Expert Systems With Applications 116 (2019) 227–242

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Merit-guided dynamic feature selection filter for data streams

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

[a] *Graduate Program in Informatics (PPGIa), Pontifícia Universidade Católica do Paraná (PUCPR), Curitiba, Brazil*
[b] *INFRES, LTCI, Télécom ParisTech, Paris, France*
[c] *Department of Computer Science, University of Waikato, Hamilton, New Zealand*

ARTICLE INFO

Article history:
Received 28 June 2018
Revised 7 August 2018
Accepted 12 September 2018
Available online 13 September 2018

Keywords:
Data stream classification
Feature selection
Concept drift

ABSTRACT

Learning from ephemeral data streams has garnered the interest of both researchers and practitioners towards adaptive learning techniques. Despite the convincing results obtained thus far, most of the current research still overlooks that the relevance of features may change throughout the learning process. Scenarios where features become - or cease to be - relevant to the learning task are called feature drifting data streams, and the identification of which features are relevant becomes even more challenging when the feature space is high-dimensional. To select relevant features during the progress of data streams, we propose a merit-guided and classifier-independent dynamic feature selection algorithm named *DynamIc SymmetriCal Uncertainty Selection for Streams* (DISCUSS). We evaluate our proposal on both synthetic and real-world datasets and show that DISCUSS can boost kNN and Naive Bayes classifiers' accuracy rates on high-dimensional data streams, while at the expense of limited processing time and memory space. Finally, the drawbacks of the proposed method are assessed, and possible future works on the topic are also discussed.

1. Introduction

Every day, companies and individuals gather enormous amounts of data at increasing rates. Most of these data are generated sequentially from different sources and are so massive that their storage would neither be practical, intelligible, nor useful. Much effort has been directed towards mining these potentially infinite sequences of data, and this area is commonly referred to as data stream mining.

By far the most widely researched task on data stream mining is classification. Data stream classification, or online classification, regards the problem of learning a model that is capable of predicting a nominal value given a feature vector. Despite being light-weighted regarding both processing time and memory usage, novel learning proposals must tackle the ephemeral property of streams, namely *concept drift*.

As described in the seminal work of Widmer and Kubat (1996), data streams are susceptible to different types of drifts: (i) changes in the characteristic feature values, (ii) evolution of the value domains of features over time, or (iii) *features that were once relevant and that now may become meaningless or the other way around*, and so on. In this paper, we target the above-emphasized type of drift, called **feature drift**. Feature drift occurs when a subset of features becomes, or ceases to be, relevant to the learning task (Barddal, Gomes, Enembreck, & Pfahringer, 2017).

To maintain an accurate predictive model on a data stream that exhibits feature drifts, a classifier must be trained and updated on the subset of features that is currently relevant. Feature selection is crucial in these scenarios since the goal is to retain the relevant subset of features of a data set. Naturally, identifying and keeping track of which features are relevant in high-dimensional data streams turn the problem even more complex, as the data becomes more sparse and computationally prohibitive. Therefore, performing feature selection over high-dimensional data streams is still an open research topic since the majority of existing feature selection algorithms require multiple passes over data (Barddal et al., 2017). In addition to feature drifts, our proposal is also tailored for tackling *redundant features*. Redundant features are problematic as they provide an extra and unnecessary computational cost for both storage and processing, and make classifiers more prone to overfit (Yu & Liu, 2003). Therefore, feature selection should also identify and eliminate redundant features during learning.

* Corresponding author.
E-mail addresses: jean.barddal@ppgia.pucpr.br (J.P. Barddal), fabricio@ppgia.pucpr.br (F. Enembreck), heitor.gomes@telecom-paristech.fr (H.M. Gomes), albert.bifet@telecom-paristech.fr (A. Bifet), bernhard@waikato.ac.nz (B. Pfahringer).

https://doi.org/10.1016/j.eswa.2018.09.031
0957-4174/© 2018 Elsevier Ltd. All rights reserved.

This paper introduces a novel dynamic feature selection method for data streams named *DynamIc SymmetriCal Uncertainty Selection for Streams* (DISCUSS). DISCUSS's core is built on top of the symmetrical uncertainty operator (Witten & Frank, 2005), a concept coming from Information Theory. DISCUSS follows the procedure earlier introduced in Barddal, Gomes, Enembreck, Pfahringer, and Bifet (2016a) to swiftly compute symmetrical uncertainty between each feature and the class over sliding windows, and thus, acting as a filter, it can dynamically select features as the stream progresses. The central claim on proposing DISCUSS is that it will dynamically check the importance of features w.r.t. class prediction, while also identifying redundant ones. Given that, DISCUSS will continuously try to maximize a merit function that aims the maximization of feature relevance while decreasing feature redundancy. As a result, classification models should be learned within smaller dimensionalities, also prospectively resulting in smaller processing times and memory consumption rates. The main contributions of this work are as follows:

- The proposal of a novel dynamic feature selection algorithm for data streams.
- The evaluation of the proposed method with different classifiers and several high-dimensional feature drifting data streams. These experiments highlight that all Naive Bayes, kNN and Decision Tree classifiers' accuracy rates suffer from mild increases in dimensionality, while DISCUSS allows all of them to overcome drifts in such scenarios; and
- A discussion on the main shortcomings of the proposed method and open gaps that should be assessed by the data stream mining community in future works.

This paper is structured as follows. Section 2 introduces the data stream classification and concept drift problem. Section 3 states the problems we intend to assess: *feature drifts* and *redundant features*. Section 4 surveys related works on feature drift adaptation, also highlighting which are used in our analysis. Section 5 describes our proposed dynamic feature selection algorithm, which is later evaluated in Section 6. Finally, Section 7 concludes this paper and states envisioned future works.

2. Data stream classification

In this paper, we focus on the streaming classification task, which regards learning and updating predictive models over time. Formally, we denote $\mathcal{S}$ to be a stream of instances in the $i^t = (\vec{x}^t, y^t)$ form, where t is the arrival timestamp, $\vec{x} \in X$ is a vector of features that are possibly categorical, numeric or most likely mixed, and $y^t \in Y$ the class. To denote the D_j-th feature of an instance x^t, we adopt the x^t_j notation, where t will be dropped throughout this paper when unnecessary, and the entire feature set is denoted by $\mathcal{D} = \{D_1, D_2, \ldots, D_d\}$. Given $\mathcal{S}$, the classification goal is to learn and update a model f: $X \rightarrow Y$ over time. Updates $f^t = f^{t-1} \pm \Delta$ can be either incremental, if the underlying patterns obtained from incoming instances adhere to the current concept; or decremental, for example, when a concept drift occurs. Generally speaking, the underlying concept C of a stream is a set of prior probabilities of the classes and class-conditional probability density functions (Nguyen, Woon, Ng, & Wan, 2012):

$$C = \bigcup_{y_i \in Y} \{(P[y_i], P[\vec{x}|y_i])\} \tag{1}$$

Given $\mathcal{S}$, instances will be labeled according to the current concept C^t. If between two timestamps t_i and $t_j > t_i$ it follows that $C^{t_i} \neq C^{t_j}$, then we have a concept drift (Webb, Hyde, Cao, Nguyen, & Petitjean, 2016). Another important categorization for concept drifts regards their length: if $C^{t_i} \neq C^{t_i+1}$ the drift is said to be abrupt, while if $C^{t_i} \neq C^{t_i+\Delta}$ with $\Delta > 1$ occurs, the drift is called gradual.

3. Problem statement

In this section, we focus on the two issues we aspire to tackle with our proposal: **feature drifts** and **redundant features**. To formalize both issues, we first need to categorize features into three types: relevant, irrelevant and redundant; where the first two are essential to introduce feature drifts, and the last should be assessed independently.

As stated in Rudnicki, Wrzesień, and Paja (2015), there do exist different relevance definitions available in the literature, yet, some may be contradictory and misleading. In this work a combined definition of relevance is used, given by Definitions 1 and 2 proposed in Kohavi and John (1997). First, strong relevance occurs when a feature is indispensable in the sense that it cannot be removed without losses in prediction accuracy, whereas weak relevance implies that a feature may contribute to prediction accuracy.

Definition 1. (Strong relevance) Assuming $S_i = \mathcal{D} \setminus \{D_i\}$, a feature D_i is **strongly relevant** *iff* there exists some q (one of the possible outcomes for a feature D_i), y and s_i for which $P[D_i = q, S_i] > 0$ such that the following holds:

$$P[Y = y, |D_i = q, S_i = s_i] \neq P[Y = y, S_i = s_i] \tag{2}$$

Definition 2. (Weak relevance) A feature D_i, will be considered as **weakly relevant** iff Definition 1 does not hold, and there exists a subset of features $S'_i \subset S_i$ for which exists some q, y and s'_i with $P[D_i = q, S'_i = s'_i] > 0$ such that the following holds:

$$P[Y = y, |D_i = q, S'_i = s'_i] \neq P[Y = y, S'_i = s'_i] \tag{3}$$

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

According to these definitions, the removal of a feature that is statistically relevant from a feature subset will result in a reduction of prediction power. This definition encompasses two possibilities for a feature to be relevant: (i) it alone is strongly correlated with the class (Kohavi & John, 1997); or (ii) it forms a feature subset with other features that together are correlated with the class (Zhao et al., 2010).

Most of the research on concept drift adaptation assumes that changes in the data distribution occur inside the skewness of the classes in ranges of features' values. Feature drifts are a different type of problem, where entire features become, or cease to be, relevant to the concept to be learned. This type of changes forces the learning algorithm to adapt its model to ignore the irrelevant features and account for the newly relevant ones (Nguyen et al., 2012).

Definition 3. (Feature drift) 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 $\forall D_i \in \mathcal{D}^*_t$ either Definitions 1 or 2 hold, and $\forall D_j \in \mathcal{D} \setminus \mathcal{D}^*_t$ the same definitions do not. A feature drift occurs if, at any two time instants t_i and $t_j = t_i + \Delta$, $\mathcal{D}^*_{t_i} \neq \mathcal{D}^*_{t_j}$ is verified.

Like other types of drifts, changes in the relevant subset of features affect the ground-truth decision boundary to be learned by the classifier. It 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 (i) discard and learn an entirely new classification model; or (ii) adapt the current model to relevance drifts (Nguyen et al., 2012).

Besides tracking relevance, another essential property of features is redundancy. Redundancy notions are given using feature correlation terms. It is common sense that two features are redundant to a concept if their values are correlated. Redundant features are problematic as they provide an extra computational cost for both storage and processing and make classifiers more prone to overfit (Yu & Liu, 2003). Therefore, feature selection should also identify and eliminate redundant features during learning.

Definition 4. (Redundant Feature) Assuming $S_i = \mathcal{D} \setminus \{D_i\}$, a feature D_i is **redundant** *iff*

$$P[Y|S_i] \approx P[Y|D_i, S_i] \tag{4}$$

According to Definition 4, a feature is redundant if there is another feature, or set of features, that yield similar prediction power. Several studies proposed the removal of redundant features as this might improve prediction accuracy since fewer features often lead to less over-fitted models (Hall, 2000; Park, 2013), while others noticed that the removal of this type of feature might cause the exclusion of potentially relevant features (Guyon, 2003).

4. Related work

Determining which features are relevant is a problem that has been widely tackled in batch machine learning. However, the same cannot be said about streaming environments. As depicted in recent surveys on the topic (Barddal, Gomes, & Enembreck, 2015; Barddal et al., 2017), there are few works that explicitly tackle the problem of drifting features. They can be categorized into: decision trees (Bifet & Gavaldà, 2009; Domingos & Hulten, 2000), rule learning (Almeida, Ferreira, & Gama, 2013) and ensembles (Abdulsalam, Skillicorn, & Martin, 2011; Nguyen et al., 2012). As the result of an evaluation process using a multitude of experiments, authors in (Barddal et al., 2017) concluded that a single adaptive decision tree, namely Hoeffding Adaptive Tree (HAT) (Bifet & Gavaldà, 2009) was the best performing classifier w.r.t. the trade-off between classification rates, processing time and memory usage.

The HAT algorithm is an extension to the incremental Very Fast Decision Tree classifier (Domingos & Hulten, 2000), which uses the ADaptive sliding WINdow (ADWIN) drift detector (Bifet & Gavaldà, 2007) inside decision nodes to monitor the internal error rates of the tree. When a drift is flagged by a decision node, the entire subtree is reset, and the tree starts re-learning. This allows the HAT classifier to detect and overcome feature drifts since changes will affect how good a split on a feature is and this will be detected by ADWIN. After each flagging, the tree is then able to replace this decision stump by a new one that will eventually branch on a newly relevant feature after the drift. Yet, it is worthy to highlight that Hoeffding Adaptive Trees have not been studied as feature selectors in feature drifting scenarios, which may be cumbersome since the ADWIN drift detector is known for flagging a reasonable number of false positives, thus rendering the tree structure too volatile even during stationary regions of streams.

More recently, Barddal et al. (2016a) proposed a dynamic weighting scheme for the problem of classification over data streams. Their proposal dynamically computes the discriminative power of each feature using the symmetrical uncertainty scoring operator (Witten & Frank, 2005) over sliding windows. The discriminative power of each feature was used as a weighting factor in the prediction process of both k-Nearest Neighbors and Naive Bayes classifiers. This weighting scheme resulted in accuracy gains, while at the expense of reasonable additional processing time and memory usage rates. Furthermore, the weighted versions of these classifiers were used at the leaves of HAT to marginally improve its prediction rates, again at the expense of limited additional processing time.

At this point, it is important to highlight that the proposal presented in the current paper follows the same sliding window procedure to calculate the symmetrical uncertainty between the features and the class. Nevertheless, in contrast to the procedures in (Barddal et al., 2016a), these scores are used for feature selection instead of feature weighting. Details of the computation of symmetrical uncertainty over sliding windows and the proposed method are discussed in Section 5. We also highlight at this point that the proposed method is different from the one presented in Barddal et al. (2017), as the method proposed here computes statistics along sliding windows instead of landmarks and embeds redundancy checks.

Finally, it is also worth to mention that the study of feature drifts has also been recently investigated in the context of regression, such as in the work of Duarte and Gama (2017), yet, regression is out of the scope of this paper.

5. DISCUSS

This section introduces a novel dynamic filter for feature selection from data streams, namely *DynamIc SymmetriCal Uncertainty Selection for Streams* (DISCUSS). DISCUSS has as its core the symmetrical uncertainty scoring operator (Witten & Frank, 2005), which is used to quantify how important a feature is w.r.t. class prediction over a sliding window and whether a pair of features is redundant. Computing symmetrical uncertainty scores over data streams is not trivial as it was originally formulated for the batch setting. Section 5.1 is devoted to the formulas and discretization techniques required to allow symmetrical uncertainty computation along data streams. This section follows the framework earlier presented in Barddal et al. (2016a), where sliding window entropy and symmetrical uncertainty computation procedures have been used for feature weighting processes and is extended in this paper with the goal of feature selection. Next, Section 5.2 introduces DISCUSS, which has its merit-guided sequential feature selection procedure detailed in Section 5.3.

5.1. Background

Several metrics were proposed and used to quantify how "relevant" a feature is to predict class labels. Examples include Pearson's (Pearson, 1920) and R^2 coefficients of determination which measure the variance of class prediction, assuming a linear relationship. Alternatively, ideas from Information Theory are employed, usually the notion of entropy (Shannon, 1948). The main benefit of using Information Theoretic metrics is that they allow for the detection of both linear and non-linear correlations between variables.

Definition 5. (Entropy) Given a discrete random variable X, its entropy is given by:

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

where x_i iterates through all possible values of X.

Definition 6. (Conditional entropy) Conditional entropy quantifies the amount of information necessary to describe the outcome of a random variable Y given that the value of another random variable X is known. Eq. (6) depicts its computation, where $H(Y|X) = 0$ occurs if Y is completely determined by X, while $H(Y|X) = H(Y)$ holds if X and Y are independent random variables.

$$\begin{aligned} H(Y|X) &= -\textstyle\sum_{x_j}^{X} P[X = x_j] \times H(Y \mid X = x_j) \\ &= -\textstyle\sum_{x_j}^{X} P[X = X_j] \times \sum_{y_i}^{Y} P[Y = y_i \mid X = x_j] \times \\ &\quad \log_2 P[Y = y_i \mid X = x_j] \end{aligned} \tag{6}$$

One problem with entropy is that it is biased towards features with more distinct values. For instance, let us assume a data set that contains an unique identifier feature. In this case, each identifier would be associated with a single class value, and thus, the entropy would be optimal, i.e., zero. This is an extreme case, which indicates that the identifier is an uninteresting feature since it fails to provide any insights for predicting the class.

Entropy is the basis for more refined metrics. For instance, entropy can be used to compute Information Gain ($IG(X,Y) = H(X) - H(X|Y)$), a popular choice for learning decision trees. 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 $H(X) - H(X|Y) = H(Y) - H(Y|X)$ and this can be derived from $H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y)$.

However, similar to entropy, information gain is also biased towards features with more distinct values. Therefore, different metrics that compensate for this bias are preferred in feature selection, and an example is the symmetrical uncertainty (Nguyen, Woon, & Ng, 2014; Zamani-Dehkordi, Rakai, & Zareipour, 2017).

Definition 7. (Symmetrical uncertainty) The symmetrical uncertainty between two discrete random variables X and Y is given by:

$$\begin{aligned} SU(X,Y) &= 2\left[\frac{IG(X,Y)}{H(X)+H(Y)}\right] \\ &= 2\left[\frac{H(Y)-H(Y|X)}{H(X)+H(Y)}\right] \end{aligned} \tag{7}$$

Symmetrical uncertainty values lie within 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.

For this metric to be used in streaming scenarios, its computation should be both efficient and adaptive so that it allows methods to "forget" old concepts and swiftly adhere to the new ones. Luckily, the computation of symmetrical uncertainty depends on entropies, and these have been calculated along sliding windows in the works of Barddal et al. (2016a) and Sovdat (2014).

By definition, in sliding window models, only the most recent data obtained from a data stream is stored in a FIFO (first in, first out) data structure. Therefore, this structure only considers information from the current instant up to a certain period in the past (Silva et al., 2013). The organization and manipulation of objects inside this structure follow the principles of queue processing.

Given that, to update the symmetrical uncertainty for a feature D_i to the class Y, 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 updated following Algorithm 1 , while $H(Y|D_i)$ can be broken down into several conditional entropies in the $H(Y|D_i = q)$ form, which can also be computed by the same algorithm. The proofs for the formulas used in Algorithm 1 can be found at Sovdat (2014).

Algorithm 1: Entropy computation over sliding windows (Sovdat, 2014).

```
input : a sequence of values $x^1, \dots, x^t$ for a discrete random variable $X$.
output : be ready to provide the entropy $h$ at any time.
1 $n \leftarrow 0$ ;      /* the counter of values stored in the window. */
2 $n_i \leftarrow 0$ ; /* the counter of values stored in the $i^{th}$ partition of $X$ */
3 $h \leftarrow 0$ ;                          /* The entropy of $X$ */
4 Enqueueing of a value $x^t$ that belongs to the $i^{th}$ partition of $X$
5     $n \leftarrow n + 1$;
6     $n_i \leftarrow n_i + 1$;
7     $h \leftarrow \frac{n-1}{n}\left(h - \log_2 \frac{n-1}{n}\right) - \frac{n_i}{n}\log_2\frac{n_i}{n} + \frac{n_i-1}{n}\log_2\frac{n_i-1}{n}$;
8 Dequeueing of a value $x^t$ that belongs to the $i^{th}$ partition of $X$
9     $n \leftarrow n - 1$;
10    $n_i \leftarrow n_i - 1$;
11    $h \leftarrow \frac{n+1}{n}\left(h + \frac{n_i+1}{n+1}\log_2\frac{n_i+1}{n+1} - \frac{n_i}{n+1}\log_2\frac{n_i}{n+1}\right) + \log_2\frac{n}{n+1}$;
```

5.1.1. Discretizing numeric features

One of the drawbacks of adopting entropy-based metrics as a scoring operator is that numeric features must be discretized. Discretizing numeric variables in which their minimum and maximum values are *a priori* unknown is difficult, and thus, specific techniques shall be used for this purpose.

In this work, a two-layer histogram strategy is proposed as an extension to the Partition Incremental Discretization algorithm (PiD) (Gama & Pinto, 2006), hereafter referred as Partition Adaptive Discretization (PaD). In the first layer, PaD constructs and maintains equal-width bins that summarize the values provided by a stream. Given the partitions computed in the first layer, whenever a partition is accentuated compared to others, PaD's second layer of partitions is reconstructed, following an equal frequency strategy. In contrast to PiD, PaD updates its internal counters along a sliding window. This is required as bins should reflect the same data stored in DISCUSS' sliding window, which is used to keep entropy values up-to-date.

Algorithm 2 depicts the pseudocode for PaD. Similarly to PiD, PaD's first layer summarizes data, while the second layer constructs the final histogram.

Algorithm 2: Partition Adaptive Discretization (PaD) pseudocode, which is inspired by Gama's PiD (Gama & Pinto, 2006).

```
input  : a stream of numeric values $\mathcal{S}$ that correspond to a single
         feature $D_i$, the number of partitions $F$ and the sliding
         window size $w$.
output : be able to provide at any moment $F$ equal-width bins:
         $sLayer$.
1 Let $fLayer \leftarrow \emptyset$ be the histogram for the 1st layer;
2 Let $sLayer \leftarrow \emptyset$ be the histogram for the 2nd layer;
3 Let $V \leftarrow \emptyset$ be a sliding window;
4 foreach $x^t \in \mathcal{S}$ do
5     $mustReconstruct \leftarrow FALSE$ ;          /* flag for second layer reconstruction */
6     $V \leftarrow V \cup \{x^t\}$;
7     if $|V| > w$ then
8         Dequeue oldest element $x^{t-w}$ from $V$;
          /* Increment and decrement both layers with the arriving and dequeued values */
9         Update the 1st layer using $x^t$ and $x^{t-w}$;
10        Update the 2nd layer using $x^t$ and $x^{t-w}$ while finding the new
          counts of the updated partitions $c_a$ and $c_b$;
          /* Checks if we need to reconstruct the second layer */
11        if $c_a > (1+\alpha) \times T_{\max}$ or $c_a < (1-\alpha) \times T_{\min}$ or
          $c_b > (1+\alpha) \times T_{\max}$ or $c_b < (1-\alpha) \times T_{\min}$ then
12            $mustReconstruct \leftarrow TRUE$;
13    else if $|V| = w$ then
          /* first window is complete, and thus, both layers are constructed */
14        $buildFirstLayer()$;
15        $mustReconstruct \leftarrow TRUE$;
16    if $mustReconstruct$ then $buildSecondLayer()$;
```

PaD's first layer maintains an equal-width histogram compatible with the values present in a sliding window. The challenging decision here is how to determine an appropriate width for this layer, and this is solved after the sliding window is filled with data: given the sliding window of values V, the width will be equal to $\frac{1}{\max(V)-\min(V)}$. After the initial computation of the first layer, arriving values are used to increment the first layer histogram, while dequeued values are used to decrement the same histogram. Details on the construction of the first layer and how it is updated can be found in Gama and Pinto (2006).

The second layer is reconstructed whenever a "significant" change is detected in the first layer. Given two thresholds $T_{\min}$ and $T_{\max}$, which are the minimum and maximum frequencies of all existing partitions, the second layer of PaD is reconstructed whenever a partition with a frequency above $(1+\alpha) \times T_{\max}$ or below $(1-\alpha) \times T_{\min}$ is observed. As noticed by Gama and Pinto (2006), the choice of α is not trivial, however, $\alpha = 1\%$ seemed reasonable in PiD's original experiments, and the same value is adopted here.

The construction of PaD's second layer is straightforward. Given some partitions F and the sliding window size w, each partition in the second layer will possess approximately $\frac{w}{F}$ frequency and occurrences of the same value are stored in the same partition. The second layer is constructed linearly by traversing and aggregating the first layer bins until a $\frac{w}{F}$ frequency is reached. More details about the construction of the second layer and how it is flagged for reconstruction can be found in Gama and Pinto (2006).

In the following experiments, the classic rule of thumb that numeric features are discretized into $F = 10$ equal-frequency partitions is followed (Boulle, 2005), but a deeper analysis on this choice is presented in Section 6.3.

5.2. Overview

In this section, we present the overall functioning of the DISCUSS filter. The pseudocode of DISCUSS is reported in Algorithm 3 , which consists of (i) updating the necessary entropies for symmetrical uncertainty computation between each feature and the class (lines 9–11); (ii) a baseline comparison (lines 13–16); (iii) a selection procedure; and (iv) a redundancy check (both detailed in Algorithm 4).

As input, DISCUSS receives a data stream $\mathcal{S}$ and the buffer size w. First, DISCUSS retrieves the first w instances from $\mathcal{S}$ to increment the symmetrical uncertainty of each feature w.r.t. the class, i.e., $SU(D_i, Y), D_i \in \mathcal{D}$ (lines 26–30). This process is repeated until the DISCUSS buffers w instances, and then it selects the first subset of features $\mathcal{D}'$ and the baseline feature $D_{baseline}$ is chosen (line 30). From this point on, DISCUSS operates over its buffer W as a sliding window (lines 6–25), where the symmetrical uncertainty scores $SU(D_i, Y), D_i \in \mathcal{D}$ are updated as new instances become available and others are forgotten (lines 7–11). We also highlight that the updates in the symmetrical uncertainty scores between each feature and the class can be performed in parallel, yet, the pseudocode and implementation discussed in this paper are sequential. This is relevant since DISCUSS can be applied to high dimensional data and its computational overhead regarding processing time (see Section 6, and more specifically, the SPAM experiment in Section 6.5) can be significantly reduced.

Followed by the symmetrical uncertainty updates, features are compared to verify whether a feature that was irrelevant has now surpassed the baseline feature $D_{baseline}$ or a feature that was relevant had its discriminant power decreased and is now below $D_{baseline}$ (lines 12–16). If either of the conditions above is observed, a new feature selection is triggered (lines 17–25), or otherwise, the classification model is updated using the arriving instance (lines 24–25). Whenever new features are selected, the classifier is also reset and re-trained with the instances stored in the buffer using only the newly-selected features (lines 19–22). Finally, during the prediction step, DISCUSS returns the current subset of selected features $\mathcal{D}'$ and filters the instance before the actual prediction is performed by the classifier.

Algorithm 3: Pseudocode of the proposed filter during the training phase.

```
input : a data stream $\mathcal{S}$ that provides instances $i^t = (\vec{x}^t, y^t)$ and a buffer
        size $w$.
1  Let $W \leftarrow \emptyset$ be a queue that maintains a sliding window with length $w$;
2  Let expert be a classifier;
3  $\mathcal{D}' \leftarrow \emptyset$ ;                              /* Subset of selected features */
4  $D_{baseline} \leftarrow NULL$ ;                                   /* Baseline feature */
5  foreach $i^t = (\vec{x}^t, y^t) \in \mathcal{S}$ do
6      if $|W| = w$ then
7          $W \leftarrow W \cup \{i^t\}$;
8          Increment the $SU(\cdot,\cdot)$ values $\forall D_i \in \mathcal{D}, SU(D_i, Y)$ with $i^t$;
9          if $|W| > w$ then
10             Dequeue the oldest instance $i^{t-w-1}$ from $W$;
11             Increment the $SU(\cdot,\cdot)$ values $\forall D_i \in \mathcal{D}, SU(D_i, Y)$ given $i^{t-w-1}$;
12         $flag \leftarrow FALSE$;
13         foreach $D_i \in \mathcal{D} \setminus \{D_{baseline}\}$ do
14             if $(SU(D_i, Y) > SU(D_{baseline}, Y)$ and $D_i \notin \mathcal{D}')$ or
               $(SU(D_i, Y) < SU(D_{baseline}, Y)$ and $D_i \in \mathcal{D}')$ then
                   /* This condition is satisfied when a feature either
                      (i) becomes 'relevant' and surpasses $D_{baseline}$, or
                      (ii) turns 'irrelevant' and has its SU w.r.t. the
                      class now below $D_{baseline}$'s                          */
15                 $flag \leftarrow TRUE$;
16                 break;
17         if $flag$ then
18             $(\mathcal{D}', D_{baseline}) \leftarrow selectFeatures(\mathcal{D})$ ;   /* Selects new features
               and defines a new baseline feature based on Algorithm 4
               */
19             expert.reset() ;                          /* Resets the learner */
               /* Starts the learning of a new model with the instances
                  in buffer given the newly selected features in $\mathcal{D}'$    */
20             foreach $i' \in W$ do
21                 $i' = extract(i', \mathcal{D}')$;
22                 $expert.train(i')$;
23         else
24             $i' \leftarrow extract(i^t, \mathcal{D}')$;
25             $expert.train(i')$;
26     else
           /* Condition met during the first $w$ instances obtained from
              $\mathcal{S}$.                                                    */
27         $W \leftarrow W \cup \{i^t\}$;
28         Increment the $SU(\cdot,\cdot)$ values $\forall D_i \in \mathcal{D}, SU(D_i, Y)$ with $i^t$;
29         if $|W| = w$ then
               /* Selects the first subset of relevant features given
                  the instances stored in $W$ and also sets a baseline $\mathcal{D}'$
                  that will be used during the main loop.                  */
30             $(\mathcal{D}', D_{baseline}) \leftarrow selectFeatures(\mathcal{D})$;
```

5.3. Merit-Guided sequential feature selection

DISCUSS adopts a merit-guided sequential feature selection strategy, which is inspired by the Correlation-based Feature Selection (CFS) algorithm introduced in Hall (2000).

Definition 8. (Merit of a feature subset) The merit of a feature subset $\mathcal{D}'$ with cardinality n is given by Eq. (8), where the numerator describes how predictive of a class a group of features is, and the denominator depicts how much redundancy there is amongst

Algorithm 4: Selection scheme for the merit-guided selection procedure.

```
input : the feature set $\mathcal{D}$ and the number of attempts for merit
        improvement $nMaxAttempts$.
output : the selected features $\mathcal{D}'$ and the baseline feature $D_{baseline}$
1  Sort $\mathcal{D}$ in descending order of $SU(\cdot, Y)$;
2  $\mathcal{D}' \leftarrow \emptyset$ ;                        /* set of selected features */
3  $M \leftarrow 0$ ;                    /* the best metric value obtained */
4  $sRelevances \leftarrow 0$ ;                          /* sum of relevances */
5  $sRedundancies \leftarrow 0$ ;                       /* sum of redundancies */
6  $nAttempts \leftarrow 1$ ;       /* counter of attempts for merit improvement */
7  foreach $D_i \in \mathcal{D}$ do
8      $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{D_i\}$;
9      $sRelevances \leftarrow sRelevances + SU(D_i, Y)$;
10     foreach $D_j \in \mathcal{D}'$ do
11         $sRedundancies \leftarrow sRedundancies + SU(D_i, D_j)$;
12     $n \leftarrow |\mathcal{D}'|$;
13     $merit \leftarrow \frac{sRelevances}{\sqrt{n+2\times sRedundancies}}$;
14     if $merit > M$ then
15         $M \leftarrow merit$;
16         $nAttempts \leftarrow 1$;
17     else
18         $nAttempts \leftarrow nAttempts + 1$;
19         $\mathcal{D}' \leftarrow \mathcal{D}' \setminus \{D_i\}$;
           /* Restore the relevance and redundancy scores                */
20         $sRelevances \leftarrow sRelevances - SU(D_i, Y)$;
21         foreach $D_j \in \mathcal{D}'$ do
22             $sRedundancies \leftarrow sRedundancies - SU(D_i, D_j)$;
23     if $nAttempts = nMaxAttempts$ then break;
24 return $(\mathcal{D}', D_{baseline} = \mathcal{D}'.last())$ ; /* the last feature is used as baseline
   */
```

them (Hall, 2000).

$$M(\mathcal{D}') = \frac{\sum_{D_i}^{\mathcal{D}'} SU(D_i, Y)}{\sqrt{n + 2 \sum_{\substack{D_i, D_j \\ i \neq j}}^{\mathcal{D}'} SU(D_i, D_j)}} \tag{8}$$

The merit-guided strategy embedded within DISCUSS, presented in Algorithm 4, receives as input a single parameter, which is the number of consecutive attempts (*nMaxAttempts*) that the algorithm is allowed to fail when trying to increase the merit of the selected feature subset. This hyper-parameter prevents DISCUSS from behaving like a conventional hill-climbing algorithm. By allowing the selection procedure to fail a limited amount of times, it prevents the algorithm from getting stuck in a local minimum.

First, the list of features is sorted in descending order given their symmetrical uncertainty scores. The main loop (lines 7–23) retrieves at each iteration the best-ranked feature, which is then temporarily added to the subset of selected features $\mathcal{D}'$. Next, both the sums of relevances (*sRelevances*) and redundancies (*sRedundancies*) are incremented with the respective symmetrical uncertainty values (lines 9–11). These updated summations allow the computation of the current merit obtained with the addition of D_i (lines 12–13). If the merit obtained surpasses the best merit obtained thus far (M), the merit is updated, and the number of attempts (*nAttempts*) is reset (lines 14–16). Otherwise, D_i is eliminated from the set of selected features, and the number of attempts is incremented (lines 17–22).

If *nAttempts* reaches the hyper-parameter *nMaxAttempts* (line 23), the algorithm halts and returns the current subset of selected features $\mathcal{D}'$ and the baseline feature is set as the last feature selected (line 24). To avoid prohibitive computational overheads, $nMaxAttempts = 3$ was chosen since it needs the least amount of computational resources while still enabling a fair number of unsuccessful attempts.

Finally, it is important to emphasize that the proposed method acts as a filter, since is scores each feature w.r.t. the class without checking the impact on accuracy rates of a classifier or inside the model learning (such as in the decision tree branching process).

Table 1
Classifiers and parameters used in experiments.

Classifier	Parameter	Value
*k*NN	Number of neighbors - *k*	9
	Window size - *w*	500
VFDT and HAT	Grace period - *gp*	200
	Split criterion	Information gain

6. Analysis

In this section, we analyze the proposed method against traditional data stream classifiers in both synthetic and real-world datasets. First, in Section 6.1 we describe the experimental protocol adopted, including: (i) classifiers tested, (ii) synthetic and real-world datasets used, (iii) evaluation metrics, (iii) validation procedure, and (iv) statistical tests adopted. Next, we show in Sections 6.2 and 6.3 how window size and discretization hyper-parameters affect the final outcomes. The results obtained by DISCUSS in synthetic and real-world datasets are reported in Sections 6.4 and 6.5, respectively. Finally, a theoretical discussion about when and why DISCUSS fails is reported in Section 6.6.

6.1. Experimental protocol

In the following experiments, DISCUSS is used in conjunction with *k*NN, Naive Bayes, Very Fast Decision Tree (VFDT) (Domingos & Hulten, 2000) and Hoeffding Adaptive Tree (HAT) (Bifet & Gavaldà, 2009) classifiers. All of the classifiers mentioned above are available as part of the Massive Online Analysis (MOA) framework (Bifet, Holmes, Kirkby, & Pfahringer, 2010), which has also been used to implement our proposed method. We highlight that more robust methods such as Adaptive Random Forests (Gomes et al., 2017) and Oza's Adaptive Boosting (Oza, 2005) could be used as base learners here, yet, they have not been tested since random forests already perform feature selection internally, and adaptive boosting methods such as Oza's Boost also account for internal feature selection as Hoeffding Trees are often employed as the core base learner. Also, it is important to emphasize that both random forests and boosting methods are ensembles, so it is hard to determine whether they are successful or not since many components affect their outcomes, such as drift detection and adaptation method, diversity induction process, and vote combination scheme.

The parameters adopted for *k*NN, Very Fast Decision Tree and Hoeffding Adaptive Tree classifiers are reported in Table 1. It is important to highlight that Naive Bayes is not listed as it does not have any parameters to be set. The parameters used in DISCUSS are the following: a window size of 300 and the number of partitions equal to 10 for discretization. The rationale behind the choice of values for the window size and the number of partitions will be discussed in Sections 6.2 and 6.3, respectively. At this point, we must clarify that these parameters were not tuned to achieve higher accuracy rates with the goal of avoiding over-optimistic results.

To evaluate whether a learning algorithm can work in different scenarios, it is necessary to assess its performance across data streams with different characteristics. DISCUSS was applied to several synthetic data streams which are known for being able to simulate feature drifts (Barddal et al., 2017) and on real-world

Table 2
Synthetic data stream experiments. The number of irrelevant features is the difference between the total number of features and the relevant ones.

Experiment identifier	# of Features	# of Relevants	# of Redundants (Linear/RBF/Cosine)
BG1(A/G)	100/200/500	4 ± 1	15/0/0
BG2(A/G)	100/200/500	3	15/0/0
BG3(A/G)	100/200/500	3	15/0/0
RTG(A/G)	100/200/500	5	5/5/5
SEA(A/G)	100/200/500	2	5/5/5

datasets. The choice of working on synthetic streams is due to the possibility of configuring drifts types and locations during execution. Even though real-world datasets do not allow researchers to pinpoint whether feature drifts occur and when, they were still used to verify if DISCUSS is feasible in complex and real scenarios.

Regarding synthetic experiments, we followed the protocol and generators proposed in Barddal et al. (2017) and performed the experiments depicted in Table 2. Every execution contains 1 million instances, possesses 2 equally-distributed feature drifts (positioned at 333,333 and 666,666), are binary and balanced classification problems, and only a small fraction of features in each experiment are relevant (up to 10%). Experiments marked with an (A) are abrupt while those with a (G) mark are gradual with a drift window length $\Delta = 10,000$. Feature drifts occur in all of the presented experiments since the prior and posterior concepts rely on a different subset of relevant features.

First, inspired by the work of Hall (2000), the **Binary Data with Feature Drift Generator (BG-FD)** creates instances composed of boolean ({0, 1}) features (Barddal et al., 2017). BG-FD has three functions: BG1-FD, BG2-FD, and BG3-FD. In BG1-FD, presented in Eq. (9), 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 (Eq. (10)) and BG3-FD (Eq. (11)), the size of the relevant subset of features is fixed, where $\mathcal{D}^* = \{D_\alpha, D_\beta, D_\epsilon\}$.

$$y = \begin{cases} 1, & \text{if } \bigwedge_{D_i \in \mathcal{D}^*} \vec{x}_i \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

$$y = \begin{cases} 1, & \text{if } (\vec{x}_\alpha \wedge \vec{x}_\beta) \vee (\vec{x}_\alpha \wedge \vec{x}_\epsilon) \vee (\vec{x}_\beta \wedge \vec{x}_\epsilon) \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

$$y = \begin{cases} 1, & \text{if } (\vec{x}_\alpha \wedge \vec{x}_\beta \wedge \vec{x}_\epsilon) \vee (\neg\vec{x}_\alpha \wedge \neg\vec{x}_\beta \wedge \neg\vec{x}_\epsilon) \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

The **Random Tree Generator with Feature Drift (RTG-FD)** builds a decision tree by randomly performing splits on features and assigning a random class label to each leaf. Instances are created by generating a random valued $\vec{x}$ and traversing the tree for its corresponding label. This generator is extended in this work to allow that given a random $\mathcal{D}^* \subset \mathcal{D}$ is relevant, where $|\mathcal{D}^*| < |\mathcal{D}|$ is a user-given parameter. Therefore, the remaining features generated will be either irrelevant or redundant. Barddal, Gomes, de Souza Britto Jr., and Enembreck (2016b) proposed a data stream generator that extends the Streaming Ensemble Algorithm (SEA) generator (Street & Kim, 2001) namely **SEA-FD**, where the suffix stands for *Feature Drift*. It 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: $\mathcal{D}^* = \{D_\alpha, D_\beta\}$. An instance is labeled as positive if $\vec{x}_\alpha + \vec{x}_\beta \leq \theta$ and negative otherwise. To simulate feature drifts, each of the concepts will rely on different features, since D_α and D_β are randomly selected in each concept.

Finally, the generators introduced above produce concepts where most (if not all of them) features are used to determine the class outcome of an instance. In their original form, they do not allow for the introduction of additional irrelevant or redundant features. To generate more useful and challenging learning scenarios for studying feature selection, all generators were modified to generate a user-specified mix of relevant, irrelevant, and redundant features. To add **irrelevant features** into generators we merely incremented $\mathcal{D}$ with numeric or categorical features. In the first case, values are sampled from a uniform distribution bounded in [0; 1], regardless of the class. Following this trend, irrelevant categorical features are generated with m different values, which are also equally likely. In the following experiments we set $m = 10$. As a result, we have tested experiments with 100, 200 and 500 dimensions. These experiments were tailored to check how base classifiers and DISCUSS behave in high-dimensional scenarios.

Table 3
Real-world data experiments.

Experiment identifier	# of Instances	# of Features
IADS	3279	1558
NOMAO	34,465	120
SPAM	9324	39,916

Similarly, **redundant features** were added to experiments following three simple strategies. The first is a copy with a perturbation factor scheme, in which the value of D_j is generated based on D_i by adding a factor $p \in [0; 1]$. If D_i is numeric, then $\vec{x}_j = \vec{x}_i \pm p \times \delta$ with $\delta = (\max D_i - \min D_i)$ is used, while if D_i is categorical, then $\vec{x}_j$ is set to $\vec{x}_i$ with $1 - p$ chance or otherwise randomly distributed across the remaining possible values. Either way, this strategy leads to a linear correlation, with the Pearson coefficient decaying exponentially with the growth of p. For the following experiments, we use $p = .15$. The Radial Basis Function (RBF) strategy generates numeric values by projecting $\vec{x}_i$ into a Gaussian distribution. Working under the assumption that D_i's distribution is uniform (as it holds for all generators earlier described), its expectation is given by $E(D_i) = \frac{\delta}{2}$. Given $E(\cdot)$, the value of D_i can be projected using Eq. (12) to obtain $\vec{x}_j$. The last strategy is similar and applies the cosine function to create redundant features as depicted in Eq. (13).

$$\vec{x}_j = (\vec{x}_i - E(D_i))^2 \tag{12}$$

$$\vec{x}_j = \cos\left(360 \frac{\vec{x}_i}{\delta}\right) \tag{13}$$

In addition to synthetic data, DISCUSS was also analyzed with real-world datasets. Even though it is unclear whether the dataset contains any drifts, these were still used for testing how DISCUSS behaves in potential applications. In Table 3 we can see the main characteristics of the datasets used. The Internet Advertisements (IADS) dataset (Kushmerick, 1999) represents the problem of identifying advertisements on Internet pages. It contains 1558 features representing phrases and the geometry of the 3279 URLs. The Nomao dataset (NOMAO) represents a binary classification problem with 120 features and 34,465 instances (Candillier & Lemaire, 2012). Finally, the Spam Corpus (SPAM) dataset was developed in Katakis, Tsoumakas, and Vlahavas (2006) as a result of a text mining process on an online news dissemination system with the goal of classifying e-mails as spam or ham.

In the experiments reported, accuracy is calculated following the Prequential test-then-train evaluation procedure (Gama & Rodrigues, 2009), which allows the monitoring of classifier performance over time according to a sliding window. For all synthetic experiments, the prequential window size was set to 5% of the stream and the results reported are averages obtained after 30 independent runs, while experiments using real-world data were evaluated every 10% of the length of the stream and with a single run since DISCUSS does not present stochastic behavior. Besides measuring accuracy, it is crucial that data stream mining algorithms perform both rapidly and within memory boundaries. In the following experiments, processing time is reported as the time that the algorithms spend in the processor (in seconds), namely CPU Time; while memory usage is presented in RAM-Hours, where 1 RAM-Hours equals 1GB of RAM used per hour. All experiments were conducted using the MOA framework on an Intel Xeon CPU E5649 @ 2.53GHz $\times 8$ based computer running CentOS with 20GB of memory. To provide statistical confidence to the results obtained, Wilcoxon's (Wilcoxon, 1945) method was used to test pairs of hypotheses and Friedman, and Nemenyi tests (Demsar, 2006) were used to perform multiple pair-wise comparisons.

6.2. Selecting an appropriate window size

As any other window-based approach for learning from data streams, the proposed symmetrical uncertainty scoring operator used in DISCUSS also requires the definition of a proper window size w. The size of a window should be as small as possible to allow quick drift recognition and adaptation (*plasticity*), but at the same time, large enough so it correctly reflects the distribution of stable regions of a stream (*stability*) (Webb et al., 2016). Instead of tuning w based on accuracy, which could vary depending on the classifier used, we chose to pick a window size according to the error rates obtained by the symmetrical uncertainty scoring operator in batch and sliding window versions. The rationale here is to compare the symmetrical uncertainty values obtained by our sliding window variant against those obtained over an entire stable data stream. Following this rationale, an experimental window size evaluation strategy was built as follows.

First, several synthetic stationary data streams were generated, each with 100,000 instances. Given each stream, all of its features had their features' symmetrical uncertainty to the class computed in batch mode to serve as a gold standard. Later, different sliding window sizes $w \in [10; 1000]$ were evaluated by computing the deviation between the obtained symmetrical uncertainty values and the gold standard for each feature. Given these results, it is possible to verify the smallest window size value that also satisfies a feasible symmetrical uncertainty deviation compared to the entire sample distribution.

Results are presented in Fig. 1a, which shows that the error rates between the obtained symmetrical uncertainty scores over sliding windows quickly decay with increasing w. In practice, very small values, e.g., 100 to 150, already allow a good symmetrical uncertainty adherence regardless of the experiment domain during the evaluated stationary experiments. Therefore, slightly higher values around 200 or 300 (Fig. 1b) are expected to provide a fair trade-off between drift adaptation and correct symmetrical uncertainty rendering in stable regions.

Moreover, this experiment also shows that the symmetrical uncertainty values provided by the proposed scoring operator are excellent approximations, with errors of less than 10^{-3} when $w \geq 250$. Following these results, $w = 300$ was selected for further accuracy and processing time analyses. We note, however, that this is an important parameter that deserves to be tuned depending on each data stream domain DISCUSS will be applied to.

6.3. The impact of different numbers of partitions during discretization

In addition to defining an appropriate window size, it is also essential to assess the impact of different numbers of partitions during discretization. The procedure adopted here tracks the performance of DISCUSS regarding accuracy, processing time and memory usage. Only RTG and SEA datasets contain numerical features, thus only these are affected by discretization. For the sake of brevity, the results presented below are averages obtained during the experiments across different classifiers. At this point, it is important to mention that the behavior observed for each classifier was the same, i.e., more partitions resulted in higher processing times and increased memory consumption.

Figs. 2 and 3 present the impact of three different numbers of partitions (bins) used during discretization regarding accuracy, processing time and memory usage for the RTG and SEA experiments. From these plots, we can see that the accuracy results improve by using fewer partitions. As evaluated in Pfahringer, Holmes, and Kirkby (2008), this is unexpected at first, since more bins often tend to better represent the data, although ultimately leading to overfitting and therefore worse rates of generalization.

In addition to the trend observed for accuracy, the processing time and memory usage results also present an important fact: more partitions lead to much higher computational time and memory usage rates. Evidently, this is expected since storing and updating more partitions in memory is computationally more intensive for DISCUSS than storing a smaller amount of them. As a result, picking a small number of partitions, such as 10, follows the guidelines promoted in Pfahringer et al. (2008), but also have empirical evidence obtained for the proposed method.

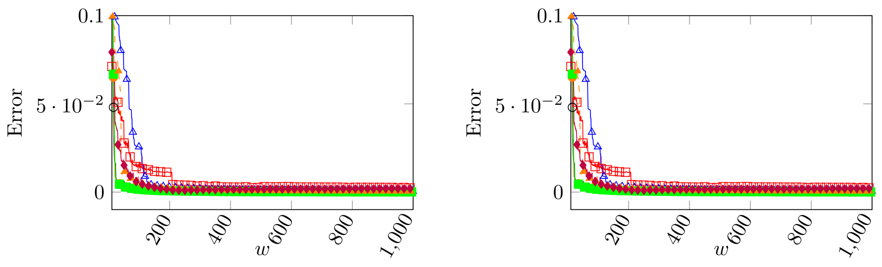
6.4. Synthetic experiments

In this section, we compare the accuracy results obtained by DISCUSS and compare them to the original classifiers and also against the Hoeffding Adaptive Tree (HAT) (Bifet & Gavaldà, 2009). Since all streams are class-balanced, accuracy will not be biased towards any specific class, and thus, it is valid for our analysis. Figs. 4–8 present the accuracy obtained for all experiments, including both abrupt and gradual variants, while Table 4 presents the average accuracy rates obtained.

The results obtained for BG1, BG2 and SEAFD experiments show promising improvements, as all classifiers benefit from DISCUSS in BG1 (Fig. 4), and NB and *k*NN are improved in both BG2 and SEAFD scenarios (Figs. 5 and 8). In these experiments, the NB classifier is highlighted, since the average accuracy gain is of 8.35% in BG1, 15.68% for BG2 and 5.71% in SEAFD scenarios.

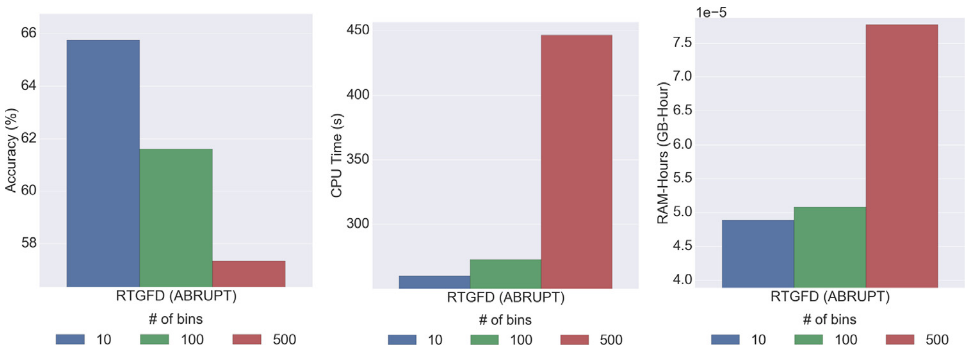
The reason behind VFDT presenting smaller improvements is inductive bias, where different classifiers operate differently when fed with the same features. This is a trend across all experiments, where the VFDT does not benefit, or benefits too little, from the selection provided by DISCUSS. This occurs because a decision tree is already a feature selection process that can capture feature interactions, which is a drawback of DISCUSS that will be further discussed in Section 6.6.

The results for BG3 and RTGFD experiments (Figs. 6 and 7) are quite similar to each other, where DISCUSS fails to select features appropriately, and as a result, classifiers end up presenting accuracy rates close to the baseline of 50% for BG3 and above 60% for the RTG experiment. A detailed discussion about why DISCUSS fails in these experiments is presented in Section 6.6.

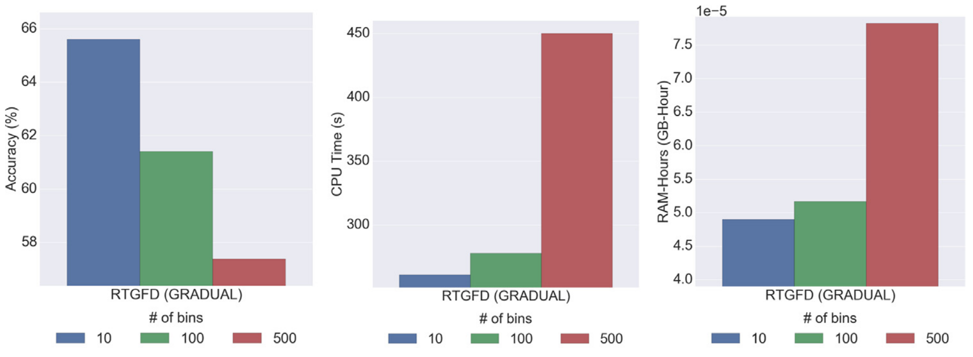


(a) Variation of w between 10 and 1000. (b) Variation of w between 10 and 200.

Fig. 1. Average error between the symmetrical uncertainty computed over the whole data set and the one provided by the sliding window version.



(a) RTG - ABRUPT



(b) RTG - GRADUAL

Fig. 2. Impact of different numbers of partitions for the RTG experiments.

Table 4
Accuracy results (%) obtained by using DISCUSS with a window size $w = 300$ against original base learners. Results in bold stand for the best results obtained for each experiment, while underlining indicates the best result per classifier, either with or without DISCUSS.

Experiment	NB	DISCUSS NB	kNN	DISCUSS kNN	VFDT	DISCUSS VFDT	HAT	DISCUSS HAT
BG1 (A)	78.49	87.13	83.84	84.58	86.13	87.05	**89.88**	87.17
BG1 (G)	78.42	86.85	83.66	84.45	86.14	86.85	**89.63**	86.96
BG2 (A)	67.38	83.10	74.96	80.73	79.56	77.70	**90.00**	79.18
BG2 (G)	67.37	83.00	74.83	80.49	79.54	77.56	**89.49**	79.16
BG3 (A)	56.26	55.60	70.08	54.85	80.44	55.56	**89.73**	55.40
BG3 (G)	56.29	55.78	69.92	54.90	80.46	55.64	**89.10**	55.55
RTGFD (A)	61.80	64.64	60.14	64.88	75.29	70.68	**91.59**	72.94
RTGFD (G)	61.77	64.48	60.12	64.96	75.14	70.65	**88.98**	72.82
SEAFD (A)	79.80	85.78	72.66	87.82	87.00	85.78	**88.59**	85.69
SEAFD (G)	79.79	85.23	72.71	87.05	87.00	85.16	**88.55**	85.16

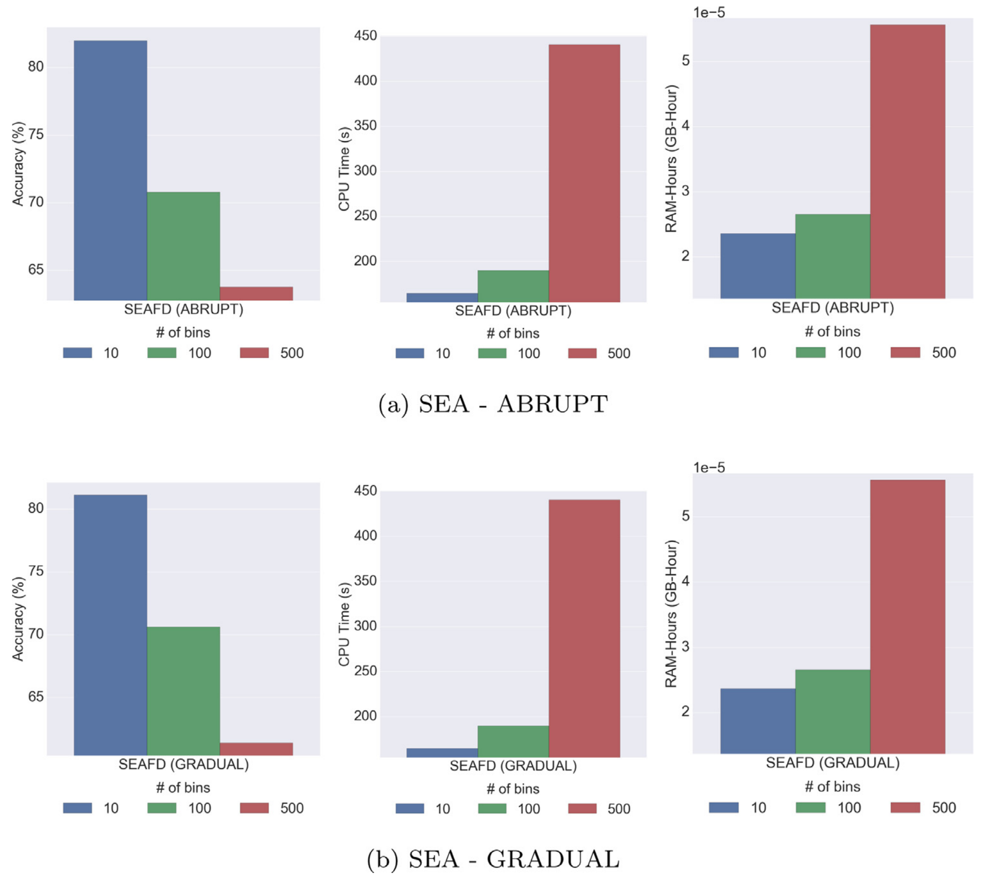


Fig. 3. Impact of different numbers of partitions for the SEA experiments.

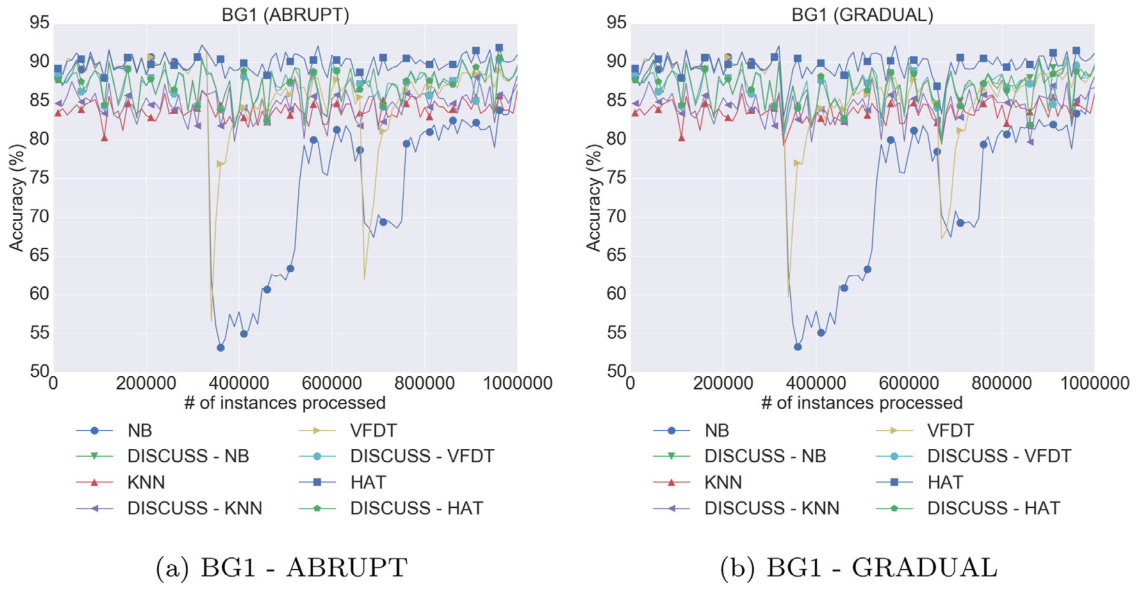
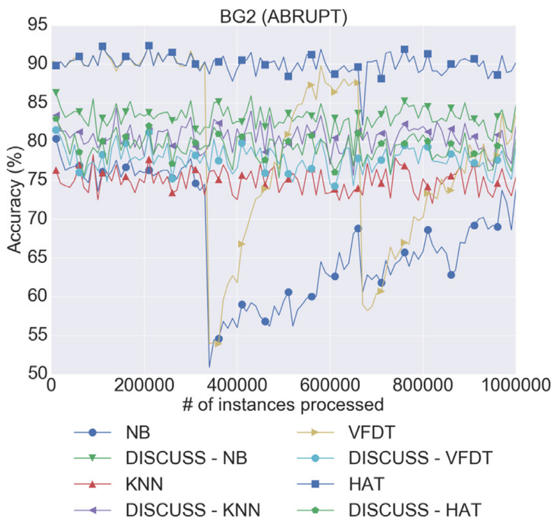


Fig. 4. Accuracy (%) obtained during the BG1 experiments.

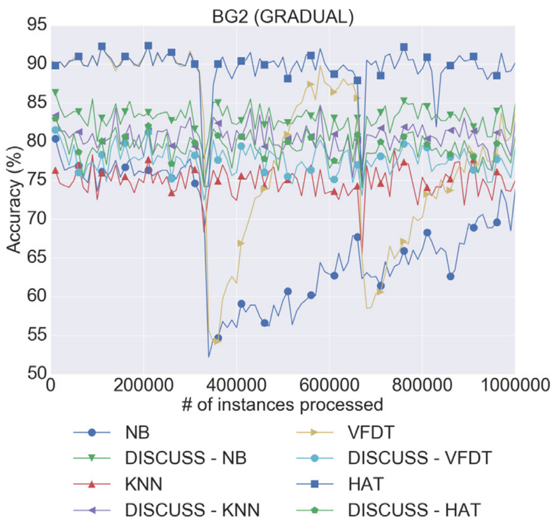
Focusing on the average results, depicted in Table 4, it is clear that all classifiers benefit from the selection made by DISCUSS at some point. It is also clear that none of these classifiers was ever able to surpass the Hoeffding Adaptive Tree. Even though DISCUSS is unable to help classifiers to beat the Hoeffding Adaptive Tree in accuracy rates, we expect DISCUSS to serve as a baseline for future proposals for dynamic feature selection from data streams.

To provide statistical background to these claims, Wilcoxon's paired test was used to compare the results obtained by each classifier against its combination with DISCUSS, while Friedman and Nemenyi tests were used to verify if any statistically significant differences occurred between all methods.

As a result, we found that the usage of DISCUSS improves NB's accuracy rates with a 99% confidence level, while significant improvements are only found for kNN if one assumes a 95% con-

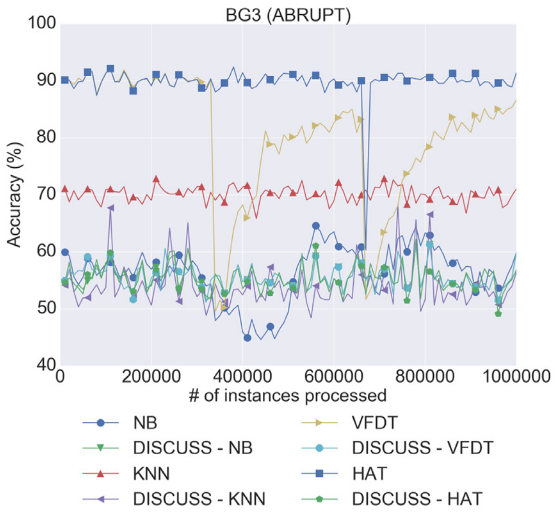


(a) BG2 - ABRUPT

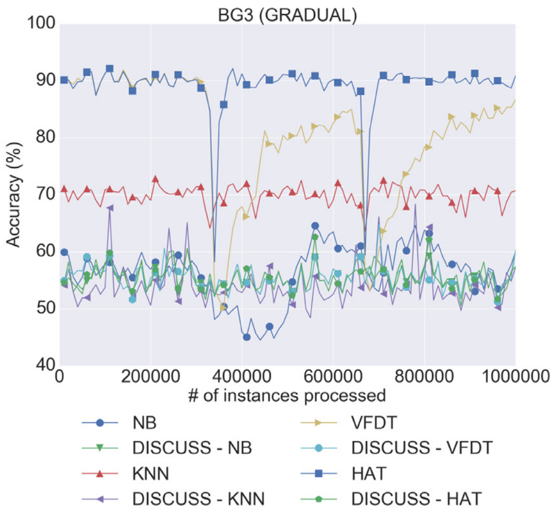


(b) BG2 - GRADUAL

Fig. 5. Accuracy (%) obtained during the BG2 experiments.



(a) BG3 - ABRUPT



(b) BG3 - GRADUAL

Fig. 6. Accuracy (%) obtained during the BG3 experiments.

fidence level. Finally, as a result of Friedman's and Nemenyi's tests (Fig. 9), we concluded that HAT and VFDT are significantly superior to the other methods also assuming a 95% confidence level.

Computational resources. Traditional feature selection is known for decreasing both processing time and memory consumption rates in batch learning. However, there is no guarantee that the same would hold in streaming scenarios since there is the need to continually update discriminative power metrics as the stream progresses (Naidu, Dhenge, & Wankhade, 2014). To correctly evaluate the efficiency of a feature selection algorithm in streaming scenarios, one must accumulate the feature selection processing and the classifier's training and prediction times. This way, it is possible to verify if the overhead of computing the scoring operators and selecting attributes can be justified by sufficiently decreasing the complexity, or in the unlikely case of an increase, to determine if the overhead is still feasible. This section compares the processing time and memory usage of DISCUSS' when applied to *k*NN, NB, VFDT and HAT classifiers.

Table 5 presents the processing time results obtained during experiments. For all experiments using NB and most using VFDT, processing times increased when associated to DISCUSS by 193.33% and 13.33%, respectively. This occurs for two reasons: (i) the extra-processing time for keeping the scores of all features plus the time necessary to train classifiers is greater than merely training a classifier with all features (despite whether they are good or bad); and (ii) the time spent on retraining classifiers also introduces an overhead. Even though these increases are statistically significant and may seem prohibitive at first sight, we recommend parsimony over their analysis since the processing times are still feasible in all scenarios. The results obtained for the *k*NN classifier are exceptions, where we see that the usage of DISCUSS has improved the processing time of the experiments by 12.63%. These results are interesting since DISCUSS decreased the complexity (number of features) of the stream sufficiently to justify the overhead of comput-

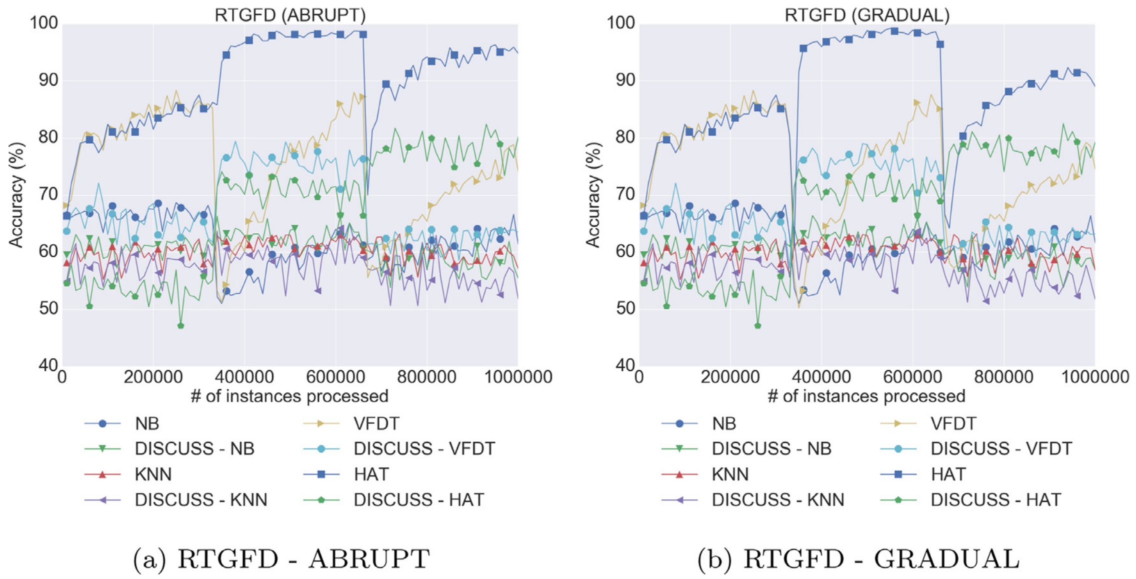


Fig. 7. Accuracy (%) obtained during the RTGFD experiments.

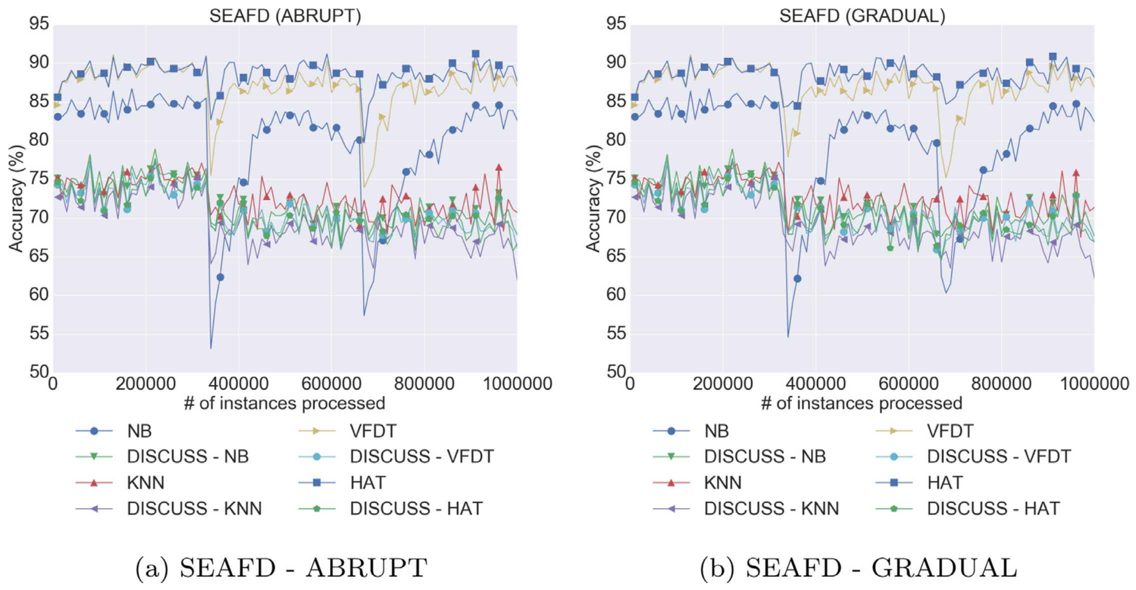


Fig. 8. Accuracy (%) obtained during the SEAFD experiments.

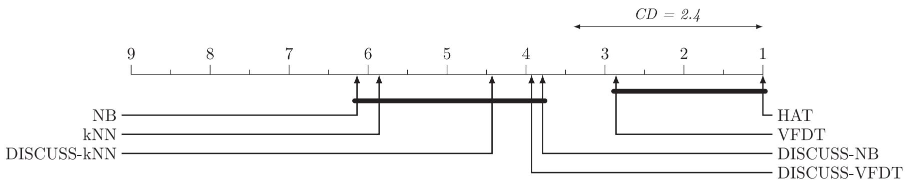


Fig. 9. Nemenyi critical differences plot for accuracy results.

ing the symmetrical uncertainty of each attribute w.r.t the class over a sliding window and selecting features given feature drifts.

Finally, the results for memory consumption are presented in Table 6, where only Naive Bayes is negatively impacted by DISCUSS, while both kNN and VFDT benefit from it. The increases obtained here are expected since DISCUSS requires storing (i) a sliding window of instances, (ii) structures for computing the necessary entropies, and (iii) discretization structures if the stream contains numeric data. On the other hand, these increases are justified when applied to *k*NN since the instances stored in the sliding window after feature selection, are much smaller when compared to the original ones. Also, the same rationale can be used to explain the memory usage results obtained for VFDT, because the tree is fed with fewer features, and as a result, it performs less splits causing the tree to be smaller. Furthermore, the costs for calculating splits in the leaves is decreased, such as the cost for maintaining the Naive Bayes predictors at the leaves is also decreased, once we are now dealing with reduced dimensionality. Once again, despite the increases being statistically significant, discretion is advised since such increases may not be prohibitive in most scenar-

Table 5
Processing time (s) results obtained by using DISCUSS with a window size $w = 300$ against original base learners. Results in bold stand for the best results obtained for each experiment, while underlining indicates the best result per classifier, either with or without DISCUSS.

Experiment	NB	DISCUSS NB	kNN	DISCUSS kNN	VFDT	DISCUSS VFDT	HAT	DISCUSS HAT
BG1 (A)	**5.15**	17.92	494.96	397.99	10.62	15.87	16.35	17.06
BG1 (G)	**4.82**	17.66	494.85	398.43	10.33	17.82	16.16	17.50
BG2 (A)	**5.04**	17.66	488.89	391.61	11.36	17.95	15.09	20.79
BG2 (G)	**4.82**	17.94	494.37	398.08	11.11	17.65	14.54	20.57
BG3 (A)	**5.04**	17.86	496.79	400.08	11.79	15.47	16.02	16.56
BG3 (G)	**4.71**	17.46	497.44	400.63	11.39	15.29	17.61	16.15
RTGFD (A)	**32.74**	50.39	537.10	583.15	58.63	41.19	54.57	67.21
RTGFD (G)	**32.99**	50.64	538.33	580.64	58.09	44.45	60.54	67.21
SEAFD (A)	**7.19**	18.08	464.40	398.38	17.64	17.26	24.71	18.19
SEAFD (G)	**6.83**	17.72	461.17	398.88	17.57	17.33	32.85	18.57

Table 6
RAM-Hours (GB-Hour) results obtained by using DISCUSS with a window size $w = 300$ against original base learners. Results in bold stand for the best results obtained for each experiment, while underlining indicates the best result per classifier, either with or without DISCUSS.

Experiment	NB	DISCUSS NB	kNN	DISCUSS kNN	VFDT	DISCUSS VFDT	HAT	DISCUSS HAT
BG1 (A)	$\mathbf{1.14 \times 10^{-8}}$	1.25×10^{-6}	8.26×10^{-5}	3.26×10^{-5}	3.77×10^{-6}	1.34×10^{-6}	3.20×10^{-6}	5.91×10^{-6}
BG1 (G)	$\mathbf{1.07 \times 10^{-8}}$	1.22×10^{-6}	8.26×10^{-5}	3.27×10^{-5}	3.62×10^{-6}	1.50×10^{-6}	3.21×10^{-6}	4.89×10^{-6}
BG2 (A)	$\mathbf{9.48 \times 10^{-9}}$	1.21×10^{-6}	8.14×10^{-5}	3.20×10^{-5}	4.80×10^{-6}	1.28×10^{-6}	2.95×10^{-6}	7.23×10^{-6}
BG2 (G)	$\mathbf{9.07 \times 10^{-9}}$	1.23×10^{-6}	8.23×10^{-5}	3.25×10^{-5}	4.71×10^{-6}	1.26×10^{-6}	2.37×10^{-6}	7.16×10^{-6}
BG3 (A)	$\mathbf{1.11 \times 10^{-8}}$	1.24×10^{-6}	8.29×10^{-5}	3.28×10^{-5}	5.75×10^{-6}	1.24×10^{-6}	3.15×10^{-6}	4.62×10^{-6}
BG3 (G)	$\mathbf{1.03 \times 10^{-8}}$	1.20×10^{-6}	8.30×10^{-5}	3.29×10^{-5}	5.58×10^{-6}	1.21×10^{-6}	4.09×10^{-6}	5.59×10^{-6}
RTGFD (A)	$\mathbf{9.94 \times 10^{-8}}$	4.58×10^{-6}	1.18×10^{-4}	6.19×10^{-5}	1.25×10^{-4}	4.68×10^{-6}	3.78×10^{-5}	3.15×10^{-5}
RTGFD (G)	$\mathbf{1.00 \times 10^{-7}}$	4.60×10^{-6}	1.18×10^{-4}	6.17×10^{-5}	1.25×10^{-4}	4.66×10^{-6}	5.62×10^{-5}	3.15×10^{-5}
SEAFD (A)	$\mathbf{1.96 \times 10^{-8}}$	1.23×10^{-6}	7.50×10^{-5}	3.16×10^{-5}	1.28×10^{-5}	2.18×10^{-6}	8.52×10^{-6}	6.15×10^{-6}
SEAFD (G)	$\mathbf{1.86 \times 10^{-8}}$	1.19×10^{-6}	7.45×10^{-5}	3.17×10^{-5}	1.26×10^{-5}	2.09×10^{-6}	2.73×10^{-5}	6.29×10^{-6}

Table 7
Accuracy results (%) obtained by using DISCUSS with a window size $w = 300$ against original base learners in real-world datasets. Results in bold stand for the best results obtained for each experiment, while underlining indicates the best result per classifier, either with or without DISCUSS.

Experiment	NB	DISCUSS NB	kNN	DISCUSS kNN	VFDT	DISCUSS VFDT	HAT	DISCUSS HAT
AIR	**68.57**	67.23	67.42	64.50	67.14	64.34	64.97	64.32
COVTYPE	74.75	76.26	**90.33**	85.55	82.53	73.35	84.51	71.12
ELEC	73.37	73.71	77.97	78.74	79.21	79.19	**83.23**	80.89
KDD99	99.92	96.31	99.98	99.88	**99.99**	77.89	99.42	99.41
SPAM	75.71	90.97	84.08	90.41	87.38	86.97	85.21	**92.95**

ios, mainly if the base learner is memory-efficient, such as for the NB classifier, while smaller increases may cause kNN-based systems to fail due to lack of memory.

6.5. Real-world data

In this section, we show how DISCUSS behaves when applied to different classifiers and different real-world datasets. The average accuracy results obtained are presented in Table 7. Here, the results in real-world datasets follow the trends observed in synthetic datasets, where DISCUSS can improve the classification rates of NB and kNN classifiers in some scenarios, while decision tree-based learners do not benefit from it. One notable exception is the high-dimensional SPAM experiment, where the Hoeffding Adaptive Tree (HAT) has its accuracy improved from approximately 85% to 92%. It is also worth to mention the improvements observed in Naive Bayes (NB) and k-Nearest Neighbors (kNN) classifiers still in the SPAM experiment, where classifiers rate improved from approximately 76% to 91% for NB and from 84% to 90% for kNN. These results show that when confronted with high-dimensional data streams, performing feature selection with DISCUSS could be beneficial, what does not occur with the other datasets consistently as they have already been pre-processed and feature selection might have been performed already.

Tables 8 and 9 report the processing time and memory consumption rates of classifiers before and after their association with DISCUSS. In these tables, we can observe the same we observed earlier on synthetic experiments, as only kNN benefits concerning processing time, and all learners suffer from higher memory consumption rates when DISCUSS is used.

6.6. When and why DISCUSS fails

In the previous sections, DISCUSS was evaluated using different classifiers and in a variety of experiments. Focusing on accuracy, we observed that the scores obtained for BG3 and RTG experiments were reasonably lower when compared to the remaining experiments. Now, we use these two generators as examples to highlight situations where DISCUSS does not improve the overall results. To facilitate this analysis, we will work under the assumption that our stream is class-imbalanced and that the values each instance can present (as depicted in Figs. 10 and 11) are equally probable.

BG3

In this concept, earlier introduced in Eq. (11), there are three dimensions relevant to the class: D_α, D_β and D_ϵ. Assuming that we have a stream that also encompasses an irrelevant feature D_ξ when generating data, we would have the following 16 features combinations and the probabilities depicted in Fig. 10.

We start our analysis by comparing a relevant and an irrelevant feature: D_α and D_ξ. For D_α, if we compute the joint probability

Table 8
Processing time (s) obtained by using DISCUSS with a window size $w = 300$ against original base learners in real-world datasets. Results in bold stand for the best results obtained for each experiment, while underlining indicates the best result per classifier, either with or without DISCUSS.

Experiment	NB	DISCUSS NB	kNN	DISCUSS kNN	VFDT	DISCUSS VFDT	HAT	DISCUSS HAT
AIR	**1.25**	1.32	17.39	12.89	5.26	5.51	20.40	21.42
COVTYPE	**4.28**	4.80	4.97	3.66	72.32	75.99	6.92	6.99
ELEC	**0.43**	0.52	7.85	5.67	0.70	0.75	1.04	1.07
KDD99	**2.63**	3.03	95.57	73.23	3.57	3.87	4.20	4.37
SPAM	351.79	378.24	19028.03	13557.47	677.68	683.10	**308.17**	315.41

Table 9
RAM-Hours (GB-Hour) obtained by using DISCUSS with a window size $w = 300$ against original base learners in real-world datasets. Results in bold stand for the best results obtained for each experiment, while underlining indicates the best result per classifier, either with or without DISCUSS.

Experiment	NB	DISCUSS NB	kNN	DISCUSS kNN	VFDT	DISCUSS VFDT	HAT	DISCUSS HAT
AIR	$\mathbf{2.49 \times 10^{-8}}$	3.09×10^{-8}	1.52×10^{-6}	1.68×10^{-6}	4.32×10^{-5}	5.22×10^{-5}	2.75×10^{-5}	3.27×10^{-5}
COVTYPE	$\mathbf{5.75 \times 10^{-8}}$	7.56×10^{-8}	2.76×10^{-5}	3.27×10^{-5}	1.75×10^{-7}	2.09×10^{-7}	1.24×10^{-7}	1.53×10^{-7}
ELEC	$\mathbf{6.34 \times 10^{-10}}$	8.05×10^{-10}	6.28×10^{-7}	7.16×10^{-7}	1.60×10^{-8}	1.93×10^{-8}	8.33×10^{-9}	1.01×10^{-8}
KDD99	$\mathbf{2.10 \times 10^{-8}}$	2.58×10^{-8}	2.65×10^{-5}	3.08×10^{-5}	1.91×10^{-7}	2.22×10^{-7}	1.75×10^{-7}	2.07×10^{-7}
SPAM	$\mathbf{2.61 \times 10^{-3}}$	3.97×10^{-3}	6.24	8.36	9.20×10^{-2}	1.10×10^{-1}	8.75×10^{-2}	1.18×10^{-1}

D_α	D_β	D_ϵ	D_ξ	Y
1	1	1	1	1
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	1
1	1	0	0	0
1	0	1	0	0
1	0	0	1	0
0	1	1	0	0
0	1	0	1	0
0	0	1	1	0
1	1	1	0	1
1	1	0	1	0
1	0	1	1	0
0	1	1	1	0
0	0	0	0	1

since all features combinations are equally likely to occur, it follows that:

$$\left\{ \begin{array}{l} \textbf{Features Probabilities} \\ P[D_\alpha = 1] = {}^1\!/_2 \\ P[D_\beta = 1] = {}^1\!/_2 \\ P[D_\epsilon = 1] = {}^1\!/_2 \\ P[D_\xi = 1] = {}^1\!/_2 \\ \textbf{Class Probabilities} \\ P[Y = 1] = {}^4\!/_{16} = {}^1\!/_4 \\ P[Y = 0] = {}^{12}\!/_{16} = {}^3\!/_4 \end{array} \right.$$

Fig. 10. BG3 feature combinations and probabilities.

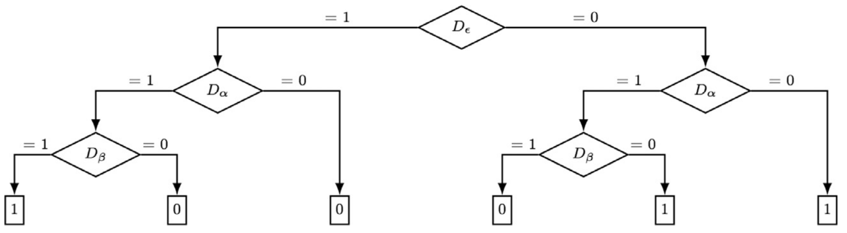


Fig. 11. Example of RTG concept.

$P[D_\alpha = 1, Y = 1]$, it follows that we have only 2 cases out of 16 ($\frac{1}{8}$) that match these criteria, which occur when $D_\alpha = D_\beta = D_\epsilon = 1$ regardless of the value of D_ξ, i.e. $D_\xi = 1$ or $D_\xi = 0$. In this case, we could verify if D_α and Y are independent from each other, by checking if $P[D_\alpha = 1, Y = 1] = P[D_\alpha = 1] \times P[Y = 1]$. The last assertion is true, since $P[D_\alpha = 1] \times P[Y = 1] = \frac{1}{2} \times \frac{1}{4} = \frac{1}{8}$, which is the same probability presented above.

We could then repeat the same process for D_ξ. First, the joint probability $P[D_\xi = 1, Y = 1]$ would end up with the same value $\frac{1}{8}$. Next, by comparing if D_ξ and Y are independent, it follows that $P[D_\xi = 1, Y = 1] = P[D_\xi = 1] \times P[Y = 1]$ holds, because $P[D_\xi = 1] \times P[Y = 1] = \frac{1}{2} \times \frac{1}{4} = \frac{1}{8}$. Now, we can infer the following: (i) both D_α and D_ξ are equally biased towards the class, and thus, probability-based measures would score them equally, and that (ii) probability-wise, both features are claimed to be independent w.r.t. the class, which should not have been held true for D_α, since the class determination depends on it.

The rationale here is simple: by evaluating each feature solely, any "single feature metric" such as symmetrical uncertainty, would fail at depicting the discriminative power of features in cases

D_α	D_β	D_ϵ	D_ξ	Y
1	1	1	1	1
1	0	0	0	1
0	1	0	0	1
0	0	1	0	0
0	0	0	1	1
1	1	0	0	0
1	0	1	0	0
1	0	0	1	1
0	1	1	0	0
0	1	0	1	1
0	0	1	1	0
1	1	1	0	1
1	1	0	1	0
1	0	1	1	0
0	1	1	1	0
0	0	0	0	1

since all features combinations are equally likely to occur, it follows that:

Features Probabilities

$P[D_\alpha = 1] = 1/2$

$P[D_\beta = 1] = 1/2$

$P[D_\epsilon = 1] = 1/2$

$P[D_\xi = 1] = 1/2$

Class Probabilities

$P[Y = 1] = 8/16 = 1/2$

$P[Y = 0] = 8/16 = 1/2$

Fig. 12. RTG feature combinations and probabilities.

where the class seems independent from them. In such cases, more complex metrics that evaluate subsets of features, such as a joint symmetrical uncertainty $SU([D_\alpha, D_\beta, D_\epsilon], Y)$ would be sufficient at depicting these complex relationships between features, however, it's computation is not trivial nor efficient to be performed on data streams.

RTG

Similarly to the BG3 discussion, let us now work under the assumption that our RTG concept relies on D_α, D_β and D_ϵ to determine the class Y and that we also have an irrelevant feature D_ξ. Given that all of these features are binary, a possible RTG concept would be the one depicted in Fig. 11, which would result in features-class combinations and probabilities presented in Fig. 12. Using the same procedure as before, let us now compare two relevant features (D_α and D_ϵ) and an irrelevant one (D_ξ). Computing the joint probabilities between each feature and the class, we obtain: $P[D_\alpha = 1, Y = 1] = \frac{1}{4}$, $P[D_\epsilon = 1, Y = 1] = \frac{1}{8}$ and $P[D_\xi = 1, Y = 1] = \frac{1}{4}$.

Next, we could verify if these features are independent w.r.t. the class as follows: (i) D_α is independent since $P[D_\alpha = 1, Y = 1] = P[\alpha = 1] \times P[Y = 1]$ holds, (ii) D_ϵ is not dependent since $P[D_\epsilon = 1, Y = 1] \neq P[\epsilon = 1] \times P[Y = 1]$; and (iii) D_ξ is independent since $P[D_\xi = 1, Y = 1] = P[\xi = 1] \times P[Y = 1]$ also holds. As a result, DISCUSS would compute the following symmetrical uncertainty values: $SU(D_\alpha, Y) = 0.0000$, $SU(D_\epsilon, Y) = 0.1887$ and $SU(D_\xi, Y) = 0.0000$. These results show that DISCUSS is unable to depict correlations between relevant features and the class since it performs a "flat" evaluation of the features, while on an RTG concept it would be necessary to evaluate different partitions of data, such as the ones that are traversed by the first node of the tree. As before, these complex relationships amongst subsets of variables would require the computation of joint symmetrical uncertainties.

7. Conclusion

Motivated by drifting and redundant features, we proposed in this paper a novel dynamic feature selection algorithm for data streams, namely DISCUSS. The key concept behind DISCUSS is the symmetrical uncertainty scoring operator, which can identify irrelevant and redundant features with reasonable success. Even though a single merit-guided sequential feature selection strategy was proposed in this paper, DISCUSS can be easily configured and assessed in the future with novel selection schemes. DISCUSS was evaluated in both synthetic and real-world scenarios in conjunction with different classifiers, showing its ability in being classifier independent.

DISCUSS improves the performance of methods that do not have any internal mechanism for feature selection, and for kNN, the combination with DISCUSS can even result in lower total resource consumption.

Conversely, combining DISCUSS with methods that use intrinsic feature selection, like decision trees, may result in lower accuracy rates, especially in the presence of complex feature interactions. Identifying and exploring feature interactions in streaming scenarios is an open gap that shall be targeted in future works.

Also in future works, we plan to adapt feature selection-specific evaluation metrics from batch machine learning to streaming scenarios. For instance, Selection Accuracy (Galelli et al., 2014) is a widely used metric that compares the selected features with a ground-truth, thus allowing the feature selection accuracy to be quantified without the need of testing different classifiers. Another important metric is Stability (Kuncheva, 2007), which quantifies how 'volatile' a feature selector is given perturbations in the input data. Finally, we also highlight an important limitation that regards the streams adopted during the analysis since all of them are binary classification problems. Therefore, it becomes of the utmost importance to assess other datasets and streams with more than 2 class possible outcomes in future works, which is a more general limitation of data stream works, as cited in the seminal work of Gama, Zliobaite, Bifet, Pechenizkiy, and Bouchachia (2014).

Acknowledgements

This research was financially supported by the CoordenaÃo de AperfeiÃoamento de Pessoal de Nḍvel Superior through the Programa de Suporte à Pòs-Graduação de Instituições de Ensino Particulares (PROSUP) program (grant no. 88881.131817/2016-01).

References

Abdulsalam, H., Skillicorn, D. B., & Martin, P. (2011). Classification using streaming random forests. *IEEE Transactions on Knowledge and Data Engineering, 23*(1), 22–36. doi:10.1109/TKDE.2010.36.

Almeida, E., Ferreira, C. A., & Gama, J. (2013). Adaptive model rules from data streams. In *ECML/PKDD (1)*. In *Lecture Notes in Computer Science: Vol. 8188* (pp. 480–492). Springer.

Barddal, J. P., Gomes, H. M., & Enembreck, F. (2015). A survey on feature drift adaptation. In *Proceedings of the international conference on tools with artificial intelligence*. IEEE.

Barddal, J. P., Gomes, H. M., Enembreck, F., & Pfahringer, B. (2017). A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. *Journal of Systems and Software, 127*, 278–294. doi:10.1016/j.jss.2016.07.005.

Barddal, J. P., Gomes, H. M., Enembreck, F., Pfahringer, B., & Bifet, A. (2016a). On dynamic feature weighting for feature drifting data streams. *ECML/PKDD'16. Lecture Notes in Computer Science*. Springer.

Barddal, J. P., Gomes, H. M., de Souza Britto Jr. , A., & Enembreck, F. (2016b). A benchmark of classifiers on feature drifting data streams. *ICPR'16. Lecture Notes in Computer Science*. Springer.

Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. *SIAM international conference on data mining*.

Bifet, A., & Gavaldà, R. (2009). *Adaptive learning from evolving data streams* (pp. 249–260)). Berlin, Heidelberg: Springer Berlin Heidelberg.

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). MOA: Massive online analysis. *The Journal of Machine Learning Research, 11*, 1601–1604.

Boulle, M. (2005). Optimal bin number for equal frequency discretizations in supervized learning. *Intelligent Data Analysis, 9*(2), 175–188.

Candillier, L., & Lemaire, V. (2012). Design and analysis of the nomao challenge active learning in the real-world. In *Proceedings of the ALRA: Active learning in real-world applications, workshop ECML-PKDD*.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

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*. In *KDD '00* (pp. 71–80). New York, NY, USA: ACM. doi:10.1145/347090.347107.

Duarte, J., & Gama, J. (2017). Feature ranking in hoeffding algorithms for regression. In *Proceedings of the symposium on applied computing, SAC 2017, Marrakech, Morocco, april 3–7, 2017* (pp. 836–841). doi:10.1145/3019612.3019670.

Galelli, S., Humphrey, G. B., Maier, H. R., Castelletti, A., Dandy, G. C., & Gibbs, M. S. (2014). An evaluation framework for input variable selection algorithms for environmental data-driven models. *Environmental Modelling & Software, 62*, 33–51. doi:10.1016/j.envsoft.2014.08.015.

Gama, J., & Pinto, C. (2006). Discretization from data streams: Applications to histograms and data mining. In *Proceedings of the 2006 ACM symposium on applied computing*. In *SAC '06* (pp. 662–667). New York, NY, USA: ACM. doi:10.1145/1141277.1141429.

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* (pp. 329–338). ACM SIGKDD.

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

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., et al. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning, 106*(9), 1469–1495. doi:10.1007/s10994-017-5642-8.

Guyon, I. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research, 3*, 1157–1182.

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*. In *ICML '00* (pp. 359–366). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc..

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2006). Dynamic feature space and incremental feature selection for the classification of textual data streams. In *ECML/PKDD-2006 international workshop on knowledge discovery from data streams. 2006* (p. 107). Springer Verlag.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence, 97*(12), 273–324. Relevance. doi: 10.1016/S0004-3702(97)00043-X.

Kuncheva, L. I. (2007). A stability index for feature selection. In *Proceedings of the 25th conference on proceedings of the 25th IASTED international multi-conference: Artificial intelligence and applications*. In *AIAP'07* (pp. 390–395). Anaheim, CA, USA: ACTA Press.

Kushmerick, N. (1999). Learning to remove internet advertisements. In *Proceedings of the third annual conference on autonomous agents* (pp. 175–181). ACM.

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*. In *CSNT '14* (pp. 468–471). Washington, DC, USA: IEEE Computer Society. doi:10.1109/CSNT.2014.99.

Nguyen, H.-L., Woon, Y.-K., & Ng, W.-K. (2014). A survey on data stream clustering and classification. *Knowledge and Information Systems*, 1–35. doi:10.1007/s10115-014-0808-1.

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

Oza, N. (2005). Online bagging and boosting. In *Systems, man and cybernetics, 2005 IEEE international conference on: Vol. 3* (pp. 2340–2345Vol. 3). doi:10.1109/ICSMC.2005.1571498.

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

Pearson, K. (1920). Notes on the history of correlation. *Biometrika, 13*(1), 25–45.

Pfahringer, B., Holmes, G., & Kirkby, R. (2008). Handling numeric attributes in hoeffding trees. In *Proceedings of the 12th pacific-asia conference on advances in knowledge discovery and data mining*. In *PAKDD'08* (pp. 296–307). Berlin, Heidelberg: Springer-Verlag.

Rudnicki, W. R., Wrzesień, M., & Paja, W. (2015). All relevant feature selection methods and applications. In *Feature selection for data and pattern recognition*. In *Studies in Computational Intelligence: Vol. 584* (pp. 11–28). Springer Berlin Heidelberg. doi:10.1007/978-3-662-45620-0_2.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal, 27*(3), 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x.

Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., de Carvalho, A., & Gama, J. (2013). Data stream clustering: A survey. *ACM Computing Surveys, 46*(1), 13:1–13:31. doi:10.1145/2522968.2522981.

Sovdat, B. (2014). Updating formulas and algorithms for computing entropy and gini index on time-changing data streams. *CoRR, abs/1403.6348*. arXiv: 1403.6348

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* (pp. 377–382). ACM SIGKDD.

Webb, G. I., Hyde, R., Cao, H., Nguyen, H. L., & Petitjean, F. (2016). Characterizing concept drift. *Data Mining and Knowledge Discovery, 30*(4), 964–994. doi:10.1007/s10618-015-0448-4.

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning, 23*(1), 69–101. doi:10.1023/A:1018046501280.

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

Witten, I. H., & Frank, E. (2005). Data mining: Practical machine learning tools and techniques. *(Morgan Kaufmann series in data management systems)* (2nd ed.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc..

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* (pp. 856–863). AAAI Press.

Zamani-Dehkordi, P., Rakai, L., & Zareipour, H. (2017). Estimating the price impact of proposed wind farms in competitive electricity markets. *IEEE Transactions on Sustainable Energy, 8*(1), 291–303. doi:10.1109/TSTE.2016.2598265.

Zhao, Z., Morstatter, F., Sharma, S., Alelyani, S., Anand, A., & Liu, H. (2010). Advancing feature selection research. *ASU Feature Selection Repository*, 1–28.