On Dynamic Feature Weighting for Feature Drifting Data Streams

Jean Paul Barddal[1], Heitor Murilo Gomes[1], Fabrício Enembreck[1], Bernhard Pfahringer[2], and Albert Bifet[3]

[1]Graduate Program in Informatics (PPGIa), Pontifícia Universidade Católica do Paraná
Curitiba, Brazil
[2]Department of Computer Science, University of Waikato
Hamilton, New Zealand
[3]Computer Science & Networks Dept (INFRES), Institut Mines-Télécom, Télécom ParisTech
Université Paris-Saclay
Paris, France
{jean.barddal,hmgomes,fabricio}@ppgia.pucpr.br
bernhard.pfahringer@cs.waikato.ac.nz
abifet@waikato.ac.nz

Abstract. Abstract.

Keywords: Data Stream Mining, Concept Drift, Feature Drift, Feature Weighting

1 Introduction

Data streams are ubiquitous and are generated in a every day faster and bigger manner. Examples include, but are not limited to: ATM transactions, readings in mobile sensor networks and stocks trades. With the uprising popularity of data streams, both researchers and practitioners developed techniques for learning from these potentially unbounded sequences of data in incremental, fast and memory-bounded fashion. Nowadays, this research area focuses on the ephemeral characteristics of data streams, i.e. when the underlying data distribution shifts with time, phenomenon named *concept drift* [35].

More recently, studies [7, 9] has shed light onto a specific kind of drift which has been nearly neglected, the so-called *feature drifts* (also referred as contextual concept drifts in seminal works [35]). In practice, a feature drift occurs whenever a subset of features of a data stream becomes, or ceases to be, relevant to the learning task. As surveyed in [9] and empirically analyzed in [7], feature drifts do pose challenges that are yet to be tackled by the data stream mining community.

In this paper we propose a memory-bounded solution to track the relevance of features during a data stream and show how this can be used to enhance prediction accuracy in k-Nearest Neighbor and Naïve Bayes classifiers during both stable and feature drifting regions of a stream.

2 Learning from Data Streams

Data acquisition and storage is everyday cheaper and easier. Nowadays, a variety of computational systems, from credit card transactions, through wearable gadgets, to video surveillance, create enormous amounts of data, mostly in sequential fashion. Since this abundant – however raw – data do not provide interesting behavior patterns, data mining techniques, especially inductive learning, have been applied to extract useful knowledge from this type of data [13, 10, 21, 6]. Extracting patterns from data streams and their usage in real-time is an effervescent research topic that has been tackled during the last decades [14, 27, 4, 15, 28].

Learning from data streams incorporates all the problems of conventional batch learning, e.g. missing values, noisy data, outliers; but also comprises its own constraints. Most of the conventional learning techniques assume that there is a static dataset generated by an unknown yet stationary probability distribution, which can be stored and analyzed in multiple steps. Nevertheless, none of the latter assumptions are verifiable in several streaming scenarios and the development of new learners must account for several constraints [1, 2, 10, 21, 22, 30, 33]:

- **(Single pass processing).** Classifiers must be able to process instances sequentially accordingly to their arrival. Additionally, the sequence in which instances become available for training and testing is unknown. A classifier must either process an instance, or ignore it, and delete it right after. Although there is no restriction against buffering instances for a limited amount of time, this must not jeopardize *memory space* nor *processing time* constraints.
- **(Memory space).** Primary memory is finite and its usage must be optimized. Therefore, classifiers and their respective models must be bounded accordingly to existing available hardware. Otherwise, the system is jeopardized and will eventually fail.
- **(Processing time).** The processing time of each arriving instance must not surpass the ratio in which new instances become available. If the processing time of each instance scales up, arriving instances will be discarded or enqueued for processing until the system fails. Also, if the algorithm is unable of processing instances in real-time, it will also be unable to adapt to *concept drifts* rapidly.
- **(Concept drift).** 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 (see Sec. 2.2).
- **(Label availability).** After the arrival of an instance $\boldsymbol{x}^t$, it is assumed that its label y^t becomes available for training before the arrival of the following instance $\boldsymbol{x}^{t+1}$. This is, by far, the most used framework for the development of data stream learners [**?**,**?**,**?**,**?**] and frameworks, e.g. Massive Online Analysis (MOA) [12] and Scalable Advanced Massive Online Analysis (SAMOA)

[29]. Although other problem settings for data streams do exist, e.g. semi-supervised and unsupervised learning fashions, they are not yielded in this paper, therefore, the reader is referred to specific works on other learning schemes, e.g. unsupervised learning and semi-supervised learning [8, **?**,**?**,**?**].

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 $\boldsymbol{x}_i$ its corresponding class y_i accurately [10].

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

Accordingly to the Bayesian theory, classification can be posed as the prior probabilities of the classes $P[y]$ and the class conditional probability density functions (*pdfs*) $P[\boldsymbol{x}|y]$ for all possible classes $y_i \in Y$ [21]. 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 $P[\boldsymbol{x}] = \sum_{y_i \in Y} P[y_i] \times P[\boldsymbol{x}|y_i]$.

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

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 incrementally adapt its model f as new instances become available for training [20].

Let $\mathcal{S} = [i^t]_{t=0}^{\infty}$ define a data stream providing instances $i^t = (\boldsymbol{x}^t, y^t)$, each of which arriving at a timestamp t, where $\boldsymbol{x}^t$ 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

Most of the existing classification techniques assume that there is a static dataset generated by a unknown and stationary probability distribution, which can be physically stored and analyzed in multiple steps by a batch algorithm. Nonetheless, none of the latter assumptions can be verified in the streaming scenario and the development of algorithms must account for several constraints [2, 21, 33].

Firstly, instances arrive continuously over time and there is no control over the order that they 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, phenomenons named concept drift and evolution.

Let Eq. 2 denote a concept C, a set of prior probabilities of the classes and class-conditional probability density function [31].

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

Given a stream $\mathcal{S}$, retrieved instances i_t will be generated by a concept C_t. If during every instant t_i of $\mathcal{S}$ 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 $t_j = t_i + \Delta$ (with $\Delta \geq 1$) occurs that $C_{t_i} \neq C_{t_j}$, we have a concept drift.

3 Problem Statement

Until this point, the term "relevance" was used without a formal definition. As stated in [26, 32, 16], there do exist different definitions available in the literature, nevertheless, several may be contradictory and misleading. In this paper we assume a simple, yet, widely used definition of relevance, early provided in [37].

Assuming $S_i = \mathcal{D} \setminus \{D_i\}$, a feature D_i is **relevant** *iff* $\exists S'_i \subset S_i$, such that $P[Y|D_i, S'_i] \neq P[Y|S'_i]$ holds. Otherwise, D_i is said to be **irrelevant**. Accordingly to the previous definition, if a feature that is statistically relevant is removed from a feature set, it will reduce overall prediction power [16]. This definition encompasses two possibilities for a feature to be statistically significant: (i) it is strongly correlated *wrt* the class; or (ii) it forms a feature subset with other features and this subset is strongly correlated *wrt* the class [37].

Most of existing algorithms for data streams tackle the infinite length and drifting concept characteristics. However, not many attention has been given to a specific kind of drift: feature drifts. Feature drifts occur whenever a subset of features becomes, or ceases to be, relevant to the concept to be learned.

Given a feature space $\mathcal{D}$ at a timestamp t, we are able to select the ground-truth relevant subset $\mathcal{D}^*_t \subseteq \mathcal{D}$ such that $\forall D_i \in \mathcal{D}^*_t$ the relevance definition 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, $\mathcal{D}^*_{t_i} \neq \mathcal{D}^*_{t_j}$ betides.

Let $r(D_i, t_j) \in \{0, 1\}$ denote a function which determines whether the relevance definition holds for a feature D_i in a timestamp t_j of the stream. A positive relevance ($r(D_i, t_j) = 1$) states that $D_i \in \mathcal{D}^*$ in a timestamp t_j and that D_i impacts the underlying probabilities $P[\boldsymbol{x}|y_i]$ 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. 3.

$$\exists t_j \exists t_k,\ t_j < t_k,\ r(D_i, t_j) \neq r(D_i, t_k) \tag{3}$$

Changes in $r(\cdot, \cdot)$ directly affect the ground-truth decision boundary to be learned by the learning algorithm. Therefore, feature drifts can be posed as a specific type of concept drift that may occur with or without changes in the data distribution $P[\boldsymbol{x}]$ [9]. This enforces learning algorithms to detect changes in $\mathcal{D}^*$, discerning between features that became irrelevant and the ones that are now relevant and vice-versa. Finally, it is necessary to either swiftly (i) discard and learn an entirely new classification model; or (ii) adapt the current model to relevance drifts [31].

4 Dynamic Feature Weighting

Feature weighting is broadly used in batch learning [25, 3] to assign different weights to feature accordingly to their relevance to the concept to be learned and improve prediction accuracy. As shown earlier, in opposition to static scenarios, the relevance of features may augment or diminish during a data stream, thus, techniques for detecting these changes are needed.

The main hypothesis behind our proposal is that features can be weighted dynamically in order to augment the importance of relevant features and diminish the importance of those which are deemed irrelevant accordingly to feature drifts. First, we show how Entropy and Symmetrical Uncertainty can be computed along sliding windows, and how (i) these metrics can be used to detect changes in features' relevances; and (ii) applied in two different learning schemes to boost prediction accuracy during both stable and feature drifting regions of data streams. Finally, we detail the bounded computational overhead this proposal provides in processing time and memory usage.

4.1 Preliminaries

The relevance of a feature can be computed in diverse ways. In this section we discuss evaluation techniques for measuring the goodness of features for classification. Generally, a feature is good if it relevant to predict the class. If one adopts correlation to measure the goodness of a feature, a feature will be deemed as relevant it its value surpasses a given threshold.

There do exist several approaches to measure the correlation between two random variables. The first is using linear correlation and the other is based on information theory aspects.

The most common formula for computing the correlation for a pair of variables (X, Y) is linear correlation coefficient:

$$r(X, Y) = \frac{\sum_{q \in D_i} \sum_{y_i \in Y} (q - \bar{D}_i)(y_i - \bar{Y})}{\sqrt{\sum_{q \in D_i} (q - \bar{D}_i)^2} \sqrt{\sum_{y_i \in Y} (y_i - \bar{Y})^2}} \tag{4}$$

The linear correlation coefficient is bounded in the $[-1;1]$ interval. If X and Y are completely correlated, r takes the value of 1 or -1; and if these variables are completely uncorrelated, r is 0. Adopting linear correlation as a feature goodness measure has the benefit of eliminating completely uncorrelated features. Also, if data is linearly separable in the original representation if it is still separable if all but one of a group of linearly dependent features are removed [36]. Nevertheless, assuming linear correlations is not safe for a variety of domains. Linear correlation is likely to be unable to depict correlations which are non-linear in nature. In our proposal, we adopt information theory approaches to compute the goodness of a feature. The first one is a measure of uncertainty of a random variable, named entropy. The entropy of a variable X is given by:

$$H(X) = -\sum_{x_i}^{X} P[X = x_i] \log_2 P[X = x_i] \tag{5}$$

On the other hand, the entropy of a variable X after observing values of a variable Y is given by:

$$H(X|Y) = -\sum_{y_j}^{Y} P[Y = y_j] \sum_{x_i}^{X} P[X = x_i | Y = y_j] \tag{6}$$

We now present the mathematical foundation for computing entropy along sliding windows.

Assuming $X = \{x_i\}_{i=1}^{n}$ to be a sample of real numbers, $S_n = \sum_{i=1}^{n} x_i$ be the sum of these elements, the sample entropy H_s is defined as follows:

$$H_n = -\sum_{i=1}^{n} \frac{x_i}{S_n} \log_2 \frac{x_i}{S_n} \tag{7}$$

Lemma 1. *Let $X = \{x_i\}_{i=1}^{n}$ be a sample of real numbers, $S_n = \sum_{i=1}^{n} x_i$ be the sum of the sample elements and H_s be the entropy of this sample. For any positive real number $R > 0$, it occurs that [34]:*

$$-\sum_{i=1}^{n} \left(\frac{x_i}{S_n + R} \log_2 \frac{x_i}{S_n + R} \right) = \frac{S_n}{S_n + R} \left(H_n - \log_2 \frac{S_n}{S_n + R} \right) \tag{8}$$

Proof. If one writes $\frac{x_i}{R+S_n} = 1 \cdot \frac{x_i}{R+S_n} = \frac{S_n}{S_n}\frac{x_i}{S_n+R} = \frac{x_i}{S_n}\frac{S_n}{S_n+R}$, it follows:

$$\begin{aligned}
-\sum_{i=1}^{n}\left(\frac{x_i}{S_n+R}\log_2\frac{x_i}{S_n+R}\right) &= -\sum_{i=1}^{n}\frac{x_i}{S_n}\frac{S_n}{S_n+R}\log_2\left(\frac{x_i}{S_n}\frac{S_n}{S_n+R}\right)\\
&= -\sum_{i=1}^{n}\frac{x_i}{S_n}\frac{S_n}{S_n+R}\left(\log_2\frac{x_i}{S_n}+\log_2\frac{S_n}{S_n+R}\right)\\
&= -\frac{S_n}{S_n+R}\left(\sum_{i=1}^{n}\frac{x_i}{S_n}\log_2\frac{x_i}{S_n}+\sum_{i=1}^{n}\frac{x_i}{S_n}\log_2\frac{S_n}{S_n+R}\right)\\
&= \frac{S_n}{S_n+R}\left(H_n-\log_2\frac{S_n}{S_n+R}\sum_{i=1}^{n}\frac{x_i}{S_n}\right)\\
&= \frac{S_n}{S_n+R}\left(H_n-\log_2\frac{S_n}{S_n+R}\right) \quad \square
\end{aligned}$$

The sample entropy can be updated when a new positive real number $x_i > 0$ enters it as follows.

Lemma 2. *Based on H_n and S_n, the entropy can be updated after the arrival of a new positive real number $x_{n+1} > 0$ accordingly to:*

$$H_{n+1} = \frac{S_n}{S_{n+1}}\left(H_n - \log_2\frac{S_n}{S_{n+1}}\right) - \frac{x_{n+1}}{S_{n+1}}\log_2\frac{x_{n+1}}{S_{n+1}} \tag{9}$$

Proof. By definition, and by following Lemma 2, the updated entropy is given by:

$$\begin{aligned}
H_{n+1} &= -\sum_{i=1}^{n+1}\frac{x_i}{S_{n+1}}\log_2\frac{x_i}{S_{n+1}}\\
&= -\frac{x_{n+1}}{S_n} - \sum_{i=1}^{n}\frac{x_i}{S_{n+1}}\log_2\frac{x_i}{S_{n+1}}\\
&= \frac{S_n}{S_{n+1}}\left(H_n - \log_2\frac{S_n}{S_{n+1}}\right) - \frac{x_{n+1}}{S_{n+1}}\log_2\frac{x_{n+1}}{S_{n+1}} \quad \square
\end{aligned}$$

In Algorithm 1 we present the pseudocode for entropy computation over sliding windows. Proofs for Entropy equations (lines 14 and 18) can be found in [34].

Entropy is the base for computing more robust metrics. One example is the Information Gain, which is the amount by which the entropy of a variable X decreases reflects additional information about X provided by Y and is given by:

$$IG(X|Y) = H(X) - H(X|Y) \tag{10}$$

An important trait of information gain is that it is symmetrical, i.e. $IG(X|Y) = IG(Y|X)$. To prove it, one needs to verify that $H(X) - H(X|Y) = H(Y) -$

Algorithm 1: Sliding window entropy. Adapted from [34].

```
input  : window size $w$, a data stream $\mathcal{S}$.
output : be ready to provide the entropy $h$ at any time.
1  Let $W \leftarrow \emptyset$ be the sliding window;
2  Let $h \leftarrow 0$ be the entropy;
3  Let $n \leftarrow 0$ be the amount of instances in $W$;
4  Let $n_i \leftarrow 0$ be the number of instances with the $y_i$-th label;
5  foreach $(\boldsymbol{x}_i, y_i) \in \mathcal{S}$ do
6      if $|W| = w$ then
7          Dequeue oldest element from W from the $y_j$-th class;
8          $h \leftarrow DEC(h, n, n_j)$;
9      $W \leftarrow W \cup \{(\boldsymbol{x}_i, y_i)\}$;
10     $h \leftarrow INC(h, n, n_i)$;
11 Function $INC(h, n, n_i)$
12     Update $n \leftarrow n + 1$;
13     Update $n_i \leftarrow n_i + 1$;
14     return $\frac{n-1}{n}\left(h - \log_2 \frac{n-1}{n}\right) - \frac{n_i}{n} \log_2 \frac{n_i}{n} + \frac{n_i-1}{n} \log_2 \frac{n_i-1}{n}$
15 Function $DEC(h, n, n_i)$
16     Update $n \leftarrow n - 1$;
17     Update $n_i \leftarrow n_i - 1$;
18     return $\frac{n+1}{n}\left(h + \frac{n_i+1}{n+1} \log_2 \frac{n_i+1}{n+1} - \frac{n_i}{n+1} \log_2 \frac{n_i}{n+1}\right) + \log_2 \frac{n}{n+1}$
```

$H(Y|X)$. This can be derived from $H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y)$.

As entropy, information gain is biased towards features with more values. Therefore, different metrics that compensate for this bias are preferred. In this paper we picked symmetrical uncertainty (SU) as a goodness measure since it atones this bias. Symmetrical uncertainty can be computed as follows:

$$SU(X,Y) = 2\left[\frac{IG(X|Y)}{H(X) + H(Y)}\right] = 2\left[\frac{H(Y) - H(Y|X)}{H(X) + H(Y)}\right] \tag{11}$$

SU is ranged in the $[0;1]$ interval where 1 indicates that the value of a variable completely predicts the other, while 0 indicates that X and Y are completely independent.

In order to compute SU along a sliding window, one must keep track of $H(D_i)$, $H(Y)$ and $H(Y|D_i)$ entropies. Both $H(D_i)$ and $H(Y)$ can be incremented and decremented in $\mathcal{O}(1)$ accordingly to Alg. 1, while the conditional entropy $H(Y|D_i)$ can be computed with separate $H(Y|D_i = q)$ entropies, also given by Alg. 1. If we assume that such that $q \in D_i$ and $|D_i| = m$, then SU can be computed with low computational complexity in the $\mathcal{O}(m)$ order for a single feature and $\mathcal{O}(dm)$ for all features in a d-dimensional data stream.

Memory-wise, the cost of tracking $H(Y)$ is $\mathcal{O}(|Y|)$, while the cost for $H(D_i)$ is $\mathcal{O}(m)$, thus, a total complexity of $\mathcal{O}(md)$ for a d-dimensional stream. Finally,

$H(Y|D_i = q)$ has each a cost $\mathcal{O}(|Y|)$, thus, totalizing a cost of $\mathcal{O}(md \times |Y|)$ for all features $D_i \in \mathcal{D}$.

4.2 Applying Feature Weighting on k-Nearest Neighbor Learning

KNN-FW

k-Nearest Neighbor (*kNN*) [**?**] is one of the most fundamental, simple and widely used classification methods, which is able to learn complex (non-linear) functions [**?**]. *kNN* is a lazy learner since it does not require building a model before actual use. It classifies unlabeled instances accordingly to "closest" previously seen labeled ones stored in buffer. The definition of "close" means that a distance measure is used to determine how similar/dissimilar two instances are. There are several approaches to compute distances between instances, nevertheless, the most used one is the Euclidian distance, given by Eq. 12, where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are two arbitrary instances, and the summation occurs over all features $D_k \in \mathcal{D}$.

$$d_E(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum_{D_k \in \mathcal{D}} (\boldsymbol{x}_i[D_k] - \boldsymbol{x}_j[D_k])^2} \tag{12}$$

kNN classifies unlabeled instances accordingly to the label of the majority of the k closest instances stored in buffer. Therefore, picking an appropriate value of k for each application is significant. If k is too small, *kNN* becomes more prone to overfitting and tends to misclassify instances in easy situations. Conversely, bigger values of k may mislead classification in cases when an instance is surrounded by several instances of an opposite label.

Another important trait of kNN refers to the dimensionality of the problem, either in static or streaming scenarios. As discussed in a variety of works [15, 33], Euclidian distances fail on representing in effective fashion the distance between points (instances) in a high-dimensional space, phenomenon named "curse of dimensionality".

Although the "curse of dimensionality" is commonly tackled in static learning scenarios, very few works aim at providing techniques for dealing with it in streaming scenarios, either through feature selection or dimensionality reduction.

Performing *kNN* classification in data streams requires an additional important trait: dealing with time and memory limitations. Continuously buffering instances as they arrive is unfeasible since the stream is potentially unbounded, therefore, an incremental version of *kNN* must "forget" older instances as the stream progresses. A naive approach to perform forgetting is storing instances in a queue with size W. Again, defining a value for W is non trivial and it must be set accordingly to available memory space and processing time, since the computational time for classifying each new instance is $\mathcal{O}(Wd)$.

4.3 Applying Feature Weighting on Naïve Bayes

NB-FW

problema no produtrio pois se o peso for zero, zera a probabilidade inteira. deve ser utilizado um padding factor muito baixo como 0.0001 ao invs de 0.

5 Analysis

In order to assess our proposal's performance, we built an experimentation environment encompassing both synthetic and real-world data. This analysis centers on prediction accuracy, processing time and memory usage.

5.1 Synthetic Data Stream Generators

Drifts are synthesized as the combination of two pure distributions. The probability that an instance is drawn from the prior or posterior concept inside a drift window is given by a sigmoid function. This drift framework is the default provided in the MOA framework [12] and all drifts windows have a length of 1,000 instances. All synthesized data streams contain 100,000 instances and contain 9 feature drifts.

We used the following three synthetic data generators to induce feature drifts in our experiments: AGRAWAL [5], Assets Negotiation (ASSETS) [18] and SEA-FD [7]. In all drifts, the relevant subset of features in the prior and posterior concepts differ.

5.2 Real-World Data

Jutting synthetic data, real-world datasets were used in the evaluation of our proposal. The adoption of real-world data is beneficial since they present differentiated behavior, e.g. the class distribution is often unbalanced and data is often noisy. On the other hand, it is nearly impossible to affirm whether drifts occur, making evaluation of drift detection unfeasible. We refrain from providing a detailed description of each used dataset for brevity. The used datasets are: Airlines (AIR) [**?**], Electricity (ELEC) [**?**] and Kaggle's Give me Some Credit[1] (GMSC).

5.3 Evaluated Algorithms

Besides kNN and Naive Bayes, we also report results for a Very Fast Decision Tree (VFDT) and a Concept-adapting Very Fast Decision Tree (CVFDT) since both perform embedded feature selection during training.

VFDT Very Fast Decision Tree (VFDT) is an incremental decision tree learner for non-drifting data streams [17]. The tree is recursively built as instances arrive and new split nodes are generated if the information gain of the two most discriminative features differ at least by ϵ, given by the Hoeffding bound [23].

[1] Available at: `https://www.kaggle.com/c/GiveMeSomeCredit`. Last access in Feb. 25th, 2016.

Table 1: Prequential accuracy (%).

Experiment	kNN	kNN-SU	NB	NB-SU	VFDT	CVFDT
AGRAWAL						
ASSETS						
SEA-FD						
AIR						
ELEC						
SPAM						
GMSC						

CVFDT Concept-adapting Very Fast Decision Tree (CVFDT) algorithm is an extension to the VFDT to deal with drifts [24]. CVFDT updates its tree model over a sliding window and creates or updates decision nodes if the data distribution changes at an arbitrary split node. In this paper, we adopted a version of CVFDT which detects changes in data distribution accordingly to ADWIN change detector [11] provided in MOA [12]. Whenever ADWIN detects a change in a split node, the entire subtree is replaced by a new split node with the most discriminant feature if the Hoeffding bound is met.

5.4 Experimental Protocol

Accuracy is computed accordingly to the Prequential test-then-train procedure [19]. Prequential was chosen due to the monitoring of models' performance over time. Although it is considered pessimistic, authors in [19] observe that this estimative converges to a holdout when estimated along a sliding window. Equation 13 presents the Prequential accuracy computation, where $L(\cdot, \cdot)$ is a loss function (0-1 in this paper) between the ground-truth class y_k and the predicted $\hat{y}_k$, i is the current timestamp and w is the length of the sliding window ($w = 1,000$ in experiments).

$$P_{acc} = 1 - \frac{1}{w} \sum_{k=i-w+1}^{i} L(y_k, \hat{y}_k) \tag{13}$$

Processing time is measured as the time, in seconds, the algorithm spends in CPU, while memory usage is given in RAM-Hours, where 1 RAM-Hour equals 1 GB of RAM dispended per hour of processing (GB-Hour).

5.5 Discussion

discussion

6 Conclusion

This paper presented a time and memory-bounded solution for tracking the relevance of features based on the information theoretic concepts of Entropy

Table 2: Processing time (s).

Experiment	kNN	kNN-SU	NB	NB-SU	VFDT	CVFDT
AGRAWAL						
ASSETS						
SEA-FD						
AIR						
ELEC						
SPAM						
GMSC						

Table 3: RAM-Hours (GB-Hour).

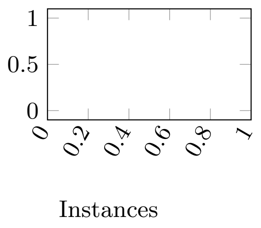
Experiment	kNN	kNN-SU	NB	NB-SU	VFDT	CVFDT
AGRAWAL						
ASSETS						
SEA-FD						
AIR						
ELEC						
SPAM						
GMSC						

and Symmetrical Uncertainty. Additionally, it showed how these metrics can be successfully used to enhance both k-Nearest Neighbor and Naïve Bayesian algorithms during both stable and feature drifting regions of data streams. Empirical evidences show that the gains in prediction accuracy are significant and occur in both synthetic and real-world datasets. Therefore, the results shown here point out the need for future research into feature drift detection and adaptation.

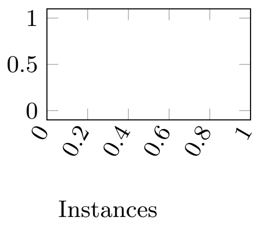
Both entropy and symmetrical uncertainty are computed along a sliding window, thus allowing adaptation to feature drifts. Future works include the adopting of change detectors (e.g. ADWIN [**?**] and EWMA [**?**]) to eliminate the need of a predefined window size, which is a drawback of the proposed method.

Finally, there is the need to investigate the usage of these adaptive metrics (entropy and symmetrical uncertainty) in the task of feature selection for data streams. This would allow a more generic filter method that does not depend on the base classifier and that selects features dynamically accordingly to the occurrence of feature drifts.

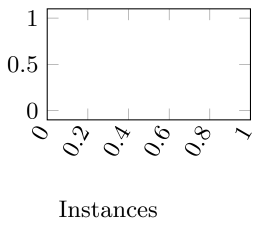
Acknowledgments. This project was partially financially supported by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* (CAPES) through the *Programa de Suporte à Pós-Graduação de Instituições de Ensino Particulares* (PROSUP) program for Doctorate students.



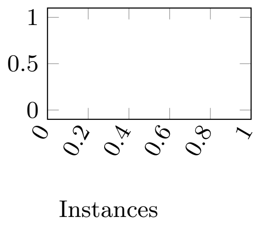
(a) AGRAWAL



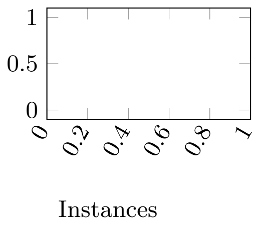
(b) ASSETS



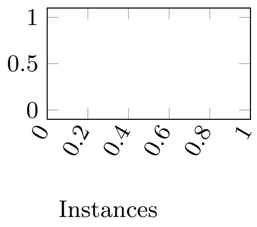
(c) SEA-FD



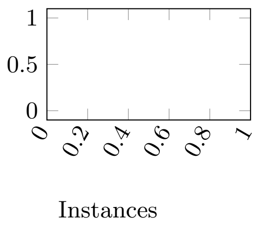
(d) AIR



(e) ELEC



(f) SPAM



(g) GMSC

Fig. 1: Prequential accuracy (%) obtained during experiments.

References

1. H. Abdulsalam, D.B. Skillicorn, and P. Martin. Streaming random forests. In *Database Engineering and Applications Symposium, 2007. IDEAS 2007. 11th International*, pages 225–232, Sept 2007.
2. Hanady Abdulsalam, David B. Skillicorn, and Patrick Martin. Classification using streaming random forests. *IEEE Trans. on Knowl. and Data Eng.*, 23(1):22–36, January 2011.
3. Charu C. Aggarwal. An introduction to data classification. In *Data Classification: Algorithms and Applications*, pages 1–36. 2014.
4. Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 81–92. VLDB Endowment, 2003.

5. R. Agrawal, T. Imielinski, and Arun Swami. Database mining: a performance perspective. *Knowledge and Data Engineering, IEEE Transactions on*, 5(6):914–925, Dec 1993.
6. Ezilda Almeida, Petr Kosina, and João 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*, pages 813–814, 2013.
7. Jean Paul Barddal, Heitor Murilo Gomes, and Fabrício Enembreck. Analyzing the impact of feature drifts in streaming learning. In *Proceedings of the 22th International Conference on Neural Information Processing*, ICONIP 2015. Springer, November 2015.
8. Jean Paul Barddal, Heitor Murilo Gomes, and Fabrício Enembreck. SNCStream: A social network-based data stream clustering algorithm. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC)*, SAC 2015. ACM, April 2015.
9. Jean Paul Barddal, Heitor Murilo Gomes, and Fabrício Enembreck. A survey on feature drift adaptation. In *Proceedings of the International Conference on Tools with Artificial Intelligence*. IEEE, November 2015.
10. Albert Bifet. *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*, volume 207 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2010.
11. Albert Bifet and Ricard Gavaldà. Learning from time-changing data with adaptive windowing. In *In SIAM International Conference on Data Mining*, 2007.
12. Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: Massive online analysis. *The Journal of Machine Learning Research*, 11:1601–1604, 2010.
13. Albert Bifet, Bernhard Pfahringer, Jesse Read, and Geoff Holmes. Efficient data stream classification via probabilistic adaptive windows. In *SAC*, pages 801–806, 2013.
14. Albert Bifet, Jesse Read, Indre Zliobaite, Bernhard Pfahringer, and Geoff Holmes. Pitfalls in benchmarking data stream classification and how to avoid them. In *ECML/PKDD (1)*, pages 465–479, 2013.
15. Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *SDM*, pages 328–339, 2006.
16. K.J. Cios, W. Pedrycz, R.W. Swiniarski, and L. Kurgan. *Data Mining: A Knowledge Discovery Approach*. Springer US, 2007.
17. Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 71–80, New York, NY, USA, 2000. ACM.
18. Fabrcio Enembreck, Brulio Coelho vila, Edson E. Scalabrin, and Jean-Paul A. BarthÈs. Learning drifting negotiations. *Applied Artificial Intelligence*, 21(9):861–881, 2007.
19. J. Gama and P. Rodrigues. Issues in evaluation of stream learning algorithms. In *Proc. of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 329–338. ACM SIGKDD, Jun. 2009.
20. Joao Gama. *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 1st edition, 2010.
21. Joo Gama, Indre Zliobaite, Albert Bifet, Mykole Pechenizkiy, and Abderlhamid Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, March 2014.

22. Manish Gupta, Jing Gao, Charu Aggarwal, and Jiawei Han. Outlier detection for temporal data. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 5(1):1–129, 2014.
23. Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.
24. Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 97–106, New York, NY, USA, 2001. ACM.
25. Norbert Jankowski and Krzysztof Usowicz. *Neural Information Processing: 18th International Conference, ICONIP 2011, Shanghai, China, November 13-17, 2011, Proceedings, Part II*, chapter Analysis of Feature Weighting Methods Based on Feature Ranking Methods for Classification, pages 238–247. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
26. Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(12):273 – 324, 1997. Relevance.
27. Petr Kosina and Joao Gama. Very fast decision rules for multi-class problems. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 795–800, New York, NY, USA, 2012. ACM.
28. Philipp Kranen, Ira Assent, Corinna Baldauf, and Thomas Seidl. The clustree: Indexing micro-clusters for anytime stream mining. *Knowl. Inf. Syst.*, 29(2):249–272, November 2011.
29. Gianmarco De Francisci Morales and Albert Bifet. Samoa: Scalable advanced massive online analysis. *Journal of Machine Learning Research*, 16:149–153, 2015.
30. Hai-Long Nguyen, Yew-Kwong Woon, and Wee-Keong Ng. A survey on data stream clustering and classification. *Knowledge and Information Systems*, pages 1–35, 2014.
31. Hai-Long Nguyen, Yew-Kwong Woon, Wee-Keong Ng, and Li Wan. Heterogeneous ensemble for feature drifts in data streams. In Pang-Ning Tan, Sanjay Chawla, ChinKuan Ho, and James Bailey, editors, *Advances in Knowledge Discovery and Data Mining*, volume 7302 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2012.
32. Witold R. Rudnicki, Mariusz WrzesieŃ, and Wiesaw Paja. All relevant feature selection methods and applications. In *Feature Selection for Data and Pattern Recognition*, volume 584 of *Studies in Computational Intelligence*, pages 11–28. Springer Berlin Heidelberg, 2015.
33. Jonathan A. Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, André C. P. L. F. de Carvalho, and João Gama. Data stream clustering: A survey. *ACM Comput. Surv.*, 46(1):13:1–13:31, July 2013.
34. Blaz Sovdat. Updating formulas and algorithms for computing entropy and gini index on time-changing data streams. *CoRR*, abs/1403.6348, 2014.
35. Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Mach. Learn.*, 23(1):69–101, April 1996.
36. Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 856–863. AAAI Press, 2003.
37. Zheng Zhao, Fred Morstatter, Shashvata Sharma, Salem Alelyani, Aneeth Anand, and Huan Liu. Advancing feature selection research. *ASU feature selection repository*, pages 1–28, 2010.