2015 IEEE 27th International Conference on Tools with Artificial Intelligence

A Survey on Feature Drift Adaptation

Jean Paul Barddal, Heitor Murilo Gomes and Fabrício Enembreck
Graduate Program in Informatics (PPGIa)
Pontifícia Universidade Católica do Paraná
Curitiba, Brazil
Email: {jean.barddal, hmgomes, fabricio}@ppgia.pucpr.br

***Abstract*—Mining data streams is of the utmost importance due to its appearance in many real-world situations, such as: sensor networks, stock market analysis and computer networks intrusion detection systems. Data streams are, by definition, potentially unbounded sequences of data that arrive intermittently at rapid rates. Extracting useful knowledge from data streams embeds virtually all problems from conventional data mining with the addition of single-pass real-time processing within limited time and memory space. Additionally, due to its ephemeral nature, it is expected that streams undergo changes in its data distribution denominated concept drifts. In this work, we focus on one specific kind of concept drift that has not been extensively addressed in the literature, namely feature drift. A feature drift happens when changes occur in the set of features, such that a subset of features become, or cease to be, relevant to the learning problem. Specifically, changes in the relevance of features directly imply modifications in the decision boundary to be learned, thus the learner must detect and adapt to according to it. Timely detection and recover from feature drifts is a challenging task that can be modeled after a dynamic feature selection problem. In this paper we survey existing work on dynamic feature selection for data streams that acts either implicitly or explicitly. We conclude that there is a need for future research in this area, which we highlight as future research directions.**

I. Introduction

In the last decades the interest in mining massive and potentially unbounded datasets which arrive at rapid rates, namely data streams, has grown substantially. Mining data streams is of the utmost importance since many available data generators produce enormous amounts of data over time. Examples of data streams include sensor networks, wearable sensors, computer network traffic sniffers and video surveillance, to name a few. 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 [1], [2] and unsupervised [3], [4], [5], [6] tasks.

The most common learning problem in a streaming context is classification. In this problem, instances are associated with labels and the main objective is to learn from labeled data how to determine the label of future instances. Data stream classification algorithms are presented to a great and possibly unbounded amount of data, each of which are made available to the algorithm in a serialized fast-paced fashion [7]. Moreover, due to inherent aspects of data streams, we must assume that the underlying concept is unstable, i.e. changes in the concept to be learned are expected to occur, phenomenon named concept drift [8].

Although current techniques for data stream classification handle most of the challenges posed by the streaming environment, not many attention has been given to possible changes in the relevance of features through time, i.e. feature drifts. This enforces classification algorithms to include strategies to keep track of the most discriminative set of features of the stream by using feature selection methods. The benefits of an accurate feature selection method is that the classifier can process instances faster using less memory space usage, and present higher acuity [9]. These benefits are due to the diminished dimensionality, which requires less resources and maintains only meaningful features for the classifier training. Nevertheless, feature selection is a difficult task and in a streaming environment it becomes even more complicated. In this paper we review the classification task for data streams (Sec. II), and highlight the feature drift problem (Sec. III). Additionally, we survey existing work that performs feature selection during stream processing in both explicit and implicit fashions, highlighting theirs major limitations (Sec. IV). Finally, we state the challenges of this research area and future directions (Sec. VI).

II. 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 discrete classes accordingly to relations or affinities. The classification task can be formalized as follows: a set of n training instances in the form $i = (\vec{x}_i, y_i)$ where y_i is a discrete class label and $\vec{x}_i$ is a d-dimensional vector of attributes belonging to a feature set (dimensions) $\mathcal{D} = \{D_1, \ldots, D_d\}$, which can be categorical, ordinal, numeric or mixed. A classifier produces from this training set a model $f : (\vec{X} \rightarrow Y)$ that is used to classify future unlabeled instances.

Data stream classification or online classification, is a variant of the machine learning task namely batch classification, however, both are concerned with the problem of learning a model which is able to predict a nominal value for future instances. The difference between these two approaches concerns about how data is presented to the learner. In batch configuration, a static and entirely accessible dataset is provided to the learning algorithm, which returns a model to predict future instances. 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 adapt its model as new instances arrive [10], [11].

Let $\mathcal{S} = [i_t]_{t=0}^{\infty}$ represent a data stream providing instances $i_t = (\vec{x}_t, y_t)$, each of which arrives at a timestamp t, where

1082-3409/15 $31.00 © 2015 IEEE
DOI 10.1109/ICTAI.2015.150

$\vec{x}_t$ is a d-dimensional attribute vector belonging to a attribute set $\mathcal{D}$ and y_t is the ground-truth label of $\vec{x}_t$.

Most of traditional existing classification algorithms depend on the existence of a static dataset generated by a unknown, yet stationary, probability distribution. Also, some algorithms require multiple passes over data to obtain an accurate hypothesis. Nonetheless, none of the latter assumptions can be verified in the streaming scenario and the development of algorithms must take into account several constraints [7], [10], [12], [13], [14], [15], [16].

Firstly, instances arrive continuously over time and there is no control over the order that instances arrive nor how they should be processed. Additionally, streams are potentially unbounded, therefore, 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 dynamically change over time, implying in changes in the concept to be learned, phenomenon named concept drift.

Let Eq. 1 denote a concept C, a set of prior probabilities of the classes and class-conditional probability density function [17].

$$C = \{(P[y_1], P[\vec{x}|y_1]), \ldots, (P[y_c], P[\vec{x}|y_c])\} \tag{1}$$

Given a stream $\mathcal{S}$, instances i_t retrieved will be generated by a concept C_t. If during every instant t_i of $\mathcal{S}$ we have $C_{t_i} = C_{t_{i-1}}$, it occurs that the concept is stable. Otherwise, if between any two timestamps t_i and t_j occurs that $C_{t_i} \neq C_{t_j}$, we have a concept drift.

III. Feature Drift

Most of existing algorithms for data streams tackle the infinite length and drifting concept characteristics. However, not much attention has been given to the feature drift problem. Feature drifts occur whenever the relevance of a feature D_i, such that $D_i \in \mathcal{D}$, grows or shrinks for incoming instances. This requires that the learning algorithm adapt its model in a way that relevant features are highlighted, while irrelevant attributes are obfuscated [17].

Given a feature space $\mathcal{D}$ at a timestamp t, we are able to select the ground-truth discriminative subset $\mathcal{D}^*_t \subseteq \mathcal{D}$. A feature drift occurs if, at any two time instants t_i and t_j, $\mathcal{D}^*_{t_i} \neq \mathcal{D}^*_{t_j}$ betides. Let $r(D_i, t_j) \in \{0, 1\}$ denote a function which determines the relevance of a feature D_i in a timestamp t_j of the stream. A positive relevance ($r(D_i, t_i) = 1$) states that $D_i \in \mathcal{D}^*$ in 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. 2.

$$\exists t_j \exists t_k,\ t_j < t_k,\ r(D_i, t_j) \neq r(D_i, t_k) \tag{2}$$

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

As in concept drifts, changes in $r(\cdot, \cdot)$ may occur during the stream. This enforces learning algorithms to detect changes in $\mathcal{D}^*$, discerning between features that became irrelevant and the ones that are now relevant. Finally, it is necessary to either (i) discard and learn an entirely new classification model; or (ii) adapt the current model to these relevance changes [17].

Although feature drifts may occur in a variety of environments, one of the most common is text mining. In order to exemplify a feature drift, we refer to the e-mail spam detection system presented in [18]. This system was a result of a text mining process on an online news dissemination system. Essentially, this work intended on creating 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 created contains 9,324 instances and 39,917 features, such that each attribute represents the presence of a single word (feature) in an instance (e-mail). This dataset is known for containing a feature drift which occurs gradually around the instance of number 1,500 [18], [19] and highly impacts the learner.

In Fig. 1a we present a plot of the information gain (presented in Eq. 3, however, relies on the computation of Entropy stated in Eq. 4) [20] of two specific attributes presented in this problem, namely "directed" and "info", where one can see that the importance of these two attributes start exchanging gradually around instance 1,500.

$$IG(D_i) = H(D_i) - \prod_{D_j \in \mathcal{D}, D_j \neq D_i} \frac{H(D_j)}{n} \tag{3}$$

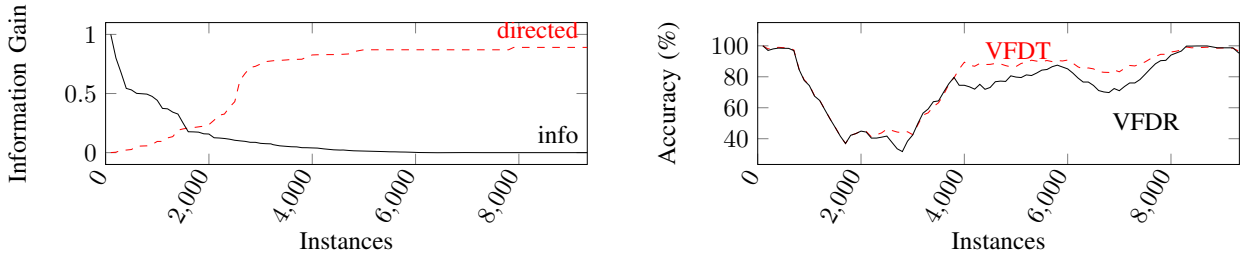
$$H(D_i) = -\sum_{q \in D_i} P[q] \log_2 P[q] \tag{4}$$

Detecting and discerning the two features that exchange relevances 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. 1b) and learning with a subset of the whole feature set of 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) will be discussed in Secs. IV-A and IV-B, respectively.

IV. Existing Works

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 the existing algorithms that perform feature selection as the stream progresses, either assuming the existence or not of feature drifts. We emphasize that most of the surveyed work here presented were not developed aiming at detecting and adapting to feature drifts, however, they do so through randomness or combinatorics; and henceforth are referred as implicit approaches.

In Table I we summarize the existing algorithms. We categorize these algorithms accordingly to four characteristics: it's learning approach, the type of feature selection algorithm,



(a) (b)

Fig. 1: Analysis of information gain for two specific features and accuracy obtained on the Spam Corpus dataset. (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).

feature drift adaptation method adopted; and whether it performs explicit dynamic feature selection or not.

We start this section by discussing two important and widely used approaches for classifying data streams: decision trees (Sec. IV-A) and decision rules (Sec. IV-B). 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 face this problem, either through randomness (Sec. IV-C), combinatorics (Sec. IV-D) or windowing (Sec. IV-E).

A. Decision Trees

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, each branch from a node corresponds to an outcome of the test and each leaf contains a possible prediction (class value from Y) [10].

Predictions for instances $\vec{x}_i$ 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 $D_i \in \mathcal{D}$ accordingly to some heuristic measure (e.g. Gain Ratio, Information Gain and Entropy).

1) Very Fast Decision Tree: The Very Fast Decision Tree (VFDT) constructs decision trees by using constant memory and constant time per sample [21]. 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. 4), Information Gain (Eq. 3) and Gini Coefficient (Eq. 5) [27], where n is the amount of instances in the dataset analyzed.

$$GI(D_i) = 1 - \sum_{q \in D_i} P[q] \tag{5}$$

To do so, VFDT assumes that the input data meets the Hoeffding bound [28]. 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. 6, 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 $\bar{n}$ is the mean computed by the latter observations.

$$\epsilon = \sqrt{\frac{R^2 \ln\left(\frac{1}{\delta}\right)}{2n}} \tag{6}$$

The Hoeffding bound is able to give results regardless 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 [29]. Generally, with probability $(1-\delta)$, one can say that one attribute is superior when compared to others when observed difference of information gain (or any other heuristic goodness metric of an attribute) is greater than ϵ.

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

B. Decision Rules

Although decision trees account for readability, in some specific scenarios, where trees tend to grow largely, they become hard to understand since nodes appear in a specific context defined by tests at antecedent nodes [25]. In contrast, classifiers based on rules have the advantage of both modularity and interpretability [30], where each rule is independent of the others and can be interpreted isolated from others.

A decision rule is a logic predicate in the ***IF*** **antecedent** ***THEN*** **label** form, where the antecedent is a conjunction of conditions over attributes D_i and the label is a possible class value that belongs to Y. Rules can be updated or removed when outdated so the model may evolve naturally in streaming scenarios.

1) Facil: The first streaming rule learner published was Facil [22]. Facil creates rules accordingly to the arrival of instances in 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 $\mathcal{W}$, adopting a sliding window approach to eliminate old rules. Conversely, implicit forgetting occurs when removing rules that are not relevant

TABLE I: 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	–		[21]
Facil	Rules	Purity	–		[22]
VFDR	Rules	Entropy	–		[23]
CVFDT	Tree	Entropy Information Gain Gini Coefficient	Windowing	✓	[24]
Random Rules	Ensemble (Rules)	–	Randomness/Combinatorics		[25]
Streaming Random Forest	Ensemble (Trees)	–	Randomness/Combinatorics		[12], [13]
Streaming Stacking	Ensemble (Trees)	–	Combinatorics		[26]
HEFT-Stream	Ensemble	FCBF	Windowing	✓	[17]

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 which are near to one another at the decision boundaries. 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. One of the major restrictions of Facil is that numeric input data must be normalized in the $[0; 1]$ interval.

2) Very Fast Decision Rules: A more robust approach for learning rules from data streams is proposed in [23]. This algorithm, namely 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 accordingly to the minimization of entropy of class labels of instances covered by each rule. Rules are grown accordingly to the arrival of instances, by selecting attributes that minimize the entropy (Eq. 4) of the class distribution in each data partition. Additionally, VFDR decided whether a rule should be expanded given the Hoeffding bound (Eq. 6).

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 by it. In the latter, labeled instances update statistics of all covering rules. In both cases, if no rules cover an instance, the default rule is updated.

Finally, VFDR encompasses two classification strategies. The first uses only the information about class distributions and does not account for attribute's 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 ($P[y_i|\vec{x}] \propto P[y_i] \prod P[\vec{x}_j|y_i]$).

C. Randomness

Diversity is a trait of a variety of recently proposed algorithms for learning from data streams [31], [32], [33]. In these approaches, ensembles of experts are trained in parallel, and each of which receives different inputs for training [34]. The most well-known approach for inducing diversity in ensembles is Bagging [35]. Originally, a bagging ensemble is composed of m classifiers, which are trained with bootstraps $\vec{X}_j$ of the whole training set $\vec{X}$. Subsets are formed by sampling with replacement from training set $\vec{X}$. However, sampling usually is not feasible in a data stream configuration, since that would require storing all instances before creating subsets. Therefore, authors in [33] 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 boost accuracy of learners, more recent approaches aim at promoting diversity through different feature subsets [12], [13]. By learning through ensembles with different features, experts learn partially (or completely) disjoint areas of the feature space, implying in a highly diverse ensemble. Although these algorithms do not focus explicitly in adapting to feature drifts, they do present implicit adaptation to this characteristic of data streams.

1) Streaming Random Forest: The Streaming Random Forest classifier is an adaptation of the ensemble-based Random Forest classifier [36]. 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 from a bootstrap from the whole 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 [13], [37].

The construction of each tree follows a variant of 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 tree of the ensemble is a randomly chosen subset $\mathcal{D}' \subset \mathcal{D}$, where $|\mathcal{D}'| \leq M$.

Since a random forest is an ensemble classifier, the classification of each new instance is the fusion of the votes of the containing trees. Therefore, the random forest classification error depends on (i) the correlation among its component trees, since smaller correlations imply in 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 smaller is the error rate [12].

The value of M is a sensitive parameter of random forests and must be chosen carefully. Small values of M tend to increase the strength of each individual tree, while decreases the correlation between them [12].

2) Random Rules: In [25], authors extend the VFDR algorithm by promoting randomness. This algorithm, namely Random Rules for Data Stream (RR), encompasses the following parameters: a number of rule sets (N_s) and the number of

attributes M that must respect 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 so, all covering rules are expanded using only the features adopted by the rule set $\mathcal{D}'$. Otherwise, it is, if no rules cover i_t, 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 ($\max_{y_i \in Y} P[y_i|\vec{x}_t]$).

D. Combinatorics

By exploring combinatorics, random forest and random rules algorithms can be extended and posed as dynamic wrappers for dynamic feature selection for data streams. If one assumes a random forest or a random rule algorithm, where each of its containing expert is trained with a different subset from 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 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 [13]. Therefore, by applying weighted majority voting [38], feature drifts can be detected accordingly 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 diminished weights due to lower acuity.

1) Streaming Stacking: In [26], authors produce a classification model based on an ensemble of decision trees, each of which is built from a random and distinct subset of $\mathcal{D}$. 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, their method generates trees in parallel and combines them using perceptron classifiers by applying stacking [39]. Due to the streaming scenario, VFDTs are used as ensemble members since they can be trained incrementally. Additionally, the ensemble adopts the ADWIN change detector [29] 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 large number of features, although certainly not feasible for high-dimensional data (e.g. Spam Corpus).

E. 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 [16]. There are three major windowing techniques in the literature: sliding, damped and landmark; and in all cases, the difficulty is to select their appropriate size. While short windows reflect the current data distribution and ensures fast adaptation to drifts, shorter ones worsen the performance of the system in stable areas. Conversely, larger windows give better performance in stable periods, however, these imply in slower reaction to drifts [40], [11].

Sliding windows store in memory a fixed or variable amount of recent examples. In the fixed approach, whenever a new instance arrives, it is saved in a FIFO (first in, first out) policy data structure, where the oldest one is discarded. In variable-sized windows, the amount of instances in this data structure may change over time, usually accordingly to the outputs of a change detector. A straightforward idea is to shrink the window when changes in data are detected, so that the data stored in memory reflects the current concept.

In opposition to sliding windows, in damped windows, data is associated with time decaying weights [41]. Therefore, more recent instances receive higher weight than older ones, and these weights decay with time accordingly to a decaying function. This windowing technique is interesting because these 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 require processing a stream by handling disjoint chunks 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 [42]. All instances belonging to a same landmark window are stored or summarized into a same data structure, which is used for training. When a new landmark is reached, all data in the current window is discarded and further instances retrieved from the stream are kept until a new landmark is reached.

The problem in using any fixed-length window approach is how to define its size. Small windows guarantee that the stream learning algorithm is able to rapidly adapt to drifts in underlying data, while in long stable phases, its performance may be put in risk [16], [43]. Conversely, large windows are desirable in stable phases, while these may not allow quick adaptation to drifts [43].

In this section we present existing works that rely in windowing approaches to explicitly adapt to feature drifts.

1) Concept-adapting Very Fast Decision Tree: Concept-adapting Very Fast Decision Tree (CVFDT) algorithm is an extension to the VFDT to deal with concept drifts [24]. 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. Whenever a new instance i_t arrives, CFVDT updates the statistics at its nodes by decrementing counters accordingly 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. Accordingly to user-given parameters T_0,

T_1 and T_2, CFVDT traverses all the decision tree and checks at each node if the splitting attribute is still the best when compared to others. If there is an alternate better splitting attribute, the whole subtree is replaced by a new split node with this attribute. Later, 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.

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 [17]. HEFT-Stream adopts an modification of the Fast Correlation-Based Filter (FCBF) algorithm so it updates the selected relevant feature subset of a stream.

FCBF is a feature selection algorithm where the class relevance and the pairwise dependency between features are accounted for. Based on information theory, FCBF adopts symmetrical uncertainty (SU) to compute dependencies of features and class relevance. On a top-bottom approach, it is, 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 accordingly to Eq. 7, where $H(\cdot)$ is the entropy of a feature (shown in Eq. 4), $H(\cdot,\cdot)$ is the conditional entropy and $MI(\cdot,\cdot)$ is the mutual information between two features (stated in Eq. 8).

$$SU(D_i, D_j) = 2\left[\frac{MI(D_i, D_j)}{H(D_i) + H(D_j)}\right] \tag{7}$$

$$MI(D_i, D_j) = \sum_{q \in D_i} \sum_{r \in D_j} P[q, r] \log \frac{P[q, r]}{P[q] \times P[r]} \tag{8}$$

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. In the end of a chunk, this subset is stored and the process is repeated in the following chunk. Whenever two consecutive subsets are different, HEFT-Stream postulates that a feature drift has occurred.

Additionally, in order to boost the ensemble overall accuracy, HEFT-Stream assumes that diversity among member classifiers, therefore, encompasses an online bagging [33] approach to promote it. Originally, a bagging ensemble is composed of base classifier each of which are trained with bootstraps of data (as discussed in Sec. IV-C).

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 (stated in Eq. 9) which is an accumulated error from its creation time to the current time. The weight w_k is stated in Eq. 9 where α is a padding value which was originally empirically set to 0.001 and E_k is the accumulative error of a member classifier k.

$$w_k = \frac{1}{(E_k + \alpha)} \times \left[\sum_{m=1}^{K} (E_m + \alpha)^{-1}\right] \tag{9}$$

Finally, at the end of every chunk, the classifier with the biggest value of E_k is then replaced by a new classifier. This new classifier is then associated with the feature set 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 show results only for a combination of a Updatable Naïve Bayes and VFDT algorithms.

V. Discussion

Determining the most discriminative subset of features of data streams streams is not straightforward. In this paper we presented existing work that performs feature drift adaptation in both explicit and implicit fashions. However, there exists a variety of questions that are still unanswered and pose challenges for the streaming research community.

Inductive tree learning is one of the most commonly used approach for classifying data streams. As discussed in the previous section, very few trees regard the possibility of changes in the underlying distribution of data, therefore introduce some kind of pruning strategy into tree evolution. Nevertheless, such strategies are based on equal-sized windowing techniques, a difficult parameter to be set that varies accordingly to the stream domain. 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.

Through randomness and combinatorics, the latter approaches can be combined into ensembles to boost accuracy and allow implicit drift adaptation. Nevertheless, one must have in mind that training and maintaining an ensemble is not only computationally costly, but must embed specific diversity induction and voting schemes.

Focusing on 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 $\mathcal{D}' \supseteq \mathcal{D}^*$. By associating to each expert 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 a subset $\mathcal{D}'$ and a dynamic weight, the one with $\mathcal{D}' = \mathcal{D}^*$ will present higher accuracy rates and will overcome other experts' votes in predictions. Nevertheless, by exploring combinatorics the size of the ensemble becomes intractable as the size of the expert grows factorially with M.

Finally, approaches like HEFT-Stream [17] assume that the most discriminative subset of features can be computed by filters on disjoint chunks of instances. The major limitation of this windowing approach is how to determine the size of these windows, which directly affects the learning process. Small windows allow 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 fails on quickly detecting changes in features' relevances.

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 (as CVFDT), however, the same cannot be said for other types of learners.

An interesting and so far unexplored approach is the usage of distribution estimators (e.g. Exponentially Weighted Moving Average [44]) and change detectors (e.g. DDM [43], EDDM [45], ADWIN [29] and Page-Hinkley's test [46]) to keep track of the most discriminant subset of features in streams. With these approaches, there is no need to unnecessarily discard the model learned periodically, but only when changes are detected with statistical confidence.

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

VI. Concluding Remarks

This paper presented, formalized and exemplified one uncommonly addressed characteristic of data streams: feature drifts. We surveyed algorithms that perform feature selection during the stream learning, either aiming or not at feature drifts. Besides serving as an introduction into the research area of dynamic feature selection for data streams, we expect that this paper helps to position new adaptive learning techniques and applications to which these apply.

As discussed in this paper, there are few algorithms that account for feature drifts. Through randomness and combinatorics, implicit approaches adapt to feature drifts by assigning different feature subsets and dynamic weights to experts of an ensemble. These approaches' rationale is that experts associated with most discriminative subsets of features will present higher accuracy rates, therefore, higher weights in voting. During feature drifts, experts with the worst subsets of features will misclassify more instances, therefore will present lower weights, while experts with better subsets will dominate voting due to better subsets. The major limitation of the randomness approaches reside in the sub optimality of the associated subsets of features, since it is possible that no expert is associated with the most discriminative subset of features $\mathcal{D}^*$. Conversely, combinatorics approaches can be posed as wrappers where the several subsets of features are explored in parallel. The major drawback of this approach is that the amount of possible subsets grows factorially, therefore, can be applied only in low dimensional data streams.

Through windowing, CVFDT and HEFT-Stream assume that drifts can be detected by dividing the stream into equal sized chunks, in a naïve approach. Handling streams by adopting predefined sized chunks is problematic since the decision of the chunk size is a tradeoff without solution [10]: a small size implies in a window that reflects the current distribution of data, while a large size enables a larger amount of data to work on, important in periods of stability.

Finally, we conclude that performing dynamic feature selection in data streams has not received the proper attention in the current research scenario. Studying how to perform dynamic feature selection as streams progress allows the tracking of the best subset of features in the stream. This is expected to enhance classifiers' accuracy and diminish processing time even with variations of their relevance, a current open and progressing research challenge.

References

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

[2] P. Kosina and J. a. Gama, "Very fast decision rules for multi-class problems," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, ser. SAC '12. New York, NY, USA: ACM, 2012, pp. 795–800. [Online]. Available: http://doi.acm.org/10.1145/2245276.2245431

[3] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, ser. VLDB '03. VLDB Endowment, 2003, pp. 81–92. [Online]. Available: http://dl.acm.org/citation.cfm?id=1315451.1315460

[4] J. P. Barddal, H. M. Gomes, and F. Enembreck, "SNCStream: A social network-based data stream clustering algorithm," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC)*, ser. SAC 2015. ACM, April 2015.

[5] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *SDM*, 2006, pp. 328–339.

[6] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The clustree: Indexing micro-clusters for anytime stream mining," *Knowl. Inf. Syst.*, vol. 29, no. 2, pp. 249–272, Nov. 2011. [Online]. Available: http://dx.doi.org/10.1007/s10115-010-0342-8

[7] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, Mar. 2014. [Online]. Available: http://doi.acm.org/10.1145/2523813

[8] A. Tsymbal, "The problem of concept drift: definitions and related work," The University of Dublin, Trinity College, Department of Computer Science, Dublin, Ireland, Tech. Rep. TCD-CS-2004-15, 2004.

[9] K. Naidu, A. Dhenge, and K. Wankhade, "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*, ser. CSNT '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 468–471. [Online]. Available: http://dx.doi.org/10.1109/CSNT.2014.99

[10] A. Bifet, *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, 2010, vol. 207. [Online]. Available: http://www.booksonline.iospress.nl/Content/View.aspx?piid=14470

[11] J. Gama, *Knowledge Discovery from Data Streams*, 1st ed. Chapman & Hall/CRC, 2010.

[12] H. Abdulsalam, D. Skillicorn, and P. Martin, "Streaming random forests," in *Database Engineering and Applications Symposium, 2007. IDEAS 2007. 11th International*, Sept 2007, pp. 225–232.

[13] H. Abdulsalam, D. B. Skillicorn, and P. Martin, "Classification using streaming random forests," *IEEE Trans. on Knowl. and Data Eng.*, vol. 23, no. 1, pp. 22–36, Jan. 2011. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2010.36

[14] M. Gupta, J. Gao, C. Aggarwal, and J. Han, "Outlier detection for temporal data," *Synthesis Lectures on Data Mining and Knowledge Discovery*, vol. 5, no. 1, pp. 1–129, 2014. [Online]. Available: http://dx.doi.org/10.2200/S00573ED1V01Y201403DMK008

[15] H.-L. Nguyen, Y.-K. Woon, and W.-K. Ng, "A survey on data stream clustering and classification," *Knowledge and Information Systems*, pp. 1–35, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10115-014-0808-1

[16] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. d. Carvalho, and J. a. Gama, "Data stream clustering: A survey," *ACM Comput. Surv.*, vol. 46, no. 1, pp. 13:1–13:31, Jul. 2013.

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

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

[19] J. P. Barddal, H. M. Gomes, and F. Enembreck, "SFNClassifier: A scale-free social network method to handle concept drift," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC)*, ser. SAC 2014. ACM, March 2014.

[20] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[21] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '00. New York, NY, USA: ACM, 2000, pp. 71–80. [Online]. Available: http://doi.acm.org/10.1145/347090.347107

[22] F. J. Ferrer-Troyano, J. S. Aguilar-Ruiz, and J. C. R. Santos, "Incremental rule learning and border examples selection from numerical data streams," *J. UCS*, vol. 11, no. 8, pp. 1426–1439, 2005. [Online]. Available: http://dx.doi.org/10.3217/jucs-011-08-1426

[23] J. Gama and P. Kosina, "Learning decision rules from data streams," in *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 2011, pp. 1255–1260. [Online]. Available: http://ijcai.org/papers11/Papers/IJCAI11-213.pdf

[24] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '01. New York, NY, USA: ACM, 2001, pp. 97–106. [Online]. Available: http://doi.acm.org/10.1145/502512.502529

[25] E. Almeida, P. Kosina, and J. Gama, "Random rules from data streams," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18-22, 2013*, 2013, pp. 813–814. [Online]. Available: http://doi.acm.org/10.1145/2480362.2480518

[26] M. Sugiyama and Q. Yang, Eds., *Proceedings of the 2nd Asian Conference on Machine Learning, ACML 2010, Tokyo, Japan, November 8-10, 2010*, ser. JMLR Proceedings, vol. 13. JMLR.org, 2010. [Online]. Available: http://jmlr.org/proceedings/papers/v13/

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

[28] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, March 1963. [Online]. Available: http://www.jstor.org/stable/2282952?

[29] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *In SIAM International Conference on Data Mining*, 2007.

[30] R. L. Rivest, "Learning decision lists," *Mach. Learn.*, vol. 2, no. 3, pp. 229–246, Nov. 1987. [Online]. Available: http://dx.doi.org/10.1023/A:1022607331053

[31] A. Bifet, G. Holmes, B. Pfahringer, and R. Gavald, "Improving adaptive bagging methods for evolving data streams," in *Advances in Machine Learning*, ser. Lecture Notes in Computer Science, Z.-H. Zhou and T. Washio, Eds. Springer Berlin Heidelberg, 2009, vol. 5828, pp. 23–37. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-05224-8_4

[32] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science, J. Balczar, F. Bonchi, A. Gionis, and M. Sebag, Eds. Springer Berlin Heidelberg, 2010, vol. 6321, pp. 135–150. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15880-3_15

[33] N. Oza, "Online bagging and boosting," in *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, vol. 3, Oct 2005, pp. 2340–2345 Vol. 3.

[34] L. I. Kuncheva and W. J. Faithfull, "PCA feature extraction for change detection in multidimensional unlabelled data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 69–80, 2014.

[35] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996. [Online]. Available: http://dx.doi.org/10.1023/A:1018054314350

[36] ——, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: http://dx.doi.org/10.1023/A%3A1010933404324

[37] M. Denil, D. Matheson, and N. de Freitas.

[38] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, Dec. 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1314498.1390333

[39] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992. [Online]. Available: http://dx.doi.org/10.1016/S0893-6080(05)80023-1

[40] D. Barbará, "Requirements for clustering data streams," *SIGKDD Explor. Newsl.*, vol. 3, no. 2, pp. 23–27, Jan. 2002. [Online]. Available: http://doi.acm.org/10.1145/507515.507519

[41] N. Jiang and L. Gruenwald, "Cfi-stream: Mining closed frequent itemsets in data streams," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '06. New York, NY, USA: ACM, 2006, pp. 592–597. [Online]. Available: http://doi.acm.org/10.1145/1150402.1150473

[42] A. Metwally, D. Agrawal, and A. El Abbadi, "Duplicate detection in click streams," in *Proceedings of the 14th International Conference on World Wide Web*, ser. WWW '05. New York, NY, USA: ACM, 2005, pp. 12–21. [Online]. Available: http://doi.acm.org/10.1145/1060745.1060753

[43] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence SBIA 2004*, ser. Lecture Notes in Computer Science, A. Bazzan and S. Labidi, Eds. Springer Berlin Heidelberg, 2004, vol. 3171, pp. 286–295. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-28645-5_29

[44] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially Weighted Moving Average Charts for Detecting Concept Drift," *ArXiv e-prints*, Dec. 2012.

[45] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldá, and R. Morales-Bueno, "Early drift detection method," in *In Fourth International Workshop on Knowledge Discovery from Data Streams*, 2006.

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