 (10) (2016) 2752–2764.
[5] C. Alippi, Intelligence for Embedded Systems. A Methodological Approach, Springer International Publishing, 2014.
[6] C. Alippi, G. Boracchi, M. Roveri, Change detection tests using the ICI rule, in: Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN), 2010, pp. 1–7.
[7] C. Alippi, G. Boracchi, M. Roveri, Hierarchical change-detection tests, IEEE Trans. Neural Netw. Learn. Syst. 28 (2) (2017) 246–258.
[8] C. Alippi, M. Roveri, Just-in-time adaptive classifiers. Part i: detecting nonstationary changes, IEEE Trans. Neural Netw. 19 (7) (2008) 1145–1153.
[9] B. Babenko, M. Yang, S.J. Belongie, Robust object tracking with online multiple instance learning, IEEE Trans. Pattern Anal. Mach. Intell. 33 (8) (2011) 1619–1632.
[10] M. Baena-García, J. Del Campo-Ávila, R. Fidalgo, A. Bifet, Early drift detection method, in: Proceedings of the Forth ECML PKDD International Workshop on Knowledge Discovery From Data Streams (IWKDDS'06), 2006, pp. 77–86. Berlin, Germany
[11] A. Bifet, Adaptive Learning and Mining for Data Streams and Frequent Patterns, Universitat Politécnica de Catalunya, 2009 Ph.D. thesis.
[12] A. Bifet, E. Frank, Sentiment knowledge discovery in twitter streaming data, in: Discovery Science - 13th International Conference, DS 2010, Canberra, Australia, October 6–8, 2010. Proceedings, 2010, pp. 1–15.
[13] A. Bifet, E. Frank, G. Holmes, B. Pfahringer, Accurate ensemble for data streams: combining restricted Hoeffding trees with stacking, in: 2nd Asian Conference on Machine Learning (ACML 2010), 2010a, pp. 225–240.
[14] A. Bifet, R. Gavaldà, Learning from time-changing data with adaptive windowing, Proceedings of the 7th SIAM International Conference on Data Mining, 2007.
[15] A. Bifet, R. Gavaldà, Adaptive XML tree classification on evolving data streams, in: Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2009, Bled, Slovenia, September 7–11, 2009, Proceedings, Part I, 2009, pp. 147–162.
[16] A. Bifet, G. Holmes, B. Pfahringer, Leveraging bagging for evolving data streams, in: ECML/PKDD (1), 2010b, pp. 135–150.
[17] A. Bifet, G. Holmes, B. Pfahringer, R. Gavaldà, Improving adaptive bagging methods for evolving data streams, in: Proceedings of the 1st Asian Conference on Machine Learning, in: Lecture Notes in Computer Science, 5828, Springer, 2009a, pp. 23–47.
[18] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, R. Gavaldà, New ensemble methods for evolving data streams, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery andData Mining, ACM Press, New York, NY, USA, 2009b, pp. 139–148.
[19] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, T. Seidl, Moa: massive online analysis, J. Mach.Learn. Res. (JMLR) (2010c) 1601–1604.
[20] A. Bifet, G.D.F. Morales, J. Read, G. Holmes, B. Pfahringer, Efficient online evaluation of big data stream classifiers, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10–13, 2015, 2015, pp. 59–68.
[21] A. Bifet, J. Read, B. Pfahringer, G. Holmes, I. Zliobaite, CD-MOA: change detection framework for massive online analysis, in: Advances in Intelligent Data Analysis XII - 12th International Symposium, IDA 2013, London, UK, October 17–19, 2013. Proceedings, 2013, pp. 92–103.
[22] I.I.F. Blanco, J. del Campo-Avila, G. Ramos-Jimenez, R.M. Bueno, A.A.O. Diaz, Y.C. Mota, Online and non-parametric drift detection methods based on Hoeffding's bounds, IEEE Trans. Knowl. Data Eng. 27 (3) (2015) 810–823, doi:10.1109/TKDE.2014.2345382.
[23] M. Bouguelia, Y. Belaïd, A. Belaïd, An adaptive streaming active learning strategy based on instance weighting, Pattern Recognit. Lett. 70 (2016) 38–44.
[24] G. Brown, J.L. Wyatt, R. Harris, X. Yao, Diversity creation methods: a survey and categorisation, Inf. Fusion 6 (1) (2005) 5–20.
[25] D. Brzezinski, Block-Based and Online Ensembles for Concept-Drifting Data Streams, Poznan University of Technology, 2015 Ph.D. thesis.
[26] D. Brzezinski, M. Piernik, Structural XML classification in concept drifting data streams, New Gener. Comput. 33 (4) (2015) 345–366.
[27] D. Brzezinski, J. Stefanowski, Accuracy updated ensemble for data streams with concept drift, in: Proceedings of the 6th HAIS International Conference on Hybrid Artificial IntelligentSystems, Part II, in: LNCS, 6679, Springer, 2011, pp. 155–163.
[28] D. Brzezinski, J. Stefanowski, Classifiers for concept-drifting data streams: evaluating things that really matter, in: ECML PKDD 2013 Workshop on Real-World Challenges for Data Stream Mining, September 27th, Prague, Czech Republic, 2013, pp. 10–14.
[29] D. Brzezinski, J. Stefanowski, Combining block-based and online methods in learning ensembles from concept drifting data streams, Inf. Sci. 265 (2014a) 50–67.
[30] D. Brzezinski, J. Stefanowski, Prequential AUC for classifier evaluation and drift detection in evolving data streams, in: New Frontiers in Mining Complex Patterns - Third International Workshop, NFMCP 2014, Held in Conjunction with ECML-PKDD 2014, Nancy, France, September 19, 2014, Revised Selected Papers, 2014b, pp. 87–101.
[31] D. Brzezinski, J. Stefanowski, Reacting to different types of concept drift: the accuracy updated ensemble algorithm, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2014c) 81–94, doi:10.1109/TNNLS.2013.2251352.
[32] D. Brzezinski, J. Stefanowski, Ensemble diversity in evolving data streams, in: Discovery Science: 19th International Conference, DS 2016. Proceedings, in: LNCS, 9956, Springer, 2016, pp. 229–244.
[33] D. Brzezinski, J. Stefanowski, Prequential auc: properties of the area under the roc curve for data streams with concept drift, Knowl. Inf. Syst. (2017).
[34] S. Chen, H. He, SERA: selectively recursive approach towards nonstationary imbalanced stream data mining, in: International Joint Conference on Neural Networks, IJCNN 2009, Atlanta, Georgia, USA, 14–19 June 2009, 2009, pp. 522–529.
[35] S. Chen, H. He, Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach, Evolving Syst. 2 (1) (2011) 35–50.
[36] F. Chu, C. Zaniolo, Fast and light boosting for adaptive mining of data streams, in: Proceedings of the Eight Pacific-Asia Knowledge Discovery and Data Mining Conference (PAKDD'04), 2004, pp. 282–292. Sydney
[37] I. Czarnowski, P. Jedrzejowicz, Ensemble online classifier based on the one–class base classifiers for mining data streams, Cybern. Syst. 46 (1–2) (2015) 51–68.
[38] M. Deckert, Batch weighted ensemble for mining data streams with concept drift, in: International Symposium on Methodologies for Intelligent Systems, Springer Berlin Heidelberg, 2011, pp. 290–299.
[39] M. Deckert, J. Stefanowski, Comparing block ensembles for data streams with concept drift, in: New Trends in Databases and Information Systems, Springer Berlin Heidelberg, 2013, pp. 69–78.

[40] S. Delany, P. Cunningham, A. Tsymbal, L. Coyle, A case-based technique for tracking concept drift in spam filtering, Knowl. Based Syst. 18 (4–5) (2005) 187–195.

[41] M. Denil, D. Matheson, N. de Freitas, Consistency of online random forests, in: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16–21 June 2013, 2013, pp. 1256–1264.

[42] G. Ditzler, R. Polikar, Incremental learning of concept drift from streaming imbalanced data, IEEE Trans. Knowl. Data Eng. 25 (10) (2013) 2283–2301.

[43] G. Ditzler, M. Roveri, C. Alippi, R. Polikar, Learning in nonstationary environments: a survey, IEEE Comput. Intell. Mag. 10 (4) (2015) 12–25, doi:10.1109/MCI.2015.2471196.

[44] P. Domingos, G. Hulten, Mining high-speed data streams, in: I. Parsa, R. Ramakrishnan, S. Stolfo (Eds.), Proceedings of the ACM Sixth International Conference on Knowledge Discovery and Data Mining, ACM Press, Boston, USA, 2000, pp. 71–80.

[45] A. Dries, U. Rückert, Adaptive concept drift detection, Stat. Anal. Data Min. 2 (56) (2009) 311–327, doi:10.1002/sam.v2:5/6.

[46] L. Du, Q. Song, L. Zhu, X. Zhu, A selective detector ensemble for concept drift detection, Comput. J. (2014), doi:10.1093/comjnl/bxu050.

[47] J. Duarte, J. Gama, A. Bifet, Adaptive model rules from high-speed data streams, TKDD 10 (3) (2016) 30, doi:10.1145/2829955.

[48] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2., Wiley, New York, 2001.

[49] K.B. Dyer, R. Capo, R. Polikar, COMPOSE: a semisupervised learning framework for initially labeled nonstationary streaming data, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2014) 12–26.

[50] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Trans. Neural Netw. 22 (10) (2011) 1517–1531.

[51] W. Fan, F. Chu, H. Wang, P.S. Yu, Pruning and dynamic scheduling of cost-sensitive ensembles, in: Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI'02), American Association for Artificial Intelligence, Menlo Park, USA, 2002, pp. 146–151.

[52] W. Fan, Y. an Huang, H. Wang, P.S. Yu, Active mining of data streams, Proceedings of the 4th SIAM International Conference on Data Mining, 2004.

[53] E.R. de Faria, I.R. Goncalves, J. Gama, A.C.P. de Leon Ferreira de Carvalho, Evaluation of multiclass novelty detection algorithms for data streams, IEEE Trans. Knowl. Data Eng. 27 (11) (2015) 2961–2973.

[54] T. Fawcett, An introduction to {ROC} analysis, Pattern Recognit. Lett. 27 (8) (2006) 861–874. http://dx.doi.org/10.1016/j.patrec.2005.10.010. {ROC} Analysis in Pattern Recognition

[55] A. Fern, R. Givan, Online ensemble learning: an empirical study, Mach. Learn. 53 (1–2) (2003) 71–109.

[56] Y. Freund, H.S. Seung, E. Shamir, N. Tishby, Selective sampling using the query by committee algorithm, Mach. Learn. 28 (2–3) (1997) 133–168.

[57] J. Friedman, L. Rafsky, Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests, Ann. Stat. (1979) 697–717.

[58] Y. Fu, X. Zhu, B. Li, A survey on instance selection for active learning, Knowl. Inf. Syst. 35 (2) (2013) 249–283.

[59] M.M. Gaber, A. Zaslavsky, S. Krishnaswamy, A Survey of Classification Methods in Data Streams, Springer U, pp. 39–59. doi:10.1007/978-0-387-47534-9_3

[60] J. Gama, Knowledge Discovery from Data Streams, Chapman & Hall, CRC Press, 2010.

[61] J. Gama, P. Medas, Learning decision trees from dynamic data streams, J. UCS 11 (8) (2005) 1353–1366.

[62] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Proceedings of the Seventh Brazilian Symposium on Artificial Intelligence (SBIA'04) - Lecture Notes in Computer Science, 3171, Springer, São Luiz do Maranhão, Brazil, 2004, pp. 286–295.

[63] J. Gama, R. Sebastião, P.P. Rodrigues, On evaluating stream learning algorithms, Mach. Learn. 90 (3) (2013) 317–346.

[64] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (4) (2014) 44:1–44:37, doi:10.1145/2523813.

[65] J. Gao, B. Ding, W. Fan, J. Han, P.S. Yu, Classifying data streams with skewed class distributions and concept drifts, IEEE Internet Comput. 12 (6) (2008) 37–49.

[66] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (10) (2010) 2044–2064.

[67] A. Ghazikhani, R. Monsefi, H.S. Yazdi, Ensemble of online neural networks for non-stationary and imbalanced data streams, Neurocomputing 122 (2013) 535–544.

[68] R. Greiner, A.J. Grove, D. Roth, Learning cost-sensitive active classifiers, Artif. Intell. 139 (2) (2002) 137–174.

[69] F. Gustafsson, Adaptive Filtering and Change Detection, Wiley, 2000.

[70] R.M. Haralick, Decision making in context, IEEE Trans. Pattern Anal. Mach. Intell. 5 (4) (1983) 417–428, doi:10.1109/TPAMI.1983.4767411.

[71] M. Hassani, Y. Kim, S. Choi, T. Seidl, Subspace clustering of data streams: new algorithms and effective evaluation measures, J. Intell. Inf. Syst. 45 (3) (2015) 319–335.

[72] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284.

[73] H. He, Y. Ma (Eds.), Imbalanced Learning: Foundations, Algorithms, and Applications, Wiley-IEEE Press, 2013.

[74] T.K. Ho, Complexity of classification problems and comparative advantages of combined classifiers, in: Proceedings of the First International Workshop on Multiple Classifier Systems, in: MCS '00, Springer-Verlag, London, UK, UK, 2000, pp. 97–106.

[75] T.K. Ho, J.J. Hull, S.N. Srihari, Decision combination in multiple classifier systems, IEEE Trans. Pattern Anal. Mach. Intell. 16 (1) (1994) 66–75.

[76] Hoens, N. Chawla, Learning in non-stationary environments with class imbalance, in: Proceedings of the 18th ACM SIGKDD Conference onKnowledge Discovery and Data Mining, 2012, pp. 168–176.

[77] T.R. Hoens, R. Polikar, N.V. Chawla, Learning from streaming data with concept drift and imbalance: an overview, Prog. AI 1 (1) (2012) 89–101.

[78] M.J. Hosseini, A. Gholipour, H. Beigy, An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams, Knowl. Inf. Syst. 46 (3) (2016) 567–597.

[79] H. Hotelling, The generalization of student's ratio, Ann. Math. Stat. 2 (3) (1931) 360–378.

[80] E. Ikonomovska, J. Gama, S. Dzeroski, Learning model trees from evolving data streams, Data Min. Knowl. Discov. 23 (1) (2011a) 128–168, doi:10.1007/s10618-010-0201-y.

[81] E. Ikonomovska, J. Gama, S. Dzeroski, Online tree-based ensembles and option trees for regression on evolving data streams, Neurocomputing 150 (2015) 458–470, doi:10.1016/j.neucom.2014.04.076.

[82] E. Ikonomovska, J. Gama, B. Zenko, S. Dzeroski, Speeding-up Hoeffding-based regression trees with options, in: L. Getoor, T. Scheffer (Eds.), Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28, - July 2, 2011, Omnipress, 2011b, pp. 537–544.

[83] G. Jaber, An Approach for Online Learning in the Presence of Concept Changes, AgroParisTech University, 2001 Ph.D. thesis.

[84] K. Jackowski, Fixed-size ensemble classifier system evolutionarily adapted to a recurring context with an unlimited pool of classifiers, Pattern Anal. Appl. 17 (4) (2014) 709–724.

[85] K. Jackowski, Adaptive splitting and selection algorithm for regression, New Gener. Comput. 33 (4) (2015) 425–448.

[86] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, Neural Comput. 3 (1991) 79–87.

[87] N. Japkowicz, S. Jerzy (Eds.), Big Data Analysis: New Algorithms for a New Society, Springer, 2016.

[88] N. Japkowicz, M. Shah, Evaluating Learning Algorithms: A Classification Perspective, Cambridge University Press, 2011.

[89] P.M.G. Jr., S.G. de Carvalho Santos, R.S. Barros, D.C. Vieira, A comparative study on concept drift detectors, Expert Syst. Appl. 41 (18) (2014) 8144–8156. http://dx.doi.org/10.1016/j.eswa.2014.07.019.

[90] P. Kadlec, B. Gabrys, Local learning-based adaptive soft sensor for catalyst activation prediction, AIChE J. 57 (5) (2011) 1288–1301.

[91] I. Katakis, G. Tsumakas, I. Vlahavas, An ensemble of classifier for coping with recurring contexts in data streams, in: Frontiers in Artificial Intelligence and Applications (ECAI 2008), 2008, pp. 763–764.

[92] T. Kidera, S. Ozawa, S. Abe, An incremental learning algorithm of ensemble classifier systems, in: Proceedings of the International Joint Conference on Neural Networks, IJCNN 2006, Part of the IEEE World Congress on Computational Intelligence, WCCI 2006, Vancouver, BC, Canada, 16–21 July 2006, 2006, pp. 3421–3427.

[93] H. Kim, S. Madhvanath, T. Sun, Hybrid active learning for non-stationary streaming data with asynchronous labeling, in: 2015 IEEE International Conference on Big Data, Big Data 2015, Santa Clara, CA, USA, October 29, - November 1, 2015, 2015, pp. 287–292.

[94] R. Klinkenberg, Learning drifting concepts: example selection vs. example weighting, Intell. Data Anal. (IDA) J. - Spec.Issue Incremental Learn. Syst. Capable Dealing Concept Drift 8 (3) (2004) 281–300.

[95] R. Klinkenberg, T. Joachims, Detecting concept drift with support vector machines, in: Proceedings of the Seventeenth International Conference on Machine Learning (ICML'00), Morgan Kaufmann Publishers, San Francisco, CA, 2000, pp. 487–494.

[96] R. Klinkenberg, S. Ruping, Concept Drift and the Importance of Examples, Physica-Verlag, pp. 55–77.

[97] M. Kmieciak, J. Stefanowski, Handling sudden concept drift in Enron message data streams, Control Cybern. 40 (3) (2011) 667–695.

[98] J. Kolter, M. Maloof, Dynamic weighted majority: a new ensemble method for tracking concept drift, in: Data Mining, 2003. ICDM 2003. Third IEEE International Conference on, 2003, pp. 123–130.

[99] J.Z. Kolter, M.A. Maloof, Using additive expert ensembles to cope with concept drift, in: Proceedings of the Twenty Second ACM International Conference on Machine Learning (ICML'05), 2005, pp. 449–456. Bonn, Germany

[100] J.Z. Kolter, M.A. Maloof, Dynamic weighted majority: an ensemble method for drifting concepts, J. Mach. Learn. Res. 8 (2007) 2755–2790.

[101] B. Krawczyk, Learning from imbalanced data: open challenges and future directions, Prog. Artif. Intell. 5 (4) (2016) 221–232.

[102] B. Krawczyk, M. Woźniak, Incremental one-class bagging for streaming and evolving big data, in: 2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, August 20–22, 2015, Volume 2, 2015a, pp. 193–198.

[103] B. Krawczyk, M. Woźniak, One-class classifiers with incremental learning and forgetting for data streams with concept drift, Soft Comput. 19 (12) (2015b) 3387–3400.

[104] G. Krempl, V. Hofer, Classification in presence of drift and latency, in: Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on, Vancouver, BC, Canada, December 11, 2011, 2011, pp. 596–603.

[105] G. Krempl, I. Zliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, J. Stefanowski, Open challenges for data stream mining research, SIGKDD Explor. Newsl. 16 (1) (2014) 1–10, doi:10.1145/2674026.2674028.

[106] A. Krogh, J. Vedelsby, Neural network ensembles, cross validation, and active learning, in: Advances in Neural Information Processing Systems, 7, 1995, pp. 231–238.

[107] L.I. Kuncheva, Classifier ensembles for changing environments, in: Proceedings of the 5th MCS International Workshop on Multiple ClassifierSystems, in: Lecture Notes in Computer Science, 3077, Springer, 2004a, pp. 1–15.

[108] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley-Interscience, Hoboken, NJ, 2004b.

[109] L.I. Kuncheva, Classifier ensembles for detecting concept change in streaming data: overview and perspectives, in: Proceedings of the 2nd Workshop SUEMA 2008 (ECAI 2008), 2008, pp. 5–10.

[110] B. Kurlej, M. Woźniak, Active learning approach to concept drift problem, Logic J. IGPL 20 (3) (2012) 550–559.

[111] B. Lakshminarayanan, D.M. Roy, Y.W. Teh, Mondrian forests: efficient online random forests, in: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8–13 2014, Montreal, Quebec, Canada, 2014, pp. 3140–3148.

[112] Y. Lan, Y.C. Soh, G. Huang, Ensemble of online sequential extreme learning machine, Neurocomputing 72 (13–15) (2009) 3391–3395.

[113] H.K.H. Lee, M.A. Clyde, Lossless online Bayesian bagging, J. Mach. Learn. Res. 5 (2004) 143–151.

[114] V. Lemaire, C. Salperwyck, A. Bondu, A survey on supervised classification on data streams, in: 4th European Summer School on Business Intelligence eBISS 2014, in: Lecture Notes Business Information Processing, 205, Springer, 2014, pp. 88–125.

[115] C. Li, Y. Zhang, X. Li, Ocvfdt: one-class very fast decision tree for one–class classification of data streams, in: Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data, Paris, France, June 28, 2009, 2009, pp. 79–86.

[116] R. Lichtenwalter, N.V. Chawla, Adaptive methods for classification in arbitrarily imbalanced and drifting data streams, in: New Frontiers in Applied Data Mining, PAKDD 2009 International Workshops, Bangkok, Thailand, April 27–30, 2009. Revised Selected Papers, 2009, pp. 53–75.

[117] N. Littlestone, M.K. Warmuth, The weighted majority algorithm, Inf. Comput. 108 (1994) 212–261.

[118] B. Liu, Y. Xiao, P.S. Yu, L. Cao, Y. Zhang, Z. Hao, Uncertain one-class learning and concept summarization learning on uncertain data streams, IEEE Trans. Knowl. Data Eng. 26 (2) (2014) 468–484.

[119] B.I.F. Maciel, S.G.T.C. Santos, R.S.M. Barros, A lightweight concept drift detection ensemble, in: Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on, 2015, pp. 1061–1068, doi:10.1109/ICTAI.2015.151.

[120] M. Markou, S. Singh, Novelty detection: a review–part 1: statistical approaches, Signal Process. 83 (12) (2003) 2481–2497.

[121] M.M. Masud, Q. Chen, L. Khan, C.C. Aggarwal, J. Gao, J. Han, A.N. Srivastava, N.C. Oza, Classification and adaptive novel class detection of feature-evolving data streams, IEEE Trans. Knowl. Data Eng. 25 (7) (2013) 1484–1497.

[122] M.M. Masud, J. Gao, L. Khan, J. Han, B.M. Thuraisingham, Classification and novel class detection in concept-drifting data streams under time constraints, IEEE Trans. Knowl. Data Eng. 23 (6) (2011) 859–874.

[123] M.M. Masud, C. Woolam, J. Gao, L. Khan, J. Han, K.W. Hamlen, N.C. Oza, Facing the reality of data stream classification: coping with scarcity of labeled data, Knowl. Inf. Syst. 33 (1) (2012) 213–244.

[124] F.L. Minku, H. Inoue, X. Yao, Negative correlation in incremental learning, Nat. Comput. 8 (2) (2009) 289–320.

[125] L. Minku, X. Yao, Can cross-company data improve performance in software effort estimation? in: PROMISE, 2012, pp. 69–78.

[126] L.L. Minku, A.P. White, X. Yao, The impact of diversity on online ensemble learning in the presence of concept drift, IEEE Trans. Knowl. Data Eng. 22 (5) (2010) 730–742.

[127] L.L. Minku, X. Yao, DDD: a new ensemble approach for dealing with concept drift, IEEE Trans. Knowl. Data Eng. (TKDE) (2010). 16p. (accepted)

[128] L.L. Minku, X. Yao, How to make best use of cross-company data in software effort estimation? in: ICSE, 2014, pp. 446–456.

[129] B. Mirza, Z. Lin, N. Liu, Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift, Neurocomputing 149 (2015) 316–329.

[130] M.D. Muhlbaier, A. Topalis, R. Polikar, Learn^{++}.nc: combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes, IEEE Trans. Neural Netw. 20 (1) (2009) 152–168.

[131] H.-L. Nguyen, Y.-K. Woon, W.-K. Ng, A survey on data stream clustering and classification, Knowl. Inf. Syst. 45 (3) (2015) 535–569, doi:10.1007/s10115-014-0808-1.

[132] K. Nishida, Learning and Detecting Concept Drift, Hokkaido University, Japan, 2008 Ph.D. thesis.

[133] K. Nishida, K. Yamauchi, Adaptive classifiers-ensemble system for tracking concept drift, in: Proceedings of the Sixth International Conference on Machine Learning and Cybernetics (ICMLC'07), 2007a, pp. 3607–3612. Honk Kong

[134] K. Nishida, K. Yamauchi, Detecting concept drift using statistical testing, in: Proceedings of the Tenth International Conference on Discovery Science (DS'07) - Lecture Notes in Artificial Intelligence, 3316, 2007b, pp. 264–269. Sendai, Japan

[135] A. Osojnik, P. Panov, S. Dzeroski, Comparison of tree-based methods for multi-target regression on data streams, in: New Frontiers in Mining Complex Patterns - 4th International Workshop, NFMCP 2015, Held in Conjunction with ECML-PKDD 2015, Porto, Portugal, September 7, 2015, Revised Selected Papers, 2015, pp. 17–31.

[136] N.C. Oza, Online ensemble learning, in: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, in: AAAI/IAAI, AAAI Press/The MIT Press, Austin, Texas, USA, 2000, p. 1109.

[137] N.C. Oza, S. Russell, Online bagging and boosting, in: Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics (AISTATS'01), Morgan Kaufmann, Key West, USA, 2001, p. 105112.

[138] E.S. Page, Continuous inspection schemes, Biometrika 41 (1/2) (1954) 100–115, doi:10.2307/2333009.

[139] S. Pan, J. Wu, X. Zhu, C. Zhang, Graph ensemble boosting for imbalanced noisy graph stream classification, IEEE Trans. Cybern. 45 (5) (2015) 940–954.

[140] S. Pan, X. Zhu, C. Zhang, P.S. Yu, Graph stream classification using labeled and unlabeled graphs, in: 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8–12, 2013, 2013, pp. 398–409.

[141] A. Pesaranghader, H.L. Viktor, Fast Hoeffding drift detection method for evolving data streams, in: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19–23, 2016, Proceedings, Part II, 2016, pp. 96–111.

[142] B. Pfahringer, G. Holmes, R. Kirkby, New options for Hoeffding trees, in: Proceedings of the 20th Australian Joint Conference on Artificial Intelligence, in: Lecture Notes in Computer Science, 4830, Springer, 2007, pp. 90–99.

[143] R. Polikar, L. Upda, S.S. Upda, V. Honavar, Learn++: an incremental learning algorithm for supervised neural networks, IEEE Trans. Syst. Man Cybern. Part C 31 (4) (2001) 497–508.

[144] Z. Qi, Y. Xu, L. Wang, Y. Song, Online multiple instance boosting for object detection, Neurocomputing 74 (10) (2011) 1769–1775.

[145] W. Qu, Y. Zhang, J. Zhu, Q. Qiu, Mining multi-label concept-drifting data streams using dynamic classifier ensemble, in: Advances in Machine Learning, First Asian Conference on Machine Learning, ACML 2009, Nanjing, China, November 2–4, 2009. Proceedings, 2009, pp. 308–321.

[146] S. Ramamurthy, R. Bhatnagar, Tracking recurrent concept drift in streaming data using ensemble classifiers, in: Proceedings of the Sixth International Conference on Machine Learning and Applications (ICMLA'07), 2007, pp. 404–409. Cincinnati, Ohio

[147] S. Raudys, Statistical and Neural Classifiers: An Integrated Approach to Design, Springer Publishing Company, Incorporated, 2014.

[148] J. Raviv, Decision making in Markov chains applied to the problem of pattern recognition, IEEE Trans. Inf. Theor. 13 (4) (2006) 536–551, doi:10.1109/TIT.1967.1054060.

[149] J. Read, A. Bifet, G. Holmes, B. Pfahringer, Scalable and efficient multi-label classification for evolving data streams, Mach. Learn. 88 (1–2) (2012) 243–272.

[150] J.J. Rodríguez, L.I. Kuncheva, Combining online classification approaches for changing environments, in: Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, in: SSPR & SPR '08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 520–529.

[151] F. Roli, G. Giacinto, Design of Multiple Classifier Systems, World Scientific Publishing.

[152] G. Ross, N. Adams, D. Tasoulis, D. Hand, Exponentially weighted moving average charts for detecting concept drifts, Pattern Recognit. Lett. 33 (2) (2012) 191–198.

[153] A. Saffari, C. Leistner, J. Santner, M. Godec, H. Bischof, On-line random forests, in: Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, IEEE, 2009, pp. 1393–1400.

[154] M. Scholz, R. Klinkenberg, An ensemble classifier for drifting concepts, in: Proceedings of the Second International Workshop on Knowledge Discovery from Data Streams(IWKDDS'05), 2005, pp. 53–64. Porto, Portugal

[155] M. Scholz, R. Klinkenberg, Boosting classifiers for drifting concepts, Intell. Data Anal. (IDA) - Spec. Issue Knowl. Discov. Data Streams 11 (1) (2007) 3–28.

[156] R. Sebastiao, J. Gama, A study on change detection methods, in: Progress in Artificial Intelligence, 14th Portuguese Conference on Artificial Intelligence, EPIA, 2009, pp. 12–15.

[157] A. Shaker, E. Hüllermeier, Survival analysis on data streams: analyzing temporal events in dynamically changing environments, Int. J. Appl. Math. Comput. Sci. 24 (1) (2014) 199–212.

[158] A. Shaker, E. Hüllermeier, Recovery analysis for adaptive learning from non-stationary data streams: experimental design and case study, Neurocomputing 150 (2015) 250–264.

[159] A.J.C. Sharkey, N.E. Sharkey, Combining diverse neural nets, Knowl. Eng. Rev. 12 (3) (1997) 231–247.

[160] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedure. 5th Ed., 5th ed., Boca Raton, FL: CRC Press, 2011.

[161] S.G. Soares, R. Araújo, A dynamic and on-line ensemble regression for changing environments, Expert Syst. Appl. 42 (6) (2015a) 2935–2948.

[162] S.G. Soares, R. Araújo, An on-line weighted ensemble of regressor models to handle concept drifts, Eng. Appl. AI 37 (2015b) 392–406.

[163] P. Sobolewski, M. Woźniak, Comparable study of statistical tests for virtual concept drift detection, in: R. Burduk, K. Jackowski, M. Kurzynski, M. Woźniak, A. Zolnierek (Eds.), Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013, Advances in Intelligent Systems and Computing, 226, Springer International Publishing, 2013a, pp. 329–337.
[164] P. Sobolewski, M. Woźniak, Concept drift detection and model selection with simulated recurrence and ensembles of statistical detectors, J. Universal Comput. Sci. 19 (4) (2013b) 462–483.
[165] P. Sobolewski, M. Woźniak, Scr: simulated concept recurrence a non-supervised tool for dealing with shifting concept, Expert Syst. (2013c) n/a–n/a, doi:10.1111/exsy.12059. EXSY-Oct-12-210.R1
[166] G. Song, Y. Ye, A new ensemble method for multi-label data stream classification in non-stationary environment, in: 2014 International Joint Conference on Neural Networks, IJCNN 2014, Beijing, China, July 6–11, 2014, 2014, pp. 1776–1783.
[167] M. Spiliopoulou, G. Krempl, Tutorial on mining multiple threads of streaming data, in: The Pacific-Asia Conference of Knowledge Discovery and Data Mining (PAKDD 2013), 2013.
[168] K.O. Stanley, Learning Concept Drift With a Committee of Decision Trees, Technical Report, Department of Computer Sciences, University of Texas at Austin, Austin, USA, 2003. UT-AI-TR-03-302
[169] J. Stefanowski, Adaptive ensembles for evolving data streams - combining block-based and online solutions, in: New Frontiers in Mining Complex Patterns - 4th International Workshop, NFMCP 2015, Held in Conjunction with ECML-PKDD 2015, Porto, Portugal, September 7, 2015, Revised Selected Papers, 2015, pp. 3–16.
[170] W. Street, Y. Kim, A streaming ensemble algorithm (SEA) for large-scale classification, in: Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining (KDD'01), ACM Press, New York, 2001, pp. 377–382.
[171] Y. Sun, K. Tang, L.L. Minku, S. Wang, X. Yao, Online ensemble learning of data streams with gradually evolved classes, IEEE Trans. Knowl. Data Eng. 28 (6) (2016) 1532–1545.
[172] P.B. andLuis Torgo, R. Ribeiro, A survey of predictive modeling under imbalanced distributions, ACM Comput. Surv. 49 (2) (2016) 31:1–31:50.
[173] P. Trajdos, M. Kurzynski, Multi-label stream classification using extended binary relevance model, in: 2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, August 20–22, 2015, Volume 2, 2015, pp. 205–210.
[174] I. Triguero, S. García, F. Herrera, Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study, Knowl. Inf. Syst. 42 (2) (2015) 245–284.
[175] A. Tsymbal, The Problem of Concept Drift: Definitions and Related Work, Technical Report, Trinity College Dublin, Ireland, 2004. TCD-CS-2004-15
[176] K. Tumer, J. Ghosh, Analysis of decision boundaries in linearly combined neural classifiers, Pattern Recognit. 29 (2) (1996) 341–348.
[177] A. Wald, Sequential Analysis, John Wiley and Sons, 1947.
[178] H. Wang, W. Fan, P.S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: Proceedings of the Ninth ACM International Conference on Knowledge Discovery and Data Mining (KDD'03), ACM Press, New York, 2003, pp. 226–235.
[179] P. Wang, P. Zhang, L. Guo, Mining multi-label data streams using ensemble-based active learning, in: Proceedings of the Twelfth SIAM International Conference on Data Mining, Anaheim, California, USA, April 26–28, 2012., 2012, pp. 1131–1140.
[180] S. Wang, L.L. Minku, X. Yao, Resampling-based ensemble methods for online class imbalance learning, IEEE Trans. Knowl. Data Eng. 27 (5) (2015) 1356–1368.
[181] S. Wang, L.L. Minku, X. Yao, Dealing with multiple classes in online class imbalance learning, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016, 2016a, pp. 2118–2124.
[182] Z. Wang, S. Yoon, S.J. Xie, Y. Lu, D.S. Park, Visual tracking with semi-supervised online weighted multiple instance learning, Visual Comput. 32 (3) (2016b) 307–320.
[183] G. Webb, R. Hyde, H. Cao, H.L. Nguyen, F. Petitjean, Characterizing concept drift, Data Min. Knowl. Discov. J. 30 (2016) 964–994.
[184] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden context, Mach. Learn. 23 (1996) 69–101.
[185] D.H. Wolpert, The supervised learning no-free-lunch theorems, in: Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications, 2001, pp. 25–42.
[186] M. Woźniak, B. Cyganek, A First Attempt on Online Data Stream Classifier Using Context, Springer International Publishing, Cham, pp. 497–504. doi:10.1007/978-3-319-40973-3_50
[187] M. Woźniak, B. Cyganek, A. Kasprzak, P. Ksieniewicz, K. Walkowiak, Active learning classifier for streaming data, in: Hybrid Artificial Intelligent Systems - 11th International Conference, HAIS 2016, Seville, Spain, April 18–20, 2016, Proceedings, 2016a, pp. 186–197.
[188] M. Woźniak, M. Graña, E. Corchado, A survey of multiple classifier systems as hybrid systems, Inf. Fusion 16 (2014) 3–17.
[189] M. Woźniak, A. Kasprzak, P. Cal, Application of combined classifiers to data stream classification, in: Proceedings of the 10th International Conference on Flexible Query Answering Systems FQAS 2013, in: LNCS, Springer-Verlag, Berlin, Heidelberg, 2013, p. inpress.
[190] M. Woźniak, P. Ksieniewicz, B. Cyganek, A. Kasprzak, K. Walkowiak, Active learning classification of drifted streaming data, in: International Conference on Computational Science 2016, ICCS 2016, 6–8 June 2016, San Diego, California, USA, 2016b, pp. 1724–1733.
[191] M. Woźniak, P. Ksieniewicz, B. Cyganek, K. Walkowiak, Ensembles of Heterogeneous Concept Drift Detectors - Experimental Study, Springer International Publishing, Cham, pp. 538–549. doi:10.1007/978-3-319-45378-1_48
[192] H. Xiao, C. Eckert, Lazy gaussian process committee for real-time online regression, in: Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14–18, 2013, Bellevue, Washington, USA., 2013.
[193] E.S. Xioufis, M. Spiliopoulou, G. Tsoumakas, I.P. Vlahavas, Dealing with concept drift and class imbalance in multi-label stream classification, in: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16–22, 2011, 2011, pp. 1583–1588.
[194] S. ichi Yoshida, K. Hatano, E. Takimoto, M. Takeda, Adaptive online prediction using weighted windows, IEICE Trans. 94-D (10) (2011) 1917–1923.
[195] G. Zenobi, P. Cunningham, Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error, in: Machine Learning: ECML 2001, 2001, pp. 576–587.
[196] P. Zhang, X. Zhu, J. Tan, L. Guo, Classifier and cluster ensembles for mining concept drifting data streams, in: Proceedings ofthe IEEE International Conference on Data Mining, 2010, pp. 1175–1180.
[197] Q. Zhao, Y. Jiang, M. Xu, Incremental learning by heterogeneous bagging ensemble, in: Advanced Data Mining and Applications - 6th International Conference, ADMA 2010, Chongqing, China, November 19–21, 2010, Proceedings, Part II, 2010, pp. 1–12.
[198] X. Zhu, W. Ding, P.S. Yu, C. Zhang, One-class learning and concept summarization for data streams, Knowl. Inf. Syst. 28 (3) (2011) 523–553.
[199] X. Zhu, P. Zhang, X. Lin, Y. Shi, Active learning from stream data using optimal weight classifier ensemble, IEEE Trans. Syst. Man Cybern. Part B 40 (6) (2010) 1607–1621.
[200] I. Zliobaite, Adaptive Training Set Formation, Vilnius University, 2010 Ph.D. thesis.
[201] I. Zliobaite, Controlled permutations for testing adaptive learning models, Knowl. Inf. Syst. 30 (2013a) 1–4.
[202] I. Zliobaite, How good is the electricity benchmark for evaluating concept drift adaptation, arXiv preprint arXiv:1301.3524(2013b).
[203] I. Zliobaite, A. Bifet, M. Gaber, B. Gabrys, J. Gama, L. Minku, K. Musial, Next challenges for adaptive learning systems, SIGKDD Explor. Newsl. 14 (1) (2012) 48–55, doi:10.1145/2408736.2408746.
[204] I. Zliobaite, A. Bifet, B. Pfahringer, G. Holmes, Active learning with evolving streaming data, in: Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases, in: Springer LNCS, 6913, 2011, pp. 597–612.
[205] I. Zliobaite, A. Bifet, B. Pfahringer, G. Holmes, Active learning with drifting streaming data, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2014) 27–39.
[206] I. Zliobaite, A. Bifet, J. Read, B. Pfahringer, G. Holmes, Evaluation methods and decision theory for classification of streaming data with temporal dependence, Mach. Learn. 98 (3) (2015a) 455–482.
[207] I. Zliobaite, M. Budka, F.T. Stahl, Towards cost-sensitive adaptation: when is it worth updating your predictive model? Neurocomputing 150 (2015b) 240–249.
[208] I. Zliobaite, M. Pechenizkiy, J. Gama, An overview of concept drift applications, in: N. Japkowicz, J. Stefanowski (Eds.), Big Data Analysis: New Algorithms for a New Society, Springer, 2016, pp. 91–114.