The Journal of Systems and Software 127 (2017) 278–294

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

A survey on feature drift adaptation: Definition, benchmark, challenges and future directions

Jean Paul Barddal[a,*], Heitor Murilo Gomes[a], Fabrício Enembreck[a], Bernhard Pfahringer[b]

[a] *Graduate Program in Informatics (PPGIa), Pontifícia Universidade Católica do Paraná, Brazil*
[b] *Department of Computer Science, University of Waikato, New Zealand*

ARTICLE INFO

Article history:
Received 29 September 2015
Revised 17 June 2016
Accepted 4 July 2016
Available online 7 July 2016

Keywords:
Feature drift
Feature selection
Data stream mining

ABSTRACT

Data stream mining is a fast growing research topic due to the ubiquity of data in several real-world problems. Given their ephemeral nature, data stream sources are expected to undergo changes in data distribution, a phenomenon called concept drift. This paper focuses on one specific type of drift that has not yet been thoroughly studied, namely feature drift. Feature drift occurs whenever a subset of features becomes, or ceases to be, relevant to the learning task; thus, learners must detect and adapt to these changes accordingly. We survey existing work on feature drift adaptation with both explicit and implicit approaches. Additionally, we benchmark several algorithms and a naive feature drift detection approach using synthetic and real-world datasets. The results from our experiments indicate the need for future research in this area as even naive approaches produced gains in accuracy while reducing resources usage. Finally, we state current research topics, challenges and future directions for feature drift adaptation.

1. Introduction

In the last decades the interest in mining massive and potentially unbounded datasets that arrive at rapid rates, namely data streams, has grown substantially. Examples of data streams include, but are not limited to, sensor networks, wearable sensors, computer network traffic sniffers and video surveillance. Aiming at extracting useful knowledge from these massive amounts of data, a variety of inductive learning techniques were developed and achieved concrete results in both supervised (Barddal et al., 2014; Bifet et al., 2013b; Gama and Kosina, 2011) and unsupervised (Aggarwal et al., 2003; Barddal et al., 2015b; Cao et al., 2006; Kranen et al., 2011) settings.

The most common task in streaming scenarios is classification. In this task, instances are labeled according to a finite set of labels and the goal is to derive a model that accurately classifies upcoming unlabeled instances. Data stream classification algorithms are presented with a great and possibly unbounded amount of data, which are made available to the algorithm in a serialized fast-paced fashion (Gama et al., 2014). Moreover, due to the inherent ephemeral aspects of data streams, one must assume that the underlying concept is unstable, i.e. changes in data distribution are expected to occur, phenomenon called concept drift (Tsymbal, 2004).

Although current techniques for data stream classification handle most of the challenges posed by streaming environments, not much attention has been given to possible changes in the relevance of features through time, the so called *feature drift* (Barddal et al., 2015c). To maintain an accurate predictive model on a data stream that exhibits feature drifts, a classifier must be trained and updated on the set of features that is currently relevant. One way to select the relevant subset of features is via feature selection. However, performing feature selection over data streams is still an open research topic since the majority of existing feature selection algorithms require multiple passes over data.

As in batch learning, if an algorithm is capable of discerning between relevant, irrelevant and redundant features, it is expected to compute faster, show lower memory usage (due to diminished dimensionality) and even produce higher prediction accuracy (Naidu et al., 2014). Nevertheless, performing feature selection incrementally and adaptively as a stream progresses is not straightforward.

In this paper we review the classification task for data streams (Section 2) and the basic concepts of feature selection (Section 3). Next, we discuss existing works that perform feature drift adaptation with both explicit and implicit approaches, showing their major limitations (Section 4). Several surveyed algorithms are then benchmarked on both synthetic and real datasets that contain fea-

* Corresponding author.
E-mail addresses: jean.barddal@ppgia.pucpr.br (J.P. Barddal), hmgomes@ppgia.pucpr.br (H.M. Gomes), fabricio@ppgia.pucpr.br (F. Enembreck), bernhard@cs.waikato.ac.nz (B. Pfahringer).

http://dx.doi.org/10.1016/j.jss.2016.07.005
0164-1212/© 2016 Elsevier Inc. All rights reserved.

ture drifts. We also propose and evaluate a naive algorithm to handle feature drifts. Results obtained corroborate the need for future research on feature selection for data streams (Section 5). Finally, we state the challenges of this research area and future directions (Sections 6 and 7).

2. Data stream mining

Data acquisition and storage is becoming cheaper and easier every day. Recent studies show that 2.5 quintillion of bytes are produced every day, and out of that it is estimated that approximately 90% of overall stored data were created between 2012 and 2014 (Amini and Wah, 2014). Since it might be difficult to extract useful knowledge from this abundant data, data mining techniques have been widely used for this task (Bifet et al., 2013a; Bifet, 2010; Gama et al., 2014; Almeida et al., 2013).

Nowadays, a variety of computational systems create enormous amounts of data, mostly in sequential fashion, and impose several constraints on available processing time and memory space. Extracting interesting patterns from data streams has received growing attention of the data mining community in the last few years (Bifet et al., 2013b; Kosina and Gama, 2012; Aggarwal et al., 2003; 2001).

2.1. Data stream classification

The most common approach for extracting useful knowledge from data streams is classification. Classification is the task that distributes a set of instances into classes (discrete values) accordingly to relations or affinities. Assuming a set of possible classes $Y = \{y_1, \ldots, y_c\}$, a classifier builds a model that predicts for every unlabeled instance $\vec{x}_i$ its corresponding class y_i, ideally with close to perfect accuracy (Bifet, 2010).

The classification task can be formalized as follows: a set of n training instances in the form $(\vec{x}_i, y_i)$ where y_i is a discrete class label and $\vec{x}_i$ is a d-dimensional vector of attributes belonging to a feature set $\mathcal{D}$ with cardinality (or number of dimensions) d, where the feature set can be categorical, ordinal, numeric or most likely a mix of all three types. A classifier uses the training set to produce a model $f : \vec{x} \to Y$ that is used to classify future unlabeled instances.

According to the Bayesian theory, classification can be posed as a function of the prior probabilities of the classes $P[y]$ and the class conditional probability density functions (*pdfs*) $P[\vec{x}|y]$ for all possible classes $y_i \in Y$ (Gama et al., 2014). Classification decision (labeling) is performed accordingly to the posterior probabilities of the classes, where Eq. (1) states the posterior probability for an arbitrary class y_i and $P[\vec{x}] = \sum_{y_i \in Y} P[y_i] \times P[\vec{x}|y_i]$.

$$P[y_i|\vec{x}] = \frac{P[y_i] \times P[\vec{x}|y_i]}{P[\vec{x}]} \tag{1}$$

Data stream classification, or online classification, is a variant of the machine learning task batch classification. Although both are concerned with the problem of learning a model which is able to predict a nominal value for future unlabeled instances, the difference between these two approaches concerns about how data is presented to the learner. In a batch configuration, a static and entirely accessible dataset is provided to the learning algorithm, which usually performs multiple passes over the training set to build its predictive model. Conversely, in streaming environments, instances are not readily available to the classifier for training, instead, these are presented sequentially over time, and the learner must incrementally adapt its model f as new instances become available (Gama, 2010).

Formally, let $\mathcal{S} = [i^t]_{t=0}^{\infty}$ define a data stream providing instances $i^t = (\vec{x}^t, y^t)$, each of which arriving at a timestamp t, where $\vec{x}^t$ is a d-dimensional feature vector belonging to a feature set $\mathcal{D}$, and y^t is the instance's ground-truth label.

2.2. Concept drift

Batch learning techniques assume that there is a static dataset generated by a unknown yet stationary probability distribution, where the data can be physically stored and analyzed in multiple steps by a batch algorithm. Nonetheless, none of the latter assumptions can be verified in streaming scenarios and the development of data stream classifiers must account for several constraints (Abdulsalam et al., 2011; Gama et al., 2014; Silva et al., 2013).

Firstly, instances arrive continuously over time and there is no control over the order that they arrive in nor how they should be processed. Additionally, streams are potentially unbounded and thus, instances should be discarded right after their processing (or accordingly to available main memory space).

Due to the inherent temporal aspect of data streams, their underlying data distribution is expected to change over time, leading to changes in the concept to be learned, a phenomenon called concept drift.

Eq. (2) defines a concept C as a set of prior probabilities of the classes and class-conditional probability density functions (Nguyen et al., 2012).

$$C = \bigcup_{y_i \in Y} \{(P[y_i], P[\vec{x}|y_i])\} \tag{2}$$

Given a stream $\mathcal{S}$, retrieved instances i_t will be generated by a concept C_t. If during every instant t_i of the stream $C_{t_i} = C_{t_{i-1}}$ holds, then C is a stable concept. Otherwise, if between any two timestamps t_i and $t_j = t_i + \Delta$ (with $\Delta \geq 1$) $C_{t_i} \neq C_{t_j}$ holds, then a concept drift has occurred. For more details on the problem of concept drift, the reader is referred to more specific works (Gama, 2010; Gama et al., 2014; Bifet, 2010).

3. Feature selection

Datasets (or streams) for analysis may contain hundreds (or even thousands or millions) of features (attributes), many of which are possibly irrelevant or redundant to the learning task. Dealing with this massive amount of features is not only computationally expensive but it also jeopardizes inductive learning algorithms since the training set would cover a dwindling part of the feature space. Even if we assume a large dataset with trillions of uniformly distributed instances in a moderate attribute space of 100 features, only about 10^{-18} of the potential space would be covered (Domingos, 2012). Also, high dimensional spaces can be a problem due to the "curse of dimensionality", where learning algorithms based on distance computations are known to fail (Aggarwal et al., 2001). To overcome these problems, a variety of feature selection algorithms were developed and aimed at performing dimensionality reduction in batch learning (Hoi et al., 2012; Li et al., 2013; Naidu et al., 2014).

In the following section we describe important concepts related to feature selection which enable us to later formalize and discuss feature drifts properly.

3.1. Definitions

Up to this point, the term "relevance" was used without a formal definition. In this section, we define the concept of relevance in the feature selection task. As stated in Kohavi and John (1997),Rudnicki et al. (2015), there are different definitions available in the literature, nevertheless, several may be contradictory and misleading. In this paper, we provide the most common definitions by dividing features in three types: relevant, irrelevant and redundant.

Definition 3.1. Assuming $S_i = \mathcal{D} \setminus \{D_i\}$, a feature D_i is **relevant** iff

$$\exists S_i' \subset S_i, \text{ such that } P[Y|D_i, S_i'] \neq P[Y|S_i'] \text{ holds} \tag{3}$$

Otherwise, D_i is said to be **irrelevant**.

According to this definition, if a feature that is statistically relevant is removed from a feature set then this will reduce overall prediction power. This definition encompasses two possibilities for a feature to be statistically significant: (i) it is strongly correlated with the class; or (ii) it forms a feature subset with other features, and this subset is strongly correlated with the class (Zhao et al., 2010).

Another goal of feature selection algorithms is to tackle redundant data. A feature becomes redundant due to the existence of other relevant features, which provide similar prediction power.

Definition 3.2. Assuming $S_i = \mathcal{D} \setminus \{D_i\}$, a feature D_i is **redundant** *iff*

$$\exists S_i' \subset S_i, \text{ such that } P[Y|D_i, S_i] = P[Y|S_i] \wedge P[Y|S_i] \neq P[Y|S_i'] \tag{4}$$

Several studies proposed the removal of redundant features as this might jeopardize prediction accuracy due to overfitting (Yu and Liu, 2003), while others noticed that the removal of this type of feature may cause the exclusion of potentially relevant features (Duangsoithong and Windeatt, 2009). Most of existing works propose to find redundant features through correlations (Hall and Smith, 1999; Hall, 2000; Yu and Liu, 2003) or clustering similar attribute patterns into feature clusters (Covões et al., 2009; Park, 2013).

3.2. Feature selection task definition

The feature selection task for data streams is to obtain the optimal subset $\mathcal{D}^* \subseteq \mathcal{D}$ of features that represents the concept to be learned from a dataset or data stream. The goal of feature selection is to remove irrelevant and redundant attributes, while maintaining the probability distribution of the original data classes $P[Y]$. Mining this reduced dimensionality dataset implies a smaller number of parameters in the patterns to be discovered, which leads to easier concepts to understand and which provides as good or better accuracy in the predictive model, whilst requiring less data (Carvalho and Cohen, 2006).

The problem of feature selection can be formalized as an optimization problem.

Definition 3.3. Assuming the full and variable set of features $\mathcal{D}$, the goal is to select a subset $\mathcal{D}^*$ that retains only the relevant information in $\mathcal{S}$. Suppose that the goodness of a subset of features $\mathcal{D}' \subseteq \mathcal{D}$ is given by $Q(\cdot)$, then feature selection can be stated as in Eq. (5), where d_{max} is the upper bound on the number of selected features.

$$\mathcal{D}^* = \underset{\mathcal{D}' \subseteq \mathcal{D}}{\text{argmax}}\ Q(\mathcal{D}') \text{ subject to } |\mathcal{D}'| \leq d_{max} \tag{5}$$

Finding $\mathcal{D}^*$ is a difficult task that, assuming $d_{max} = d$, requires an exploratory search which is by definition $\mathcal{O}(2^d)$. In Fig. 1 we present a graphical representation of the features subset space assuming $\mathcal{D} = \{D_1, D_2, D_3, D_4\}$. Due to the exponential computational complexity, most algorithms employ heuristics to guide the selection process, which may lead to suboptimal selected discriminant feature subsets.

3.3. Taxonomy of feature selection methods

Feature selection algorithms are commonly divided into three classes: filters, wrappers and embedded methods Guyon (2003). In this section we briefly describe these three categories.

Filters. Filters apply statistical measures to assign a score to each feature. Features are then ranked by scores and either selected to be kept or removed given a threshold. These methods are usually univariate and consider each feature independently. Two important traits of filters are their independence from the learning algorithm adopted and low computational cost. Examples of filter methods include the χ^2 test, Information Gain, Entropy, Correlation Coefficient Scores, Las Vegas Filters, Relief and ReliefF (Naidu et al., 2014).

Wrappers. Wrappers consider the selection of a subset of features as a search problem, where different combinations are prepared, evaluated and pairwisely compared, usually in bottom-top or top-bottom approaches. A predictive model is used to evaluate each combination of features and to assign a score based on prediction accuracy. Therefore, wrappers are sensitive to the learning algorithm's bias, i.e. recognize that certain algorithms may work better with different features (Aggarwal, 2014). An important drawback of wrappers is their computational cost, which is prohibitive in high dimensional or in real-time scenarios. The most common search processes are: best-first search, stochastic hill-climbing algorithms, forward and backward passes, beam search and simulated annealing.

Embedded methods. Embedded methods learn which features best contribute to the overall accuracy of the learning algorithm while the model is created. For instance, decision tree learning can be considered an embedded method since the construction of the tree and the selection of the features are interleaved, but the selection of features itself is done by filters. Embedded approaches interact directly with the learning algorithm and present better computational complexity than wrappers (Domingos and Hulten, 2000).

3.4. Feature drift

Most existing algorithms for data streams tackle the infinite length and drifting concept characteristics. However, not much attention has been given to feature drifts. Feature drifts occur whenever a subset of features becomes, or ceases to be, relevant to the concept to be learned. This forces a learner to adapt its predictive model to ignore irrelevant attributes and account for the newly relevant ones (Nguyen et al., 2012).

Definition 3.4. Given a feature space $\mathcal{D}$ at a timestamp t, it is possible to select the ground-truth relevant subset $\mathcal{D}_t^* \subseteq \mathcal{D}$ such that $\forall D_i \in \mathcal{D}_t^*$ Definition 3.1 holds and $\forall D_j \in \mathcal{D} \setminus \mathcal{D}_t^*$ the same definition does not. A feature drift occurs if, at any two time instants t_i and $t_j = t_i + \Delta$, $\mathcal{D}_{t_i}^* \neq \mathcal{D}_{t_j}^*$ holds.

Definition 3.5. Let $r(D_i, t_j) \in \{0, 1\}$ denote a function that determines whether Definition 3.1 holds for a feature D_i at a timestamp t_j of the stream. A positive relevance, i.e. $r(D_i, t_i) = 1$, states that $D_i \in \mathcal{D}^*$ at a timestamp t_i and that D_i impacts the underlying probabilities $P[\vec{x}|y_i]$ of the concept C_t of $\mathcal{S}$. A feature drift occurs whenever the relevance of an attribute D_i changes in a timespan between t_j and t_k, as stated in Eq. (6).

$$\exists t_j \exists t_k, t_j < t_k, r(D_i, t_j) \neq r(D_i, t_k) \tag{6}$$

Changes in $r(\cdot, \cdot)$ directly affect the ground-truth decision boundary to be learned by the classifier. Therefore, feature drifts can be posed as a specific type of concept drift that may occur with or without changes in the data distribution $P[\vec{x}]$ (Barddal et al., 2015a; 2015c).

As in conventional concept drifts, changes in $r(\cdot, \cdot)$ may occur during the processing of the stream. Thus, data stream learners are expected to detect changes in $\mathcal{D}^*$, discerning between features that

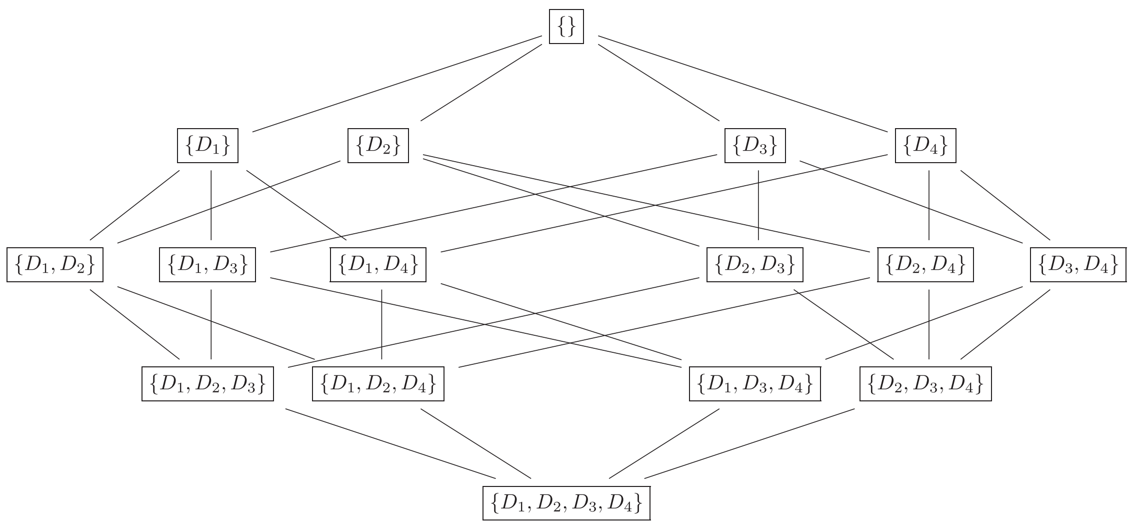
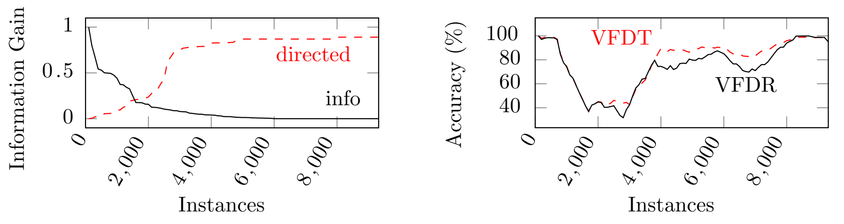


Fig. 1. Feature subsets space assuming $\mathcal{D} = \{D_1, D_2, D_3, D_4\}$.



(a) Information gain of features "directed" and "info" during the stream.

(b) Accuracy obtained for a decision tree (VFDT) and a decision rule learner (VFDR).

Fig. 2. Analysis of information gain for two specific features and accuracy obtained on the Spam Corpus dataset. Adapted from Barddal et al. (2015a; 2015c).

became irrelevant and those that are now relevant and vice-versa. Finally, it is necessary to either (i) discard and learn an entirely new classification model; or (ii) adapt the current model to these drifts (Nguyen et al., 2012).

Although feature drifts may occur in a variety of environments, one of the most common domains is text mining. In order to exemplify a feature drift, we refer to the e-mail spam detection system presented in Katakis et al. (2006). This system was a result of a text mining process on an online news dissemination system. Essentially, this work intended to create an incremental filtering of emails that classifies emails as spam or ham and, based on this classification, decides whether this email is relevant for dissemination among users. The dataset contains 9,324 instances and 39917 features, such that each attribute represents the presence of a single word (feature) in an instance (e-mail). This dataset, called Spam Corpus, is known for containing a feature drift which occurs gradually around the instance of number 1,500 (Barddal et al., 2014; Katakis et al., 2006; Barddal et al., 2015c) and that highly impacts on the learner.

In Fig. 2a we present a plot of the information gain (Hall et al., 2009) of two specific attributes presented in this problem, namely "directed" and "info", where one can see that the importance of these two features starts exchanging gradually around instance 1,500 (Barddal et al., 2015a). Detecting and discerning the two features that exchange relevance as the stream progresses is an important task that must be embedded within streaming learning algorithms, since changes greatly impact the accuracy of the model (Fig. 2b) and learning with a subset of the whole feature set is also computationally faster. We refrain from providing a detailed description of these classifiers since the Very Fast Decision Tree (VFDT) and Very Fast Decision Rules (VFDR) are discussed in Sections 4.1 and 4.2, respectively.

3.5. A note on Dynamic Feature Selection versus streaming feature selection

It is important to emphasize the difference between Dynamic Feature Selection for data streams and Streaming Feature Selection (also commonly referred as Online Feature Selection (Hoi et al., 2012; Li et al., 2013)). Streaming feature selection regards the possibility of finding the best subset of features in a very high-dimensional space (hundreds of thousands or millions of dimensions), which is a typical problem of big data (Hoi et al., 2012). Although both tasks' objectives overlap, streaming feature selection receives as input a stream of features (not instances), and their inclusion in the model is performed sequentially, without observing future features (Zhou et al., 2005). Therefore, Streaming Feature Selection is used in batch learning, when the amount of features is gigantic, scaling up to thousands or millions of attributes and the amount of instances is static and does not vary with time.

4. Existing works on overcoming feature drifts

There are few works in the literature that perform feature selection during stream learning. There are even fewer that aim at explicitly detecting and adapting to feature drifts. In this section, we summarize existing algorithms that perform feature selection as the stream progresses, either assuming the existence, or not, of feature drifts.

Table 1
Summary of existing algorithms that perform feature selection during stream learning.

Algorithm	Learning Approach	Feature Selection Algorithm	Feature Drift Adaptation Method	Explicit Dynamic Feature Selection	Reference
VFDT	Tree	Entropy Information Gain Gini Coefficient	–		Domingos and Hulten (2000)
Facil	Rules	Purity	–		Ferrer-Troyano et al. (2005)
VFDR	Rules	Entropy	–		Gama and Kosina (2011)
Streaming Random Forest	Ensemble (Trees)	–	Randomness/Combinatorics		Abdulsalam et al. (2007), Abdulsalam et al. (2011)
Random Rules	Ensemble (Rules)	–	Randomness/Combinatorics		Almeida et al. (2013)
Streaming Stacking	–	Ensemble	Combinatorics		Bifet et al. (2010a)
CVFDT	Tree	Entropy Information Gain Gini Coefficient	Windowing	√	Hulten et al. (2001)
HEFT-Stream	Ensemble	FCBF	Windowing	√	Nguyen et al. (2012)
HAT	Tree	Entropy Information Gain Gini Coefficient	Windowing	√	Bifet and Gavaldà (2009)
HUWRS	Ensemble	–	Windowing		Hoens et al. (2011)

In Table 1 we summarize existing algorithms. We categorize them accordingly to four characteristics: (i) their learning approach, (ii) the specific feature selection algorithm, (iii) feature drift adaptation method adopted; and (iv) whether they perform explicit dynamic feature selection or not.

We start by discussing two important and widely used approaches for classifying data streams: decision trees (Section 4.1) and decision rules (Section 4.2). Although most part of the summarized algorithms presented in this paper were not developed aiming at performing feature drift detection and adaptation, we discuss them and highlight their capabilities to attack this problem, either via randomness (Section 4.3), combinatorics (Section 4.4) or windowing (Section 4.5).

4.1. *Decision tree learning*

Learning with decision trees is a predictive approach used in statistics, data mining and machine learning. In its simplest implementations, each internal node contains a test on a feature $D_i \in \mathcal{D}$, each branch from a node corresponds to an outcome of the test and each leaf contains a possible prediction (class value from Y) (Bifet, 2010).

Predictions for instances $\vec{x}$ are obtained by traversing the tree with features' values, determining which branch should be followed, until a leave is reached.

Decision trees are learned by recursion, replacing leaves by test nodes, starting at the root. The feature of each test node is chosen by comparing all the available attributes $D_i \in \mathcal{D}$ given an heuristic measure.

4.1.1. *Very fast decision tree*

The Very Fast Decision Tree (VFDT) algorithm constructs decision trees by using constant memory and constant time per sample (Domingos and Hulten, 2000). Trees are built by recursively replacing leaves with decision nodes, as data arrives. Different heuristic evaluation functions are used to determine whether a split should be performed or not, such as Entropy (Eq. (7)), Correlation (Eq. (8)), Information Gain (Eq. (9)) and Gini Impurity (Eq. (10)) (Han et al., 2011), where n is the amount of instances in the dataset analyzed.

$$H(D_i) = -\sum_{q \in D_i} P[q] \log_2 P[q] \tag{7}$$

$$C(D_i, Y) = \frac{\sum_{q \in D_i} \sum_{y_i \in Y} (q - \bar{D_i})(y_i - \bar{Y})}{\sqrt{\sum_{q \in D_i} (q - \bar{D_i})^2} \sqrt{\sum_{y_i \in Y} (y_i - \bar{Y})^2}} \tag{8}$$

$$IG(D_i) = H(D_i) - \prod_{D_j \in \mathcal{D}, D_j \neq D_i} \frac{H(D_j)}{n} \tag{9}$$

$$GI(D_i) = 1 - \sum_{q \in D_i} (P[q])^2 \tag{10}$$

In order to determine whether a new branch should be built in the tree, VFDT assumes that the input data meets the Hoeffding bound (Hoeffding, 1963).

Definition 4.1. The Hoeffding Inequality states that with probability $(1 - \delta)$ the true mean of a variable is at least $(\bar{r} - \epsilon)$, where ϵ is given by Eq. (11), δ is a user-given confidence bound, $r \in \mathbb{R}^+$ is a random variable with range R, n is the number of independent observations and $\bar{r}$ is the mean computed by the latter observations.

$$\epsilon = \sqrt{\frac{R^2 \ln\left(\frac{1}{\delta}\right)}{2n}} \tag{11}$$

The Hoeffding bound is able to give results regardless of the probability distribution that generates data. However, the number of observations needed to reach certain values of δ and ϵ are different across different probability distributions (Bifet and GavaldÈâÃá, 2007), therefore, it must be seen as a pessimistic bound. Generally, with probability $(1 - \delta)$, one can say that one attribute is superior when compared to others when the observed difference of information gain (or any other heuristic metric that computes the importance of an attribute) is greater then ϵ.

Although VFDT performs embedded feature selection in data streams, it assumes that the distribution generating data does not change over time, therefore, it does not detect nor adapt to possible drifts.

4.2. *Decision rule learning*

Although decision trees account for readability, in some specific scenarios, where trees tend to grow large, they become hard to understand since nodes appear in a specific context defined by tests at antecedent nodes (Almeida et al., 2013). In contrast, classifiers based on rules have the advantage of both modularity and interpretability, where each rule is independent of the others and can be interpreted in isolation from any other rules.

A decision rule is a logic predicate in the *IF antecedent THEN label* form, where the antecedent is a conjunction of conditions over features $D_i \in \mathcal{D}$ and the label is a possible class value that belongs to Y.

4.2.1. Facil

The first streaming rule learner published was Facil (Ferrer-Troyano et al., 2005). Facil creates rules according to the arrival of instances in an incremental fashion. In order to cope with concept drifts, Facil encompasses both explicit and implicit forgetting mechanisms. The explicit approach occurs when the examples are older than a user-given threshold W, adopting a sliding window approach to eliminate old rules. Conversely, implicit forgetting occurs when removing rules that are not relevant as they do not enforce any concept description boundary. This approach's rationale is that rules are inconsistent if they store both positive and negative instances that are near to one another at the decision boundary. Therefore, rules are removed if the impurity (ratio between positive instances it covers and its total number of cover examples) of a rule reaches a user-given threshold. Whenever the removal of a rule occurs, the subset originally covered by these rules are used to form two new rules that achieve satisfiable purity.

4.2.2. Very fast decision rules

A more robust approach for learning rules from data streams is proposed in Gama and Kosina (2011). This algorithm, called Very Fast Decision Rules (VFDR), is capable of learning ordered and/or unordered rules. The algorithm starts with an empty rule set and rules are grown and expanded according to the minimization of entropy (Eq. (7)) of class labels Y of instances covered by each rule. Additionally, in order to determine whether a rule should be expanded, VDFR also adopts the Hoeffding bound (Eq. (11)).

VFDR considers two cases of rule learning: ordered and unordered sets of rules. In the former, all labeled instances update statistics of the first rule triggered. While in the latter, labeled instances update statistics of all the rules that cover it. In both cases, if no rules cover an instance, the default rule is updated to cover it.

Finally, VFDR encompasses two classification strategies. The first uses only information about class distribution and does not account for features' values. Since it uses a small part of the available information, it is a crude approximation of the instances. Conversely, in an informed strategy, instances are classified with the class that maximizes the posteriori probability assuming the independence of attributes given the class ($P[y_i|\vec{x}] \propto P[y_i] \prod P[\vec{x}|y_i]$).

4.3. Randomness

Diversity is a trait of a variety of recently proposed algorithms for learning from data streams (Bifet et al., 2009; 2010c; Oza, 2005), particularly ensembles. Ensembles are sets of classifiers that are trained in parallel or sequentially and have their predictions aggregated during polling (Dietterich, 2000). In several of these approaches, experts of an ensemble are trained with different inputs in order to promote diversity (Kuncheva and Faithfull, 2014). A well-known approach for inducing diversity in ensembles is Bagging (Breiman, 1996). Originally, a bagging ensemble is composed of m classifiers, which are trained with subsets (bootstraps) of the whole training set. However, sampling usually is not feasible in a data stream configuration, since that would require storing all instances before creating subsets. Therefore, authors in Oza (2005) observed that the probability of an instance $\vec{x}_i$ to be selected for a subset can be approximated by a Poisson distribution with $\lambda = 1$.

Although promoting diversity through instances is an interesting approach to enhance a learner's accuracy, more recent approaches exploit and induce diversity via different feature subsets, i.e. vertical partitioning of data (Abdulsalam et al., 2007; 2011). By learning through ensembles with different features, experts learn partially (or completely) disjoint areas of the feature space, resulting in a highly diverse ensemble. Although these algorithms do not focus explicitly on adapting to feature drifts, they do present implicit adaptation to this characteristic of data streams by covering different feature subspaces in parallel.

4.3.1. Streaming random forest

The Streaming Random Forest classifier is an adaptation of the ensemble-based Random Forest classifier (Breiman, 2001). Random forests are ensembles of decision trees. Assuming a dataset with n instances, each belonging to a feature set $\mathcal{D}$, random forests grow a set of trees, each using a bootstrap sample drawn from the full training set. Bootstrapping guarantees that about $\frac{n}{3}$ of the records are not included in the training set and so are available for evaluation of each tree (Abdulsalam et al., 2011).

The construction of each tree follows a variant of the typical decision tree building algorithm. In standard decision tree algorithms, the set of attributes considered at a node is the entire set $\mathcal{D}$. Conversely, in the Random Forest algorithm, the set of attributes considered at each node of each tree of the ensemble is a different randomly chosen subset $\mathcal{D}' \subset \mathcal{D}$, where $|\mathcal{D}'| \leq M$.

As an ensemble, the labeling of each new instance is the fusion of the votes of all the trees. The random forest classification error depends on (i) the correlation among its component trees, since smaller correlations cause higher variance canceling in voting and (ii) the strength of each individual tree, since the more accurate each subtree is, the better its individual vote and the smaller the error rate is (Abdulsalam et al., 2007).

Therefore, the value of M is a sensitive parameter of random forests that must be chosen carefully. Small values of M tend to increase the strength of each individual tree, while decreasing the correlation between them (Abdulsalam et al., 2007).

4.3.2. Random rules

In Almeida et al. (2013), authors extend the VFDR algorithm by promoting randomness. This algorithm, called Random Rules for Data Stream (RR), encompasses the following parameters: a number of rule sets (N_s) and the number of attributes M that respects the $M < |\mathcal{D}|$ restriction.

Initially, each of the composing rule sets is empty and each of these is associated with a random subset $\mathcal{D}' \subset \mathcal{D}$ of size M. For each instance i_t retrieved from $\mathcal{S}$, RR generates a random number p between 0 and 1 for each rule set. If $p \geq T_{rnd}$, a user-given threshold, RR verifies whether each rule set contains a rule that covers i_t, i.e. if all the literals of the rule are true for the given instance. If the above condition holds, all covering rules are expanded using only the features adopted by the rule set $\mathcal{D}'$. Otherwise, that is, if no rules cover i_t, then the default rule is updated to cover it, again, respecting the features in $\mathcal{D}'$.

Finally, authors presented two voting schemes. The first classifies $\vec{x}_t$ with the class y_i that maximizes $P[y_i]$, while the second assumes the class that maximizes the posteriori probability $\max_{y_i \in Y} P[y_i|\vec{x}_t]$ presented in Eq. (1).

4.4. Combinatorics

By exploring combinatorics, ensembles of decision trees and random rules algorithms can be extended and posed as dynamic wrappers for dynamic feature selection for data streams. If one assumes an ensemble of decision trees or a random rule algorithm, where each of its containing experts is trained with a different subset of the entire feature set $\mathcal{D}$, and that the cardinality of each subset is at maximum M, the ensemble would contain $\sum_{i=1}^{M} \binom{M}{i}$ experts. Although training this high amount of experts is computationally expensive in terms of both processing time and memory

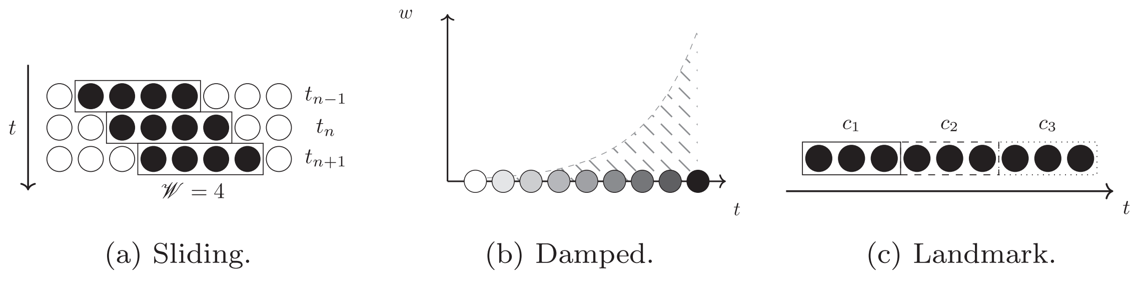


Fig. 3. Window models. Adapted from Silva et al. (2013).

space, it guarantees that a near optimal (or optimal, if $M \geq |\mathcal{D}^*|$) subset $\mathcal{D}'$ allocated to one of the experts will maximize its acuity metric (Abdulsalam et al., 2011). Therefore, by applying weighted majority voting (Kolter and Maloof, 2007), feature drifts can be detected according to the increase of the weights of experts with the current most discriminative subsets of features, while those with subsets of irrelevant features will possess lower weights due to lower accuracy performance.

4.4.1. *Streaming stacking*

In Bifet et al. (2010a), authors produce a classification model based on an ensemble of decision trees, each of which is built from a random and distinct subset of $\mathcal{D}' \subset \mathcal{D}$. The overall model is formed by combining the log-odds of the class probabilities of its containing trees using sigmoid perceptrons, with one perceptron per class. Contrarily to the conventional boosting approach, which forms an ensemble in a greedy fashion, each tree is built in sequence by assigning weights as a by-product and their method generates trees in parallel and combines them using perceptron classifiers by applying stacking (Wolpert, 1992). Due to the streaming scenario, VFDTs are used as ensemble members since they are able to be trained incrementally. Additionally, the ensemble adopts the ADWIN change detector (Bifet and GavaldÈâÃá, 2007) in order to detect and adapt to possible concept drifts.

This approach is based on generating trees for all possible feature subsets of a given size M. Assuming a feature set $\mathcal{D}$ of size d, there are $\binom{d}{M}$ possible subsets. Clearly, only moderate values of M or values close to d are practical, since $\binom{d}{M} = \binom{d}{d-M}$. Authors claim that $M = 2$ is very practical for datasets with a moderate number of features, although certainly not feasible for high-dimensional data (e.g. Spam Corpus Katakis et al., 2006).

4.5. *Windowing*

A common approach for both data management and dealing with drifting data is to maintain a predictive model consistent with a set of recent examples (Silva et al., 2013) given three window models: sliding, damped and landmark. In all cases, the difficulty is to select their appropriate size due to the stability-plasticity dilemma. While short windows reflect the current data distribution and ensure fast adaptation to drifts (plasticity), shorter ones worsen the performance of the system in stable areas. Conversely, larger windows give better performance in stable periods (stability), however, they are also slower to respond to drifts (Gama, 2010).

Sliding windows (Fig. 3a) store in memory a fixed or variable amount $\mathscr{W}$ of recent examples. Whenever a new instance arrives, it is enqueued in a FIFO (first in, first out) policy data structure, where the oldest one is discarded. The rationale behind this type of window is that buffered data reflect the current concept adequately. In opposition to sliding windows, damped windows (Fig. 3b) associate a weight w to each datum, which decays with time. This windowing technique is interesting because weights can be seen as indicatives of how important an instance is to the current concept, thus, may be accounted for during voting. Finally, landmark windows (Fig. 3c) require processing a stream by handling disjoint chunks c_i of data separately by instances called “landmarks”. Landmarks can be defined in terms of time, in terms of the number of instances seen since the previous landmark or accordingly to memory constraints. All instances belonging to the same landmark window are stored or summarized into the same data structure, which is used for training. In this section we present existing works that rely in windowing approaches to explicitly adapt to feature drifts.

4.5.1. *Concept-adapting very fast decision tree*

Concept-adapting Very Fast Decision Tree (CVFDT) is an extension to VFDT to deal with drifts (Hulten et al., 2001). CFVDT keeps a model consistent with respect to the current state of a sliding window from the data stream, thus creating and replacing alternate decision subtrees when it detects that the distribution of data is changing at a node. As instances i_t arrive, CFVDT updates the statistics at its nodes by decrementing counters according to the oldest element in the window, which is about to be dequeued and “forgotten”.

Therefore, CFVDT is an Hoeffding Tree which periodically verifies the statistics of nodes to determine if the Hoeffding criterion is still met. According to user-given parameters T_0, T_1 and T_2, CFVDT traverses the entire decision tree and checks at each test node if the splitting attribute is still the best when compared to others. If there is an alternate better splitting attribute, the entire subtree is replaced by a new split node with this attribute. Next, during the next T_1 instances, all retrieved instances from $\mathcal{S}$ are used to build the new subtree, which are then tested with the following T_2 instances.

4.5.2. *Heterogeneous ensemble for data stream*

The Heterogeneous Ensemble with Feature Drift for Data Stream (HEFT-Stream) is an algorithm that incorporates feature selection into an heterogeneous ensemble to adapt to different types of concept and feature drifts (Nguyen et al., 2012). HEFT-Stream adopts an modification of the Fast Correlation-Based Filter (FCBF) algorithm so it dynamically updates the selected relevant feature subset of a data stream.

FCBF is a feature selection algorithm where the class relevance and pairwise dependencies between features are accounted for. Based on information theory, FCBF adopts Symmetrical Uncertainty (*SU*) to compute dependencies of features and class relevance. Using a top-down approach, starting from the whole feature set $\mathcal{D}$, FCBF heuristically applies a backward selection technique to remove irrelevant and redundant features.

Symmetrical uncertainty uses both entropy and conditional entropy to calculate the dependencies of features. Assuming two arbitrary features D_i and D_j, the symmetrical uncertainty between these two can be computed according to Eq. (12), where $H(\,\cdot\,)$ is the entropy of a feature (Eq. (7)), $H(\,\cdot, \cdot)$ is the conditional entropy and $MI(\cdot, \cdot)$ is the mutual information between two features (Eq. (13)).

$$SU(D_i, D_j) = 2\left[\frac{H(D_i) - H(D_i|D_j)}{H(D_i) + H(D_j)}\right] = 2\left[\frac{MI(D_i, D_j)}{H(D_i) + H(D_j)}\right] \quad (12)$$

$$MI(D_i, D_j) = \sum_{q \in D_i}\sum_{r \in D_j} P[q, r] \log \frac{P[q, r]}{P[q] \times P[r]} \quad (13)$$

HEFT-Stream adopts a landmark windowing approach. Incoming data is stored in a buffer with a predefined size. Next, the matrix of symmetrical uncertainty values is computed to select the most relevant feature subset. After the processing of each data chunk, HEFT-Stream postulates that a feature drift has occurred if two consecutive selected subsets of features differ.

Additionally, in order to boost the overall ensemble accuracy, HEFT-Stream promotes diversity among member classifiers by encompassing an Online Bagging sampling procedure (Oza, 2005).

Classification of each instance is performed through a weighted combination of member classifiers classifications. Each member classifier k is associated to a weight w_k (Eq. (14)) which is an accumulated error from its creation time to the current time. The weight w_k is stated in Eq. (14) where α is a padding value which was originally empirically set to 0.001 and E_k is the accumulative error of the kth member classifier.

$$w_k = \frac{1}{(E_k + \alpha)} \times \left[\sum_{m=1}^{K} (E_m + \alpha)^{-1}\right] \quad (14)$$

Finally, at the end of a chunk, the classifier with the highest value of E_k is replaced by a new classifier. This new classifier is associated with the feature set $\mathcal{D}'$ selected by FCBF and its type corresponds to the most accurate expert of the ensemble.

Although HEFT-Stream is stated as a generic ensemble capable of using any kind of base classification learners, authors only show results for a combination of an Updatable Naïve Bayes algorithm and VFDT.

4.5.3. Hoeffding adaptive tree

Most of decision tree-based algorithms for learning from data streams either assume that the underlying distribution is static, e.g. VFDT (see Section 4.1.1), or contain hardwired constants concerning the speed or frequency of change, e.g. CVFDT (see Section 4.5.1). These choices are inconclusive and often incorrect due to the plasticity-stability dilemma, but also since one cannot assume that all changes in a stream occur with the same frequencies and lengths.

In Bifet and Gavaldà (2009) authors proposed the adoption of an adaptive sliding window drift detector, named ADWIN (Bifet and GavaldÈâÃá, 2007), inside decision trees for data streams. Their proposal, called Hoeffding Adaptive Tree (HAT), is an extension to CVFDT in which an ADWIN drift detector is used to monitor and flag changes in split nodes of the tree. Therefore, instead of relying on window parameters T_0, T_1 and T_2 for re-evaluating split nodes, HAT replaces split nodes when a significant error rate change occurs, given a confidence level δ that is inputted to ADWIN.

HATs are thus able to cope with both concept drifts and feature drifts since split nodes are re-evaluated. This allows split nodes to be consistent in terms of the feature adopted to perform the split and in which range/value of this feature the decision should be made. One of the major drawbacks of this method is that ADWIN is known for triggering too many false positives (Huang et al., 2015), i.e. it flags changes when they do not really occur.

4.5.4. Heuristic updatable weighted random subspaces

The Heuristic Updatable Weighted Random Subspaces (HUWRS) is a random subspace-based ensemble for data streams (Hoens et al., 2011). HUWRS works under the hypothesis that when a feature drift occurs, there is no need to learn an entirely new predictive model. Instead, authors recommend building experts of the ensemble based on random subspaces, while feature drifts are detected accordingly to a landmark window. HUWRS assumes that data arrives in batches. On each arriving batch, features are discretized in equal-sized bins and the class distribution inside each bin of every feature is computed.

HUWRS postulates that a feature drift occurs in a feature D_i if the Hellinger weight between the class distribution of the current and prior landmarks differ at least by p%, a user-given threshold. The Hellinger weight is given by Eq. (15), which is a normalization to the Hellinger distance, given by Eq. (16). In Eqs. (15) and (16), Y' and Y'' stand for the class distributions of the current and prior landmarks for an arbitrary feature D_i.

$$w_H(Y', Y'') = \frac{\sqrt{2} - d_H(Y', Y'')}{\sqrt{2}} \quad (15)$$

$$d_H(Y', Y'') = \sqrt{\sum_{q \in D_i}\left(\sqrt{P[Y'|D_i = q]} - \sqrt{P[Y''|D_i = q]}\right)^2} \quad (16)$$

Since a low Hellinger distance means a high agreement in the two distributions, a low Hellinger distance should correspond to a high weight. We emphasize that $\sqrt{2}$ is the maximum Hellinger distance between distributions for binary classification problems, thus, this value is used as a normalization factor so that the Hellinger weight is bounded in [0; 1].

Whenever a feature drift is flagged for a feature D_i, HUWRS resets only the experts associated with such feature. Therefore, HUWRS is expected to adapt to feature drifts while performing less retraining when compared to full reset approaches. One important drawback of HUWRS is that the experts adopted in the ensemble are not incremental. Therefore, classifiers are unable to increment their models if conventional concept drifts occur.

5. Empirical analysis

In this section, we assess the impact of feature drifts on data stream classification algorithms. First, we introduce the experimental protocol adopted, including data generators, one real-world dataset and statistical procedures (Section 5.1). Next, we discuss the results obtained, highlighting existing algorithms difficulties to overcome feature drifts (Section 5.2). Finally, we propose a naive solution to the feature drift problem by splitting and treating the stream into disjoint chunks of data and applying simple filters. We show the efficiency of this approach in both synthetic and real world problems, thus, highlighting the need for future research in this area and the room for more sophisticated approaches (Section 5.3).

5.1. Experimental protocol

In this section we present the experimental protocol adopted. We start by presenting the datasets used and later the evaluation procedure, focusing on accuracy, processing time and memory usage metrics and statistical testing procedure.

5.1.1. Generators

In order to evaluate whether a learning algorithm is able to work in different scenarios, it is necessary to assess its performance over different datasets. In opposition to real-world data, synthetic data stream generators are important and often used due to their flexibility, since they offer a precise definition of drifts types and locations during the streams. In this section we propose and survey generators capable of synthesizing feature drifts, thus, enabling proper evaluation of learning algorithms in these scenarios. All the values picked for presented parameters were chosen accordingly to their usage in previous papers of the area.

SEA-FD. In Barddal et al. (2015a), authors proposed a data stream generator that extends the SEA generator (Street and Kim, 2001). SEA-FD simulates streams with $d > 2$ uniformly distributed features given by the user, where $\forall D_i \in \mathcal{D}, D_i \in [0; 10]$ and only two randomly picked features are relevant to the concept to be learned: $\mathcal{D}^* = \{D_\omega, D_\zeta\}$. As in Street and Kim (2001),Barddal et al. (2015a), the class value y is defined by Eq. (17), where θ is a user-given threshold. In our experiments, $\theta = 7$ and each instance synthesized has a 5% probability of being generated as noise.

$$y = \begin{cases} 1, & \text{if } D_\alpha + D_\beta \leq \theta \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

BG-FD. The Binary Generator with Feature Drift (BG-FD) generates instances composed by boolean ({0, 1}) features. BG-FD has three functions: BG1-FD, BG2-FD and BG3-FD, all inspired by Hall (2000). In BG1-FD, from the entire set of features $\mathcal{D}$, only a random subset $\mathcal{D}^* \subset \mathcal{D}$ is relevant to the concept to be learned. Additionally, $|\mathcal{D}^*| = d_r$, where d_r is a user-given parameter. Conversely, in BG2-FD and BG3-FD we have $\mathcal{D}^* = \{D_\alpha, D_\beta, D_\epsilon\}$. Labels of instances are given according to three different functions presented in Eqs. (18)–20 for BG1-FD, BG2-FD and BG3-FD, respectively.

$$y = \begin{cases} 1, & \text{if } \bigwedge_{D_i \in \mathcal{D}^*} D_i \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

$$y = \begin{cases} 1, & \text{if } (D_\alpha \wedge D_\beta) \vee (D_\alpha \wedge D_\epsilon) \vee (D_\beta \wedge D_\epsilon) \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

$$y = \begin{cases} 1, & \text{if } (D_\alpha \wedge D_\beta \wedge D_\epsilon) \vee (\neg D_\alpha \wedge \neg D_\beta \wedge \neg D_\epsilon) \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

In all cases, class labels $y_i \in Y$ are evenly likely to occur and instances have a 5% probability of being generated as noise.

RTG-FD. The original Random Tree Generator (RTG) builds a decision tree by randomly performing splits on features and assigning a random class label to each leaf (Bifet et al., 2010b). Instances are created by generating a random valued $\vec{x}$ and traversing the tree for its corresponding label. We propose an extension to this generator, namely RTG-FD, such that only a random subset of features $\mathcal{D}^* \subset \mathcal{D}$ are relevant. Assuming $\mathcal{D}_i = \mathcal{D} \setminus \mathcal{D}^*$ as the subset of irrelevant features, $|\mathcal{D}_i|$ is a user-given parameter.

5.1.2. Drift framework

We synthesize feature drifts in streams accordingly to the framework proposed in Bifet (2010). This framework models a drift as the change between two pure distributions, each given by a distinct concept.Intuitively, at the beginning of a drift window there is a higher probability that instances belong to the prior concept C_A. As we move towards its end, the probability that an instance belongs to the posterior concept C_B increases. The drift window ends when concept C_B becomes stable. To model the probability that every new instance i_t drawn from $\mathcal{S}$ belongs to concept C_A or C_B, a sigmoid function as stated in Eq. (21) is adopted, where $P[C_B]$ and $P[C_A] = |1 - P[C_B]|$ are, respectively, the probabilities of i_t belonging to C_A or C_B, W is the drift window size, t is the current timestamp and t_0 is the time of the drift, i.e. when $P[C_A] = P[C_B]$ holds.

$$P[C_B] = |1 - P[C_A]| = \left(1 + e^{-W(t-t_0)}\right)^{-1} \tag{21}$$

Therefore, with the latter generators, feature drifts occur when the relevant subset of features $\mathcal{D}^*$ of C_A differs from the relevant subset of features $\mathcal{D}^*$ of the subsequent concept C_B.

Synthetic data streams. Synthetic experiments encompass the usage of all presented generators. All streams created have a length of 100,000 instances, $|\mathcal{D}| = 50$ and $|\mathcal{D}^*| = 3$, with the exception of SEA-FD experiments, where $|\mathcal{D}^*| = 2$. Streams with an (A) suffix contain 9 equally distributed abrupt ($w = 1$) feature drifts, while streams with a (G) contain 9 drifts at the same time points as for (A), however, these drifts are gradual ($w = 1,000$).

Real datasets. Complementing the synthetic data streams, our experiments also encompass the Spam Corpus dataset (Katakis et al., 2006). This dataset (earlier discussed in Section 3.4) was extracted from a text mining process on an online news dissemination system. The Spam Corpus dataset contains 9324 instances and 39,917 features, such that each attribute represents the presence of a single word (feature) in the instance (e-mail). Also, this dataset is known for containing a concept drift which occurs gradually around the instance number 1500 (Katakis et al., 2006; Barddal et al., 2015c).

5.1.3. Evaluation procedure

Our evaluation procedure assesses an algorithm's efficiency in terms of accuracy, processing time and memory usage. To quantify the accuracy of classifiers, we adopted the Prequential test-then-train procedure (Gama and Rodrigues, 2009). Although the Prequential evaluation is known for being pessimistic, authors in Gama and Rodrigues (2009) claim that it converges to a periodic holdout estimate when estimated over a sliding window. The Prequential accuracy of a classifier is computed, at a timestamp t_i, over a sliding window of size w', according to Eq. (22), where $L(\cdot, \cdot)$ is a loss function (in our case, we adopted a 0–1 function) for the obtained class value y_k and the expected $\hat{y}_k$.

$$P_{w'}(t_i) = 1 - \frac{1}{w'} \sum_{k=i-w'+1}^{i} L(y_k, \hat{y}_k) \tag{22}$$

Processing time is measured as the time that the algorithms spends processing in seconds, and memory usage is presented in RAM-Hours, where 1 RAM-Hour equals 1 GB of RAM being used for one hour.

All experimental results presented in this paper were obtained on a Intel Xeon CPU E5649 @ 2.53GHz × 8 based computer running CentOS with 16GB of memory and under the Massive Online Analysis (MOA) framework (Bifet et al., 2010b).

Finally, in order to determine whether there is significant statistical difference between algorithms, Wilcoxon's test (Wilcoxon, 1945) or a combination of Friedman's (Friedman, 1937) and Nemenyi's (Nemenyi, 1963) non-parametric hypothesis tests are used, according to the number of evaluated hypotheses.

5.2. Benchmarking existing works

In this section we present the results for the following algorithms: Very Fast Decision Rules (VFDR), Very Fast Decision Tree (VFDT), Hoeffding Adaptive Tree (HAT), Random Rules (RR), Streaming Random Forest (SRF), HEFT-Stream (HEFT) and Streaming Stacking (SS) (all surveyed in Section 4), an 1-Nearest Neighbor algorithm (1NN) and an Updatable Naïve Bayes (NB).

Table 2 presents the average prequential accuracy results obtained during experiments for all the algorithms. We highlight the higher results obtained by HAT and ensemble-based approaches, which outperformed its base learners in most cases. This highlights

Table 2
Average accuracy obtained during experiments.

Average Accuracy (%)											
Experiment	NB	VFDR	VFDT	HAT	1*NN*	RR ($M = 1$)	RR ($M = 2$)	SRF ($M = 1$)	SRF ($M = 2$)	HEFT	SS
RTG-FD(A)	55.43	56.43	55.65	64.19	57.21	65.98	**66.35**	65.56	65.42	59.62	56.73
RTG-FD(G)	55.41	55.86	55.68	63.33	57.14	65.78	**66.37**	65.56	65.21	62.13	56.13
SEA-FD(A)	79.83	78.92	80.82	83.73	75.28	95.57	95.65	98.29	**98.59**	83.02	75.37
SEA-FD(G)	78.92	77.62	80.80	84.16	75.28	93.23	93.32	96.97	**97.39**	83.22	75.21
BG1-FD(A)	69.99	72.67	78.21	**94.04**	76.00	78.50	76.37	78.17	78.15	90.07	92.16
BG1-FD(G)	69.99	70.21	78.25	**93.20**	75.69	78.30	76.49	78.18	78.18	89.59	92.03
BG2-FD(A)	62.02	68.17	66.63	**91.15**	73.20	59.05	61.14	60.07	61.74	88.51	77.79
BG2-FD(G)	61.94	66.41	66.73	**89.12**	72.89	58.45	62.71	59.98	63.15	88.06	76.38
BG3-FD(A)	54.99	54.24	62.43	86.04	62.96	59.18	60.82	59.91	61.05	**86.78**	53.37
BG3-FD(G)	54.74	54.07	60.70	81.24	62.93	58.19	60.65	59.78	61.40	**86.00**	57.38
Spam Corpus	71.13	74.81	79.32	**84.48**	79.85	75.22	–	74.17	–	82.65	–

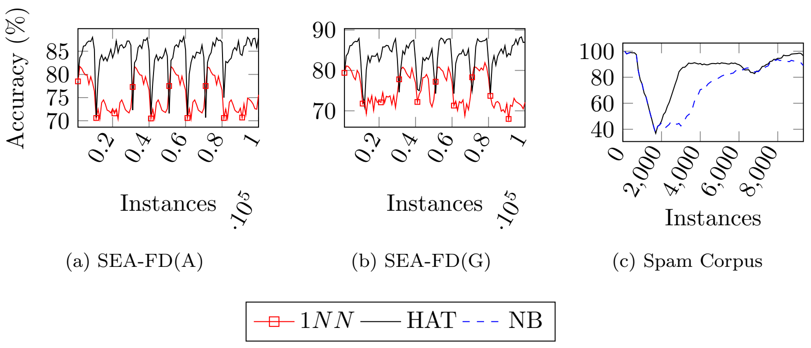


Fig. 4. Accuracy of the best and worst algorithms as the stream progresses.

Table 3
Processing time obtained during experiments.

Processing time (*s*)											
Experiment	NB	VFDR	VFDT	HAT	1*NN*	RR ($M = 1$)	RR ($M = 2$)	SRF ($M = 1$)	SRF ($M = 2$)	HEFT	SS
RTG-FD(A)	**1.94**	85.21	6.66	82.50	7.49	316.36	95.97	10.14	104.68	13.23	31.74
RTG-FD(G)	**1.62**	94.20	6.07	81.24	7.59	351.77	93.79	51.65	847.12	12.93	31.03
SEA-FD(A)	**2.07**	424.62	4.36	4.76	110.43	57.52	84.82	63.60	8598.08	29.01	30.61
SEA-FD(G)	**2.07**	192.48	4.68	3.61	110.38	58.72	79.43	65.10	8594.82	34.74	30.98
BG1-FD(A)	**1.45**	5.50	4.61	104.44	2.66	3700.78	112.59	45.47	576.54	5.20	27.75
BG1-FD(G)	**1.43**	5.54	4.44	103.24	2.86	3627.28	112.54	45.98	1036.78	5.47	27.82
BG2-FD(A)	**1.59**	7.55	4.04	104.88	3.18	2589.64	86.90	25.58	290.87	6.07	27.52
BG2-FD(G)	**1.37**	6.57	3.98	102.78	3.84	3756.22	89.38	49.93	1134.00	6.00	26.73
BG3-FD(A)	**1.46**	5.35	3.37	104.43	3.40	2330.19	86.94	25.52	295.29	5.89	27.88
BG3-FD(G)	**1.33**	5.75	3.67	102.54	4.09	3730.27	87.28	49.42	1145.98	5.74	26.81
Spam Corpus	617.66	691.51	695.06	**145.23**	6329.34	369.99	–	487.75	–	2132.02	–

the power of HAT and ensemble-based algorithms to perform feature drift detection, either via combinatorics, randomness or windowing.

In Fig. 4 we present the accuracy of the best and worst ranked algorithms during the SEA-FD(A), SEA-FD(G) and Spam Corpus experiments, where one can see that HAT is able to quickly recover from feature drifts and boosts accuracy even after them. Specifically in Fig. 4c, one can see that HAT is capable of detecting the feature drift, therefore quickly adapting to it while the Updatable Naïve Bayes (NB) slowly recovers only after half the experiment.

Tabs. 3 and 4 present results obtained for processing time and RAM-Hours, respectively, where one can see that ensemble-based algorithms possess higher processing time and memory usage, as expected. We emphasize that both Random Rules and Streaming Random Forest with $M = 2$ were incapable of performing in the Spam Corpus dataset, due to insufficient memory space ($> 16GB$).

In order to determine whether there is significant statistical difference between algorithms' accuracy, processing time and memory usage, we started with Friedman's test, while ignoring RR ($M = 2$) and SRF ($M = 2$) since they do not present accuracy values for all datasets. We divided our comparison in two distinct tests. The first test compares NB, VFDR, VFDT, HAT and 1*NN*, while the second compares ensemble-based algorithms, i.e. RR, SRF, HEFT and SS.

In our first test, Friedman test pointed out that there was a difference between algorithms by adopting a confidence level of 95% in terms of accuracy and the post-hoc Nemenyi test showed that {HAT}≻{1*NN*, VFDT, VFDR, NB} also with a 95% confidence level. The same procedure was repeated for processing time and memory usage, and results show that {NB, VFDT, 1*NN*}≻{HEFT, HAT, SRF ($M = 1$), VFDR, RR ($M = 1$)} for processing time and {NB, 1*NN*, VFDT}≻{VFDR, SRF ($M = 1$), HAT, HEFT, RR ($M = 1$)} in terms of memory space.

In the second evaluation, Friedman's test pointed that {HEFT}≻{RR ($M = 1$), SRF($M = 1$), SS} in terms of accuracy, however, there is no significant statistical between ensemble-based algorithms in processing time and memory usage.

Table 4
RAM-Hours obtained during experiments.

RAM-Hours (GB-Hour)											
Experiment	NB	VFDR	VFDT	HAT	1*NN*	RR ($M=1$)	RR ($M=2$)	SRF ($M=1$)	SRF ($M=2$)	HEFT	SS
RTG-FD(A)	$\mathbf{1.76 \times 10^{-8}}$	4.64×10^{-4}	1.85×10^{-6}	1.05×10^{-5}	1.18×10^{-6}	1.94×10^{-4}	2.90×10^{-5}	5.31×10^{-7}	8.22×10^{-5}	3.16×10^{-6}	1.82×10^{-5}
RTG-FD(G)	$\mathbf{1.62 \times 10^{-8}}$	4.91×10^{-4}	1.81×10^{-6}	1.03×10^{-5}	1.21×10^{-6}	2.28×10^{-4}	3.58×10^{-5}	7.56×10^{-6}	2.29×10^{-3}	3.25×10^{-6}	1.82×10^{-5}
SEA-FD(A)	$\mathbf{1.56 \times 10^{-8}}$	1.04×10^{-2}	6.08×10^{-7}	2.38×10^{-7}	1.36×10^{-5}	9.34×10^{-5}	3.05×10^{-4}	9.21×10^{-5}	1.10×10^{-5}	5.44×10^{-2}	2.11×10^{-5}
SEA-FD(G)	$\mathbf{1.42 \times 10^{-8}}$	2.32×10^{-3}	6.20×10^{-7}	1.42×10^{-7}	1.36×10^{-5}	9.27×10^{-5}	3.68×10^{-4}	9.09×10^{-5}	7.87×10^{-6}	5.44×10^{-2}	2.16×10^{-5}
BG1-FD(A)	$\mathbf{1.21 \times 10^{-8}}$	2.95×10^{-7}	7.36×10^{-7}	1.33×10^{-5}	3.99×10^{-8}	9.08×10^{-3}	3.48×10^{-5}	4.37×10^{-6}	3.86×10^{-4}	8.22×10^{-7}	5.57×10^{-6}
BG1-FD(G)	$\mathbf{1.29 \times 10^{-8}}$	3.15×10^{-7}	7.25×10^{-7}	1.31×10^{-5}	4.71×10^{-8}	8.95×10^{-3}	3.50×10^{-5}	4.51×10^{-6}	9.79×10^{-4}	8.81×10^{-7}	5.60×10^{-6}
BG2-FD(A)	$\mathbf{1.32 \times 10^{-8}}$	6.34×10^{-7}	4.91×10^{-7}	1.33×10^{-5}	6.70×10^{-8}	4.54×10^{-3}	2.10×10^{-5}	1.70×10^{-6}	1.30×10^{-4}	9.94×10^{-7}	5.59×10^{-6}
BG2-FD(G)	$\mathbf{1.23 \times 10^{-8}}$	4.97×10^{-7}	4.45×10^{-7}	1.31×10^{-5}	8.04×10^{-8}	9.16×10^{-3}	2.61×10^{-5}	4.90×10^{-6}	1.07×10^{-3}	9.95×10^{-7}	5.35×10^{-6}
BG3-FD(A)	$\mathbf{1.32 \times 10^{-8}}$	2.54×10^{-7}	2.89×10^{-7}	1.33×10^{-5}	7.57×10^{-8}	3.63×10^{-3}	2.13×10^{-5}	1.72×10^{-6}	1.32×10^{-4}	9.70×10^{-7}	5.68×10^{-6}
BG3-FD(G)	$\mathbf{1.20 \times 10^{-8}}$	2.99×10^{-7}	3.65×10^{-7}	1.30×10^{-5}	1.19×10^{-7}	9.15×10^{-3}	2.53×10^{-5}	4.85×10^{-6}	1.08×10^{-3}	9.53×10^{-7}	5.33×10^{-6}
Spam Corpus	2.34×10^{-2}	5.11×10^{-2}	1.56×10^{-2}	$\mathbf{1.40 \times 10^{-3}}$	5.50×10^{-2}	4.10×10^{-3}	–	4.46×10^{-3}	–	1.80×10^{-1}	–

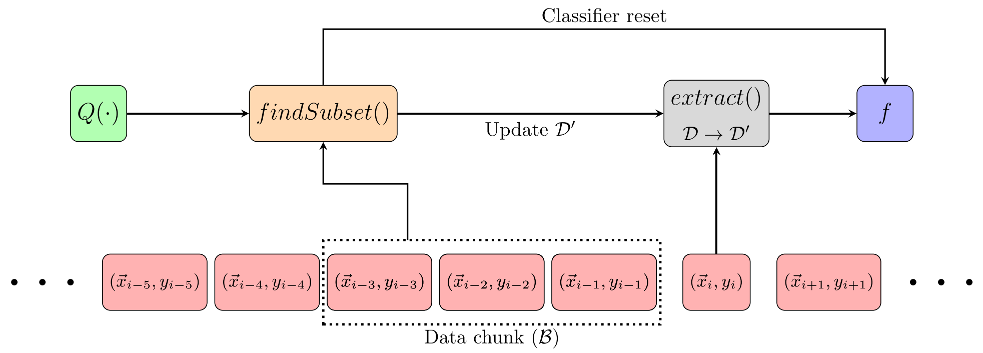

Classifier reset
$Q(\cdot)$
$findSubset()$
Update $\mathcal{D}'$
$extract()$
$\mathcal{D} \to \mathcal{D}'$
f
$(\vec{x}_{i-5}, y_{i-5})$
$(\vec{x}_{i-4}, y_{i-4})$
$(\vec{x}_{i-3}, y_{i-3})$
$(\vec{x}_{i-2}, y_{i-2})$
$(\vec{x}_{i-1}, y_{i-1})$
$(\vec{x}_{i}, y_{i})$
$(\vec{x}_{i+1}, y_{i+1})$
Data chunk $(\mathcal{B})$

Fig. 5. LFDD overview.

5.3. Performing feature selection in data chunks

In this section we propose and empirically evaluate a naive approach to handle feature drifts in data streams. We hypothesize that by splitting a stream into chunks, it is possible to determine the most discriminative subset of features of a stream, and train the classifier exclusively with them. In all cases, the most discriminative subset of features $\mathcal{D}^*$ is assumed as the union of features $D_i \in \mathcal{D}$ that maximize the goodness function $Q(D_i)$ individually. We acknowledge that this discriminative subset selection is naive hence it does not account for redundant features (Yu and Liu, 2003). This occurs due to all features being deemed relevant to the concept (correlated with the class), while they are possibly highly correlated to one another (Kira and Rendell, 1992). Additionally, other studies suggest and empirically show that features with individual low discriminative power, when put together, are able to show interesting discriminative power at times (Nguyen et al., 2012; Guyon, 2003). By performing feature selection as the stream progresses, we assume that a feature drift occurs when the most discriminative subset of features of a chunk of data differs from the subset of features of the previous chunk.

Our proposal is named Landmark-based Feature Drift Detector (LFDD). Its pseudocode is presented in Alg. 1 and an overview is depicted in Fig. 5. It receives as input a data stream $\mathcal{S}$, a base learner e (e.g. NB, VFDT, VFDR and 1*NN*), a landmark window size W, an heuristic goodness function $Q(\cdot)$ (e.g. Entropy, Gain Ratio and Information Gain) and a maximum amount of features d_{max}.

During the training step, instances $(\vec{x}_i, y_i)$ retrieved from $\mathcal{S}$ are stored in an instance buffer $\mathcal{B}$ (lines 3 and 4) and used for training after the extraction of the most discriminative subset of features $\mathcal{D}'$ (line 12).

When the size of the instance buffer reaches W (line 5), then $\mathcal{D}'$ is compared to the new most discriminant subset of features (line 6) given by a function *findSubset*$(\cdot, \cdot, \cdot)$, computed according to the instance buffer $\mathcal{B}$, to the heuristic goodness function $Q(\cdot)$ and the maximum amount of features d_{max} (line 6). If the new subset of discriminant of features $\mathcal{D}_{new}$ differs from the subset of the last chunk of data (line 7), we hypothesize that a feature drift has occurred, so the expert e is reset and $\mathcal{D}'$ is replaced with $\mathcal{D}_{new}$ (lines 8 and 9).

Algorithm 1: Landmark-based feature drift detector (LFDD) pseudocode.

```
   input : a data stream $\mathcal{S}$, a base learner $e$, a landmark
           window size $W$, a goodness function $Q(\cdot)$ and a
           maximum amount of features $d_{max}$.
   /*       TRAINING STEP                                    */
   /* buffer of instances                                    */
1  $\mathcal{B} \leftarrow \emptyset$;
   /* adopted set of features                                */
2  $\mathcal{D}' \leftarrow$ random subset of $\mathcal{D}$ subject to $d_{max}$;
3  foreach $(\vec{x}_i, y_i) \in \mathcal{S}$ do
4      $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\vec{x}_i, y_i)\}$;
5      if $|\mathcal{B}| = W$ then
6          $\mathcal{D}_{new} \leftarrow findSubset(\mathcal{B}, Q, d_{max})$;
7          if $\mathcal{D}_{new} \neq \mathcal{D}'$ then
8              $\mathcal{D}' \leftarrow \mathcal{D}_{new}$;
9              $e.reset()$;
10         $\mathcal{B} \leftarrow \emptyset$;
11     $\vec{x}_i' \leftarrow extract(\vec{x}_i, \mathcal{D}')$;
12     $e.train(\vec{x}_i')$;
   /*       EVALUATION STEP                                  */
13 $\vec{x}_i' \leftarrow extract(\vec{x}_i, \mathcal{D}')$;
14 return $e.evaluate(\vec{x}_i')$
```

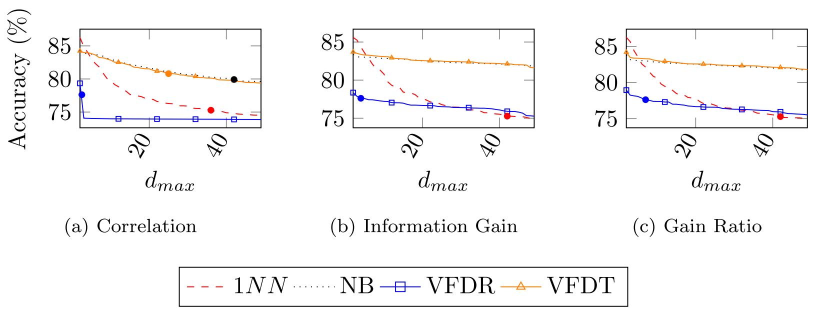


Fig. 6. Accuracy obtained by LFDD during the SEA-FD(G) experiment.

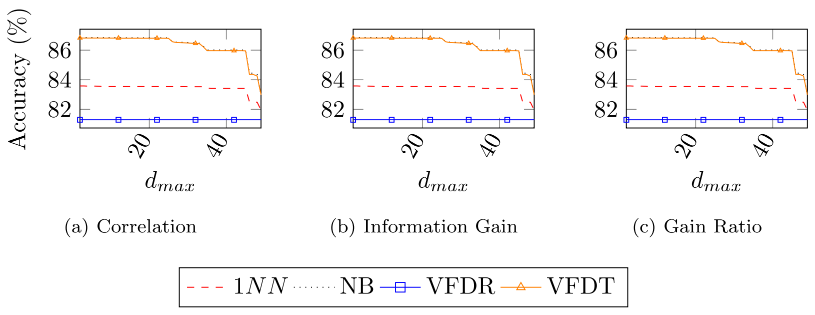


Fig. 7. Accuracy obtained by LFDD during the Spam Corpus experiment.

During the evaluation step, all instances are first translated into the reduced feature set $\mathcal{D}'$ (line 13) and then the base learner is asked for a class label (line 14).

5.3.1. Benchmarking LFDD

In this section, we evaluate the usage of LFDD in the Updatable Naive Bayes (NB), Very Fast Decision Tree (VFDT), Very Fast Decision Rules (VFDR) and a 1-Nearest Neighbor (1*NN*) algorithms. Our goal is to investigate if LFDD is able to improve overall accuracy of classification algorithms in feature drifting streams. We experimented with LFDD while varying the parameter d_{max} in the [2; 49] interval and the following heuristic goodness functions: Correlation, Gain Ratio and Information Gain. We refrain from verifying the impact of the landmark window size W since it represents a trade-off without a clear unique solution due to the plasticity-stability dilemma. A small size results in a window that reflects the current distribution of data and enables quicker drift adaptation (plasticity), while a large size enables a larger amount of data to work on, important in non-drifting periods of the stream (stability) (Bifet, 2010). Therefore, the size of the landmark window was empirically set to $W = 1,000$ for synthetic experiments and $W = 100$ for the Spam Corpus dataset.

In Figs. 6 and 7 we present the accuracy obtained by LFDD when varying the base learner, heuristic goodness function $Q(\cdot)$ and d_{max} in the SEA-FD(G) and Spam Corpus experiments, respectively. We do not provide the graphical results for other experiments since they follow the same behavior as those discussed in this section. In these plots, we mark with a dot the base learner behavior, i.e. the base learner behavior learning with all original features. Cases where no dots appear for a given learner indicate that LFDD outperformed the base learner in all cases, independently of d_{max}.

In Fig. 6 it is possible to verify that LFDD is able to boost all base learners by adopting any goodness function and for a wide range of values of d_{max}. For instance, a conventional NB, when combined with LFDD, is able to improve over its default setting by selecting between 2 and 42 features. In other cases, such as presented in Fig. 6b and c, it is possible to see that, Information Gain and Gain Ratio, when applied to both NB and VFDT are superior to the conventional base learner setting in all cases, i.e. when $d_{max} \in [2; 49]$. This highlights that both the original VFDT and NB are incapable of quickly adapting to feature drifts and the usage of LFDD helps in feature drift detection and model adaptation.

In Figs. 8 and 9 we present the processing time results for the same experiments, where one can see that processing time increases according to the growth in the dimensionality adopted d_{max}. First, it is important to notice that both NB and VFDT, when combined with LFDD, do not present significant improvements in processing time. This occurs due to the simple learning scheme adopted by both algorithms: the first (NB) works with a simple contingency table, which requires a $\mathcal{O}(d_{max})$ to classify each instance; while the second (VFDT) requires at maximum $\mathcal{O}(\log_2 d_{max})$. Although there is no interesting gain being achieved for processing time by adopting LFDD, we recall that there is a gain in accuracy (Figs. 6 and 7).

On the other hand, two other cases are the opposite: 1*NN* and VFDR. To classify each instance a 1*NN* classifier acts in $\mathcal{O}(N \times d_{max})$, therefore, the value of d_{max} highly impacts processing time. Conversely, VFDR exhibits an increased processing time due to the rule set, that grows according to d_{max}. Again, we mark the base learner default behavior, showing that in most cases, working on a reduced dimensionality up to a given threshold results in less processing time for synthetic experiments. We highlight the Spam Corpus experiment, where are all base learners, when associated with LFDD and any heuristic goodness measures, resulted in less processing time when compared to their default configuration, i.e. no feature selection.

Figs. 10 and 11 present RAM-Hours results for SEA-FD(G) and Spam Corpus experiments where one can see that with the increase of d_{max}, algorithms increase their memory usage as well. Again, we highlight both 1*NN* and VFDR, since the first stores in memory a buffer with $\mathcal{O}(N \times d_{max})$ space and the second has its rule set growing exponentially with d_{max}.

We emphasize that when comparing accuracy, processing time, and memory usage, both information gain and gain ratio have presented interesting results. While heuristic goodness functions were

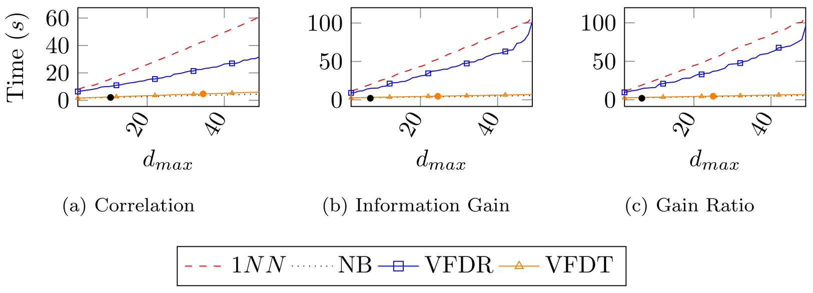


Fig. 8. Processing time (s) obtained during the SEA-FD(G) experiment.

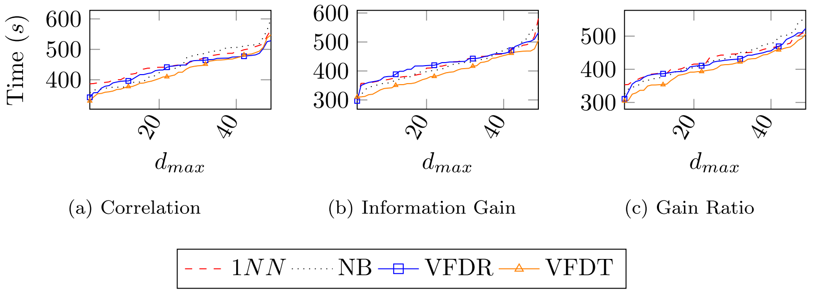


Fig. 9. Processing time (s) obtained by LFDD during the Spam Corpus experiment.

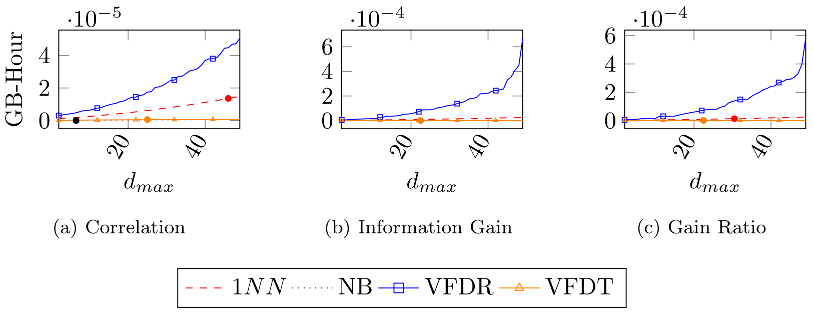


Fig. 10. RAM-Hours obtained during the SEA-FD(G) experiment.

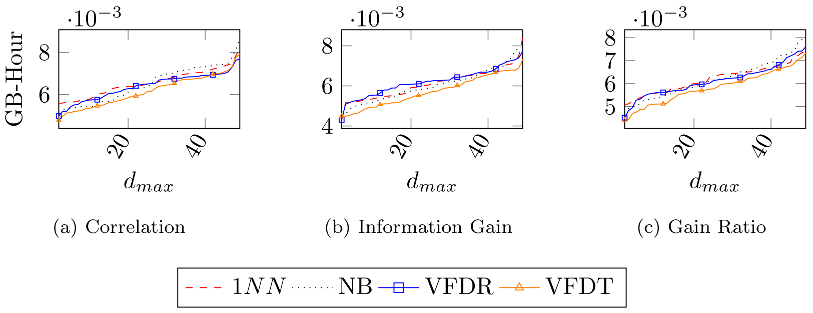


Fig. 11. RAM-Hours obtained by LFDD during the Spam Corpus experiment.

able to boost LFDD's base learner's accuracy, the overhead of selecting features and converting arriving instances into this reduced dimensionality also provided decreases in both processing time and memory space usage when compared to the default configuration of each base learner. This empirically shows that feature selection, even for data streams, is able to provide machine learning models with higher accuracy, for less processing time and memory space (Naidu et al., 2014).

Finally, we present the results obtained by the adoption of LFDD in all experiments in accuracy (Table 5), processing time (Table 6) and memory space (Table 7). Our intent was to perform a pessimistic evaluation of LFDD, therefore, the results presented in the latter cited Tables reflect the lowest accuracy obtained by LFDD regardless of the goodness function adopted, and highest processing time and memory usage. In order to assess whether the usage of LFDD presents significant differences when compared to an isolated base classifier, we performed several paired Wilcoxon's tests.

In terms of accuracy, Wilcoxon's test pointed out that LFDD outperforms base classifiers in all cases by assuming a 95% confidence level. This fact shows that a simple landmark-based filter is able to produce interesting feature drift adaptation when compared to conventional learners. It is also important to emphasize that both VFDT and VFDR had their results boosted, therefore highlighting

Table 5
Accuracy obtained by algorithms with and without LFDD.

Accuracy (%)								
Experiment	NB	LFDD-NB	VFDT	LFDD-VFDT	VFDR	LFDD-VFDR	1*NN*	LFDD-1*NN*
RTG-FD(A)	55.43	**84.90**	55.65	**84.76**	**56.43**	55.42	**57.21**	56.43
RTG-FD(G)	55.41	**84.19**	55.68	**84.23**	**55.86**	55.37	**57.14**	56.24
SEA-FD(A)	79.83	**79.96**	80.82	**86.82**	**78.92**	78.57	75.28	**86.31**
SEA-FD(G)	78.92	**79.35**	80.80	**84.23**	77.62	**79.20**	75.28	**86.27**
BG1-FD(A)	69.99	**94.06**	78.21	**93.81**	72.67	**80.11**	86.00	**88.98**
BG1-FD(G)	69.99	**93.03**	78.25	**92.81**	70.21	**79.37**	85.69	**89.01**
BG2-FD(A)	62.02	**81.44**	66.63	**86.88**	68.17	**74.18**	73.20	**87.88**
BG2-FD(G)	61.94	**76.32**	66.73	**86.17**	66.41	**73.85**	72.89	**86.72**
BG3-FD(A)	54.99	**57.73**	62.43	**73.92**	54.24	**57.40**	65.96	**83.46**
BG3-FD(G)	54.74	**56.74**	60.70	**69.92**	54.07	**57.70**	65.93	**80.68**
Spam Corpus	86.64	**86.85**	86.47	**86.83**	74.81	**81.30**	79.85	**83.58**

Table 6
Processing time obtained by algorithms with and without LFDD.

Processing time (*s*)								
Experiment	NB	LFDD-NB	VFDT	LFDD-VFDT	VFDR	LFDD-VFDR	1*NN*	LFDD-1*NN*
RTG-FD(A)	1.94	**1.77**	6.66	**6.53**	85.21	**80.95**	7.49	**6.59**
RTG-FD(G)	1.62	**1.47**	6.07	**5.77**	94.20	**85.72**	7.59	**6.53**
SEA-FD(A)	**1.93**	2.07	4.36	**4.23**	424.62	**399.14**	110.43	**7.84**
SEA-FD(G)	**1.97**	2.01	4.68	**4.49**	192.48	**182.86**	110.38	**7.95**
BG1-FD(A)	1.45	**1.31**	4.61	**4.52**	5.50	**5.00**	2.66	**2.26**
BG1-FD(G)	1.43	**1.32**	4.44	**4.26**	5.54	**5.21**	2.86	**2.52**
BG2-FD(A)	1.59	**1.51**	4.04	**4.00**	7.55	**6.87**	3.18	**2.86**
BG2-FD(G)	1.37	**1.29**	3.98	**3.86**	6.57	**6.18**	3.84	**3.42**
BG3-FD(A)	1.46	**1.36**	3.37	**3.24**	5.35	**5.08**	3.40	**2.92**
BG3-FD(G)	1.33	**1.21**	3.67	**3.63**	5.75	**5.23**	4.09	**3.48**
Spam Corpus	617.66	**586.78**	695.06	**667.26**	691.51	**636.19**	6329.34	**5506.53**

Table 7
RAM-Hours obtained by algorithms with and without LFDD.

RAM-Hours (GB-Hour)								
Experiment	NB	LFDD-NB	VFDT	LFDD-VFDT	VFDR	LFDD-VFDR	1*NN*	LFDD-1*NN*
RTG-FD(A)	1.76×10^{-8}	**1.64×10^{-8}**	1.85×10^{-6}	**1.74×10^{-6}**	4.64×10^{-4}	**4.59×10^{-4}**	1.18×10^{-6}	**1.16×10^{-6}**
RTG-FD(G)	1.62×10^{-8}	**1.51×10^{-8}**	1.81×10^{-6}	**1.68×10^{-6}**	4.91×10^{-4}	**4.86×10^{-4}**	1.21×10^{-6}	**1.19×10^{-6}**
SEA-FD(A)	1.56×10^{-8}	**1.47×10^{-8}**	6.08×10^{-7}	**5.72×10^{-7}**	1.04×10^{-2}	**1.02×10^{-2}**	1.36×10^{-5}	**1.35×10^{-5}**
SEA-FD(G)	1.42×10^{-8}	**1.32×10^{-8}**	6.20×10^{-7}	**5.83×10^{-7}**	2.32×10^{-3}	**2.23×10^{-3}**	1.36×10^{-5}	**1.35×10^{-5}**
BG1-FD(A)	1.21×10^{-8}	**1.13×10^{-8}**	7.36×10^{-7}	**6.84×10^{-7}**	2.95×10^{-7}	**2.80×10^{-7}**	3.99×10^{-8}	**3.91×10^{-8}**
BG1-FD(G)	1.29×10^{-8}	**1.20×10^{-8}**	7.25×10^{-7}	**6.74×10^{-7}**	3.15×10^{-7}	**3.09×10^{-7}**	4.71×10^{-8}	**4.62×10^{-8}**
BG2-FD(A)	1.32×10^{-8}	**1.23×10^{-8}**	4.91×10^{-7}	**4.62×10^{-7}**	6.34×10^{-7}	**6.28×10^{-7}**	6.70×10^{-8}	**6.63×10^{-8}**
BG2-FD(G)	1.23×10^{-8}	**1.16×10^{-8}**	4.45×10^{-7}	**4.14×10^{-7}**	4.97×10^{-7}	**4.92×10^{-7}**	8.04×10^{-8}	**7.88×10^{-8}**
BG3-FD(A)	1.32×10^{-8}	**1.24×10^{-8}**	2.89×10^{-7}	**2.69×10^{-7}**	2.54×10^{-7}	**2.51×10^{-7}**	7.57×10^{-8}	**7.42×10^{-8}**
BG3-FD(G)	1.20×10^{-8}	**1.13×10^{-8}**	3.65×10^{-7}	**3.43×10^{-7}**	2.99×10^{-7}	**2.84×10^{-7}**	1.19×10^{-7}	**1.17×10^{-7}**
Spam Corpus	2.34×10^{-2}	**2.18×10^{-2}**	1.56×10^{-2}	**1.45×10^{-2}**	5.11×10^{-2}	**4.96×10^{-2}**	5.50×10^{-2}	**5.45×10^{-2}**

the fact that their models are unable to adapt to feature drift more promptly.

In terms of processing time and memory usage, one can see that the adoption of LFDD when compared to the original learners, allows for faster computation and lower memory consumption in all cases, results corroborated by Wilcoxon's test. By combining results across all three aspects, one can see that performing periodical evaluations of features' discriminative power consistently leads to smaller subsets of features for classifiers to work with. As in batch learning, we showed that feature selection is beneficial since it allows learners, on average, to obtain higher accuracy, while reducing both processing time and memory usage.

6. Research challenges and future directions

Determining the most discriminative subset of features as a data stream progresses is not straightforward. In this paper, we presented and benchmarked existing works that perform feature drift adaptation in both explicit and implicit fashions. This survey shows that feature drift is another challenging trait of streaming scenarios that must be accounted for by new stream learning algorithms. Through our naive proposal, namely LFDD, we showed that it is possible to perform feature selection as the stream progresses and that this allows for quicker feature drift recovery and reduces overall processing time and memory usage. Nevertheless, there exists a number of research questions that are still unanswered and pose challenges for the streaming research community.

Inductive tree learning is one of the most commonly used approaches for classifying data streams. As discussed in Section 4, very few decision trees regard the possibility of changes in the underlying distribution of data, and therefore introduce some kind of pruning strategy into the tree evolution. With rare exceptions, existing strategies are based on equal-sized windowing techniques, where the algorithm verifies if the attributes used in split nodes are still maximizing a goodness function $Q(\cdot)$, or if they should be replaced by more appropriate splits.

The same can be said for decision rule learning. Algorithms like Facil and VFDR do not encompass strategies for adapting its model to drifts in data, therefore they must be accompanied by drift detectors (e.g. ADWIN Bifet and GavaldÈâÃá, 2007 and Page-Hinkley's test Mouss et al., 2004) that periodically reset the entire rule set according to error rates of the classifier.

Via randomness and combinatorics, the latter approaches can be combined into ensembles to boost accuracy and to allow for implicit drift adaptation. Nevertheless, training and maintaining an ensemble is not only computationally costly, but it must also employ specific diversity induction and voting schemes.

By randomness, Streaming Random Forests and Random Rules create ensembles and each of its experts are associated with a random subset of features $\mathcal{D}'$. Arriving instances are then used to train experts after their conversion to $\mathcal{D}'$. Due to randomness, it is necessary that experts are allocated with $\mathcal{D}'$ that cover diverse areas of the feature subsets space. The assumption is that at least one of the experts is associated with a useful $\mathcal{D}' \supseteq \mathcal{D}^*$. By associating each expert with a dynamic weight that grows and shrinks accordingly to correct and misclassified instances, the ensemble implicitly adapts to feature drifts since experts with the most discriminative subsets will present higher accuracy rates.

Analogously, the same algorithms can form ensembles by exploring combinatorics. Assuming a feature set $\mathcal{D}$ with $|\mathcal{D}| = M$, it is necessary to create an ensemble with $\sum_{i=1}^{M} \binom{M}{i}$ experts. Again, by associating each expert with a subset $\mathcal{D}'$ and a dynamic weight, the one with $\mathcal{D}' = \mathcal{D}^*$ will present higher accuracy rates and will outvote other experts in predictions. Nevertheless, by exploring combinatorics the size of the ensemble becomes intractable as the size of the experts grows very quickly with M.

Finally, approaches like CVFDT (Hulten et al., 2001), HEFT-Stream (Nguyen et al., 2012) and LFDD assume that the most discriminative subset of features can be computed by filters on disjoint chunks of instances. These algorithms have outperformed others in experiments, however, their major limitation is how to determine the size of these windows, which directly affects the learning process. Small windows allow for quicker recognition of possible changes in the chosen subset of features, however, this approach may lead to the detection of false changes if the stream is noisy. Conversely, bigger windows enable a larger amount of data to work on yet fail to quickly detect changes in the most discriminative subset.

Another open question regards how each classifier deals with changes in this chosen discriminative subset. For example, if a change is detected in a decision tree or decision rule learning algorithm, it is possible to adapt the model learned in order to avoid full model reset, e.g. Hoeffding Adaptive Tree (Bifet and Gavaldà, 2009), however, the same might not hold for other types of learners.

Therefore, open research topics include the development of techniques that constantly verify the relevance of features as new instances arrive in an adaptive and incremental fashion. Performing such verification as data arrives, and independently of window sizes and base classifiers is important, since it allows for faster recognition of feature drifts and improves a classifier's overall accuracy and processing time.

7. Conclusion

This paper presented, formalized and exemplified one rarely addressed characteristic of data streams: feature drifts. Additionally, we surveyed and benchmarked algorithms that perform feature selection during stream learning in both explicit and implicit fashions. Results obtained highlight that feature drift is another challenging trait of data streams that must be accounted for by new stream learning algorithms.

As a conclusion, we believe that performing dynamic feature selection in data streams has not received proper attention in the current research scenario. Studying how to perform dynamic feature selection as streams progress enables algorithms to work only with the most relevant features by discarding irrelevant ones. Throughout simple experiments based on a naive proposal, we showed that a classifier's accuracy can be boosted in feature drifting data, while reducing both processing time and memory space. We hope that the results presented here will motivate more research into developing incremental and adaptive feature selection for data streams.

References

Abdulsalam, H., Skillicorn, D., Martin, P., 2007. Streaming random forests. In: Proceedings of the 11th International Database Engineering and Applications Symposium, (IDEAS 2007), pp. 225–232. doi:10.1109/IDEAS.2007.4318108.

Abdulsalam, H., Skillicorn, D.B., Martin, P., 2011. Classification using streaming random forests. IEEE Trans. Knowl. Data Eng. 23 (1), 22–36. doi:10.1109/TKDE.2010.36.

Aggarwal, C.C., 2014. An introduction to data classification. In: Data Classification: Algorithms and Applications. CRC Press, Taylor & Francis Group, pp. 1–36.

Aggarwal, C.C., Han, J., Wang, J., Yu, P.S., 2003. A framework for clustering evolving data streams. In: Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29. VLDB Endowment, pp. 81–92.

Aggarwal, C.C., Hinneburg, A., Keim, D.A., 2001. On the surprising behavior of distance metrics in high dimensional space. In: Van den Bussche, J., Vianu, V. (Eds.), Database Theory ICDT 2001. In: Lecture Notes in Computer Science, vol. 1973. Springer Berlin Heidelberg, pp. 420–434. 1.1007/3-540-44503-X_27

Almeida, E., Kosina, P., Gama, J., 2013. Random rules from data streams. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18–22, 2013, pp. 813–814. doi:10.1145/2480362.2480518.

Amini, A., Wah, T.Y., 2014. On density-based data streams clustering algorithms: a survey. J. Comput. Sci. Technol. 29 (1), 116–141. doi:10.1007/s11390-014-1416-y.

Barddal, J.P., Gomes, H.M., Enembreck, F., 2014. Sfnclassifier: a scale-free social network method to handle concept drift. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing. ACM, New York, NY, USA, pp. 786–791. doi:10.1145/2554850.2554855.

Barddal, J.P., Gomes, H.M., Enembreck, F., 2015a. Analyzing the impact of feature drifts in streaming learning. In: Proceedings of the 22nd International Conference, ICONIP 2015, Istanbul, Turkey, November 9-12, 2015, Proceedings, Part I. Springer International Publishing, Cham, pp. 21–28. doi:10.1007/978-3-319-26532-2_3. Neural Information Processing

Barddal, J.P., Gomes, H.M., Enembreck, F., 2015b. Sncstream: a social network-based data stream clustering algorithm. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing. ACM, New York, NY, USA, pp. 935–940. doi:10.1145/2695664.2695674.

Barddal, J.P., Gomes, H.M., Enembreck, F., 2015c. A survey on feature drift adaptation. In: Proceedings of 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 1053–1060. doi:10.1109/ICTAI.2015.150.

Bifet, A., 2010. Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams. Frontiers in Artificial Intelligence and Applications, Vol. 207. IOS Press.

Bifet, A., Frank, E., Holmes, G., Pfahringer, B., 2010a. Accurate ensembles for data streams: Combining restricted hoeffding trees using stacking. In: Proceedings of the 2nd Asian Conference on Machine Learning, ACML 2010, Tokyo, Japan, November 8-10, 2010. JMLR.org, pp. 225–240.

Bifet, A., Gavaldà, R., 2007. Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM International Conference on Data Mining, pp. 443–448. doi:10.1137/1.9781611972771.42.

Bifet, A., Gavaldà, R., 2009. Adaptive learning from evolving data streams. In: Proceedings of 8th International Symposium on Intelligent Data Analysis, IDA 2009, Lyon, France, August 31, - September 2, 2009. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 249–260. doi:10.1007/978-3-642-03915-7_22. Advances in Intelligent Data Analysis VIII

Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B., 2010b. MOA: Massive online analysis. J. Mach. Learn. Res. 11, 1601–1604.

Bifet, A., Holmes, G., Pfahringer, B., 2010c. Leveraging bagging for evolving data streams. In: Balcazar, J.L., Bonchi, F., Gionis, A., Sebag, M. (Eds.), Machine Learning and Knowledge Discovery in Databases. In: Lecture Notes in Computer Science, Vol. 6321. Springer Berlin Heidelberg, pp. 135–150. doi:10.1007/978-3-642-15880-3_15.

Bifet, A., Holmes, G., Pfahringer, B., Gavaldà, R., 2009. Improving adaptive bagging methods for evolving data streams. In: Zhou, Z.-H., Washio, T. (Eds.), Advances in Machine Learning. In: Lecture Notes in Computer Science, Vol. 5828. Springer Berlin Heidelberg, pp. 23–37. doi:10.1007/978-3-642-05224-8_4.

Bifet, A., Pfahringer, B., Read, J., Holmes, G., 2013a. Efficient data stream classification via probabilistic adaptive windows. In: SAC, pp. 801–806.

Bifet, A., Read, J., Zliobaite, I., Pfahringer, B., Holmes, G., 2013b. Pitfalls in benchmarking data stream classification and how to avoid them. In: ECML/PKDD (1), pp. 465–479.

Breiman, L., 1996. Bagging predictors. Mach. Learn. 24 (2), 123–140. doi:10.1023/A:1018054314350.

Breiman, L., 2001. Random forests. Machine Learn. 45 (1), 5–32. doi:10.1023/A:1010933404324.

Cao, F., Ester, M., Qian, W., Zhou, A., 2006. Density-based clustering over an evolving data stream with noise. In: SDM, pp. 328–339.

Carvalho, V.R., Cohen, W.W., 2006. Single-pass online learning: performance, voting schemes and online feature selection. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York, NY, USA, pp. 548–553. doi:10.1145/1150402.1150466.

Covões, T.F., Hruschka, E.R., de Castro, L.N., Santos, A.M., 2009. A cluster-based feature selection approach. In: Corchado, E., Wu, X., Oja, E., Herrero, A., Baruque, B. (Eds.), Hybrid Artificial Intelligence Systems. In: Lecture Notes in Computer Science, Vol. 5572. Springer Berlin Heidelberg, pp. 169–176. doi:10.1007/978-3-642-02319-4_20.

Dietterich, T.G., 2000. Ensemble methods in machine learning. In: Proceedings of the First International Workshop on Multiple Classifier Systems. Springer-Verlag, London, UK, UK, pp. 1–15.

Domingos, P., 2012. A few useful things to know about machine learning. Commun. ACM 55 (10), 78–87. doi:10.1145/2347736.2347755.

Domingos, P., Hulten, G., 2000. Mining high-speed data streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York, NY, USA, pp. 71–80. doi:10.1145/347090.347107.

Duangsoithong, R., Windeatt, T., 2009. Relevant and redundant feature analysis with ensemble classification. In: Proceedings of Seventh International Conference on Advances in Pattern Recognition, 2009. ICAPR '09., pp. 247–250. doi:10.1109/ICAPR.2009.36.

Ferrer-Troyano, F.J., Aguilar-Ruiz, J.S., Santos, J.C.R., 2005. Incremental rule learning and border examples selection from numerical data streams. J. UCS 11 (8), 1426–1439. doi:10.3217/jucs-011-08-1426.

Friedman, M., 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Stat. Assoc. 32 (200), 675–701. doi:10.2307/2279372.

Gama, J., 2010. Knowledge Discovery from Data Streams, 1st Chapman & Hall/CRC.

Gama, J., Kosina, P., 2011. Learning decision rules from data streams. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011, Barcelona, Catalonia, Spain, July 16–22, 2011, pp. 1255–1260.

Gama, J., Rodrigues, P., 2009. Issues in evaluation of stream learning algorithms. In: Proc. of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM SIGKDD, pp. 329–338.

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

Guyon, I., 2003. An introduction to variable and feature selection. J. Mach. Learn. Res. 3, 1157–1182.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., 2009. The weka data mining software: an update. SIGKDD Explor. Newsl. 11 (1), 10–18. doi:10.1145/1656274.1656278.

Hall, M.A., 2000. Correlation-based feature selection for discrete and numeric class machine learning. In: Proceedings of the Seventeenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 359–366.

Hall, M.A., Smith, L.A., 1999. Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper. In: Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference. AAAI Press, pp. 235–239.

Han, J., Kamber, M., Pei, J., 2011. Data Mining: Concepts and Techniques, 3rd Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Hoeffding, W., 1963. Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. 58 (301), 13–30.

Hoens, T.R., Chawla, N.V., Polikar, R., 2011. Heuristic updatable weighted random subspaces for non-stationary environments. In: Proceedings of IEEE 11th International Conference on Data Mining (ICDM), 2011, pp. 241–250. doi:10.1109/ICDM.2011.75.

Hoi, S.C.H., Wang, J., Zhao, P., Jin, R., 2012. Online feature selection for mining big data. In: Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications. ACM, New York, NY, USA, pp. 93–100. doi:10.1145/2351316.2351329.

Huang, D.T.J., Koh, Y.S., Dobbie, G., Bifet, A., 2015. Drift detection using stream volatility. In: Proceedings of European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, , Part I. Springer International Publishing, pp. 417–432. doi:10.1007/978-3-319-23528-8_26. Machine learning and knowledge discovery in databases

Hulten, G., Spencer, L., Domingos, P., 2001. Mining time-changing data streams. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York, NY, USA, pp. 97–106. doi:10.1145/502512.502529.

Katakis, I., Tsoumakas, G., Vlahavas, I., 2006. Dynamic feature space and incremental feature selection for the classification of textual data streams. In: Proceedings of ECML/PKDD-2006 International Workshop on Knowledge Discovery from Data Streams. 2006. Springer Verlag, p. 107.

Kira, K., Rendell, L.A., 1992. The feature selection problem: traditional methods and a new algorithm. In: Proceedings of the Tenth National Conference on Artificial Intelligence. AAAI Press, pp. 129–134.

Kohavi, R., John, G.H., 1997. Wrappers for feature subset selection. Artif. Intell. 97 (1–2), 273–324. http://dx.doi.org/10.1016/S0004-3702(97)00043-X. Relevance

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

Kosina, P., Gama, J., 2012. Very fast decision rules for multi-class problems. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing. ACM, New York, NY, USA, pp. 795–800. doi:10.1145/2245276.2245431.

Kranen, P., Assent, I., Baldauf, C., Seidl, T., 2011. The clustree: Indexing micro-clusters for anytime stream mining. Knowl. Inf. Syst. 29 (2), 249–272. doi:10.1007/s10115-010-0342-8.

Kuncheva, L.I., Faithfull, W.J., 2014. PCA feature extraction for change detection in multidimensional unlabelled data. IEEE Trans. Neural Netw. Learn. Syst. 25 (1), 69–80. doi:10.1109/TNNLS.2013.2248094.

Li, H.-G., Wu, X., Li, Z., Ding, W., 2013. Online group feature selection from feature streams. In: des Jardins, M., Littman, M.L. (Eds.), AAAI. AAAI Press, pp. 1627–1628.

Mouss, H., Mouss, D., Mouss, N., Sefouhi, L., 2004. Test of page-hinckley, an approach for fault detection in an agro-alimentary production system. In: Proceedings of 5th Asian Control Conference, 2004., vol. 2, pp. 815–818. Vol.2

Naidu, K., Dhenge, A., Wankhade, K., 2014. Feature selection algorithm for improving the performance of classification: a survey. In: Proceedings of the 2014 Fourth International Conference on Communication Systems and Network Technologies. IEEE Computer Society, Washington, DC, USA, pp. 468–471. doi:10.1109/CSNT.2014.99.

Nemenyi, P., 1963. Distribution-free multiple comparisons, New Jersey, USA Ph.D. thesis.

Nguyen, H.-L., Woon, Y.-K., Ng, W.-K., Wan, L., 2012. Heterogeneous ensemble for feature drifts in data streams. In: Tan, P.-N., Chawla, S., Ho, C., Bailey, J. (Eds.), Advances in Knowledge Discovery and Data Mining. In: Lecture Notes in Computer Science, vol. 7302. Springer Berlin Heidelberg, pp. 1–12. doi:10.1007/978-3-642-30220-6_1.

Oza, N., 2005. Online bagging and boosting. In: Proceedings of 2005 IEEE International Conference on Systems, Man and Cybernetics, Vol. 3, pp. 2340–2345. doi:10.1109/ICSMC.2005.1571498. Vol. 3

Park, C.H., 2013. A feature selection method using hierarchical clustering. In: Prasath, R., Kathirvalavakumar, T. (Eds.), Mining Intelligence and Knowledge Exploration. In: Lecture Notes in Computer Science, vol. 8284. Springer International Publishing, pp. 1–6. doi:10.1007/978-3-319-03844-5_1.

Rudnicki, W.R., Wrzesień, M., Paja, W., 2015. All relevant feature selection methods and applications. In: Stańczyk, U., Jain, L.C. (Eds.), Feature Selection for Data and Pattern Recognition. In: Studies in Computational Intelligence, vol. 584. Springer Berlin Heidelberg, pp. 11–28. doi:10.1007/978-3-662-45620-0_2.

Silva, J.A., Faria, E.R., Barros, R.C., Hruschka, E.R., de. Carvalho, A.C.P.L.F., Gama, J.a., 2013. Data stream clustering: A survey. ACM Comput. Surv. 46 (1), 13:1–13:31. doi:10.1145/2522968.2522981.

Street, W.N., Kim, Y., 2001. A streaming ensemble algorithm (sea) for large-classification. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM SIGKDD, pp. 377–382.

Tsymbal, A., 2004. The Problem of Concept Drift: Definitions and Related Work. Technical Report, TCD-CS-2004-15. The University of Dublin, Trinity College, Department of Computer Science, Dublin, Ireland.

Wilcoxon, F., 1945. Individual comparisons by ranking methods. Biom. Bull. 1 (6), 80–83. doi:10.2307/3001968.

Wolpert, D.H., 1992. Stacked generalization. Neural Netw. 5 (2), 241–259. doi:10.1016/S0893-6080(05)80023-1.

Yu, L., Liu, H., 2003. Feature selection for high-dimensional data: A fast correlation-based filter solution. In: Proceedings of the Twentieth International Conference on Machine Learning. AAAI Press, pp. 856–863.

Zhao, Z., Morstatter, F., Sharma, S., Alelyani, S., Anand, A., Liu, H., 2010. Advancing feature selection research. ASU Feature Sel. Repos. 1–28.

Zhou, J., Foster, D., Stine, R., Ungar, L., 2005. Streaming feature selection using alpha-investing. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining. ACM, New York, NY, USA, pp. 384–393. doi:10.1145/1081870.1081914.

Jean Paul Barddal received the B.Sc. and M.Sc. degrees in Computer Science from Pontifical Catholic University of Paraná (PUC-PR), Curitiba/PR, Brazil, in 2013 and 2015, respectively. He is currently pursuing his Ph.D. in Informatics from Graduate Program in Informatics also from PUC-PR. His research is currently on Feature Drift Detection and Adaptation. He has experience in Data Analysis from Large Databases and Web Development. His research interests include Search Optimization Techniques for Artificial Intelligence, Classification, Regression and Clustering with emphasis in Data Stream Mining, Social Network Analysis, Ensemble Learning and Databases.

Heitor Murilo Gomes received his B.Sc. degree in Computer Science from Universidade Tuiuti do Paraná (UTP), Curitiba/PR, Brazil, in 2010 and his M.Sc. degree in Computer Science from the Pontifícia Universidade Católica do Paraná (PUC-PR), Curitiba/PR, Brazil, in 2012 and is currently pursuing a doctoral degree also from PUC-PR. From 2008 until 2011 he has worked as a computer programmer for several companies. Since 2011, he has been working as a Professor of Computer Science teaching Databases, Artificial Intelligence, Computational Intelligence, and Computer Programming at UTP. His research interests include Data Stream Mining, Social Network Analysis, Classifier Combination and Multi-Agent Systems.

Fabrício Enembreck received his B.Sc. degree in Computer Science from the State University of Ponta Grossa in 1997, M.Sc. in Applied Computer Science from the Pontifical Catholic University of Paraná (PUC-PR) in 1999 and his Ph.D. in Information Technologies and Systems from the Université de Technologie de Compiegne in 2003. Currently he is professor at the Pontifical Catholic University of Paraná (PUC-PR) and researcher at the Graduate Program in Informatics, directing researches of M.Sc. and Ph.D. students. He has experience in Computer Science with emphasis on Architecture of Computing Systems, acting on the following topics: Distributed Artificial Intelligence, Multi-Agent Systems, Adaptive Agents, Information Retrieval and Machine Learning.

Bernhard Pfahringer received his PhD degree from the University of Technology in Vienna, Austria, in 1995. He is a Professor working in the Computer Science Department at the University of Waikato in Hamilton, New Zealand. His research interests span a range of data mining and machine learning sub-fields, with a focus on streaming, randomization, and complex data.