2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Features and Classes Drift Detector to Deal with Imbalanced Data Streams

Silas Garrido Teixeira de Carvalho Santos*†, Danilo Rafael de Lima Cabral*, Roberto Souto Maior de Barros*
*Centro de Informática, Universidade Federal de Pernambuco, 50740-560, Recife-PE, Brazil
{sgtcs,drlc,roberto}@cin.ufpe.br
†SiDi Institute, 51110-160, Recife-PE, Brazil
{s.garrido}@sidi.org.br

***Abstract*—Data streams, due to their dynamic nature, tend to impose a number of constraints on the functioning of the learning models used to extract knowledge from these environments. In this context, concept drift is an emerging research area, as they negatively affect the performance of classifiers: after they have been trained with a specific concept, they tend to lose accuracy in the presence of a new concept. Additionally, this problem is often worsened in environments with imbalanced classes, because the identification of changes in the distributions of examples belonging to minority classes is usually more complex, due to their lack of representativeness in the data stream. This work proposes the Features and Classes Drift Detector (FCDD), a new method specially designed to deal with the problem of concept drifts in imbalanced data streams, aiming to maintain the accuracy of detections in minority classes and, in addition, to avoid discarding the knowledge inherent to the classes unaffected by drifts. Experiments conducted in an imbalanced scenario with partial concept drift demonstrated the effectiveness of the proposed method when compared to the current state of the art detectors.**

***Index Terms*—Data Streams, Concept Drifts, Imbalanced Classes, Online Learning**

I. Introduction

Traditionally, most research efforts in data mining have followed a historical bias of focusing on extracting knowledge from static data, previously known and available in repositories [1]. In contrast, given today's new technologies, the way people store, process, and transmit data has gradually changed. This evolutionary process gave rise to applications that generate data that flows continuously, at high speed, and in the form of potentially unlimited streams [2]. Examples of continuous data streams tasks include filtering spam in e-mail messages, monitoring data from sensors, intrusion detection, social networks, sentiment analysis, etc.

With the recent growth in the number of applications that process continuous data streams, it is common to observe data domains with imbalanced classes [3], such as risk management and anomaly detection. The imbalance of classes happens when a data stream arranged as a sequence of pairs containing a vector of input attributes and an associated class, does not have classes distributed equitably: at least one class is a minority when compared to the other(s) [4].

In addition to *imbalance*, another common problem in these scenarios is *concept drift*, which can be observed when a given data stream has its distribution changed over time [2]. Accordingly, such concept drifts tend to negatively affect the performance of the learning algorithms, since a model already trained with a given concept will make more mistakes, losing its accuracy, when it is presented to a new concept.

From the perspective of the classification task, class imbalance tends to worsen the concept drift problem in data streams, besides affecting the performance of the learners in the minority class [5]. This worsening occurs because most of the proposed solutions to deal with changes in the data distributions handle concept drift more generally, regardless of the class representation in the stream. Thus, identifying changes in a minority class is a difficult task.

Traditionally, concept drift detectors [6] use two alarm levels: warning and drift. A new base classifier is created and maintained in parallel with the old learning model when the warning level is signaled. If the drift level is reached, the detector deletes the old classifier and keeps only the new one. On the other hand, the new classifier is excluded if the warning signal becomes a false alarm [7].

Since such detectors, in most cases, are based only on the performance of the classifiers, they restart the base classifier in order to disregard the instances of the old concept and force the model to learn only from examples belonging to the new concept. However, in some situations, this generic approach tends to be a limited solution [8]. The main problem is that the change in distribution does not always affect all classes, and discarding all instances learned by the classifier up to that point can lead to performance loss.

Taking into account the problems presented, this article proposes the Features and Classes Drift Detector (FCDD), a detector designed to signal real and virtual concept drift associated with the class. This feature identifies changes in minority classes of imbalanced datasets and ensures that the classifier does not discard instances associated with classes not affected by drifts.

To the best of our knowledge, this is the first method able to identify drifts in imbalanced scenarios at detector level which is not affected by the number of classes of the problem in hand, i.e. binary or multiclass.

The rest of this paper is organized as follows: Section II briefly reviews the online classification and concept drift detection problems, including the issue of data imbalance in streams; Section III presents the new approach (FCDD);

Section IV details the experiments and the environment chosen for the tests; Section V discusses the results of the experiments; and, finally, Section VI draws conclusions and suggests proposals for future work.

II. Background

Given n instances in the (x, y) form, such that $x \in \mathcal{X}$ is a d-dimensional vector of attributes and $y \in \mathcal{Y}$ is the target, a hypothesis $h \in \mathcal{H}$ must be able to map an unknown instance in the form $h : \mathcal{X} \rightarrow \mathcal{Y}$ with adequate accuracy. The data stream or online classification is a variant of the traditional batch classification, differing in how the data is presented to the learner. If the dataset is static and entirely accessible, it is a batch-based configuration. On the other hand, in streaming environments, instances are not readily available to the classifier for training; instead, they are presented sequentially over time, and the learner must adapt its model according to the arrival of instances from the stream [9]. Therefore, we denote $\mathcal{S} = [(x_t, y_t)]_{t=1}^{n\rightarrow\infty}$ as a data stream providing instances (x_t, y_t), each of which arriving at a timestamp t.

Considering that the probability that an instance x belongs to a class y is $P(y \mid x) = P(y) \cdot P(x \mid y) \ / \ P(x)$ [10], a concept drift is assumed to occur when the joint probabilities $P(x_t, y_t)$ and $P(x_{t+1}, y_{t+1})$ have different values, that is, $P(x_t, y_t) \neq P(x_{t+1}, y_{t+1})$ [11]. Thus, considering Bayes' theorem, concept drifts can arise essentially in three ways 1) change in $P(y)$ – priori probability of y; 2) change in $P(x \mid y)$ – posteriori probability from x conditional to y; and 3) change in $P(y \mid x)$ – posteriori probability from y conditional to x.

Regarding these three possibilities, variations in $P(y)$ mean changes in balance levels between classes without necessarily changing their decision limits [8]. Changes in $P(x \mid y)$ result from an incomplete or outdated representation of the actual distribution of the examined data, suggesting the learning models need to be updated, even though the true boundaries of decision between classes may remain unchanged [12]. Finally, changes in $P(y \mid x)$ reveal drifts in the decision limits between the classes of the analyzed data, indicating that the learning models previously built on such data may have become ineffective and need to adapt to the new concept [3].

Since changes in $P(y)$ and $P(x \mid y)$ do not necessarily imply modifications in decision limits between classes, they are named virtual drifts. Oppositely, changes in $P(y \mid x)$ directly impact the decision boundaries between classes and, thus, they are categorized as real drifts [13].

Concerning the imbalance problem, several approaches are found in the literature, and they can be categorized into two groups: data-level and algorithm-level approaches [3]. In the former, the imbalance is treated at the dataset level, using, for example, techniques such as oversampling, undersampling, or a combination of both. Recent work in this line includes [14], [15]. In the latter, the algorithm-level, decisions are made in the scope of the classifier in order to better deal with the problem, as can be seen in [16], [17]. More details about the data stream imbalance issue and recent directions can be found at [18].

III. Proposed Method

This work proposes FCDD, a method capable of identifying concept drift in imbalanced environments by performing detections associated with the classes. Additionally, FCDD maintains samples of the unaffected classes to aid in post-drift classifier recovery. Algorithm 1 details how the method works. It receives as input the data stream ($\mathcal{S}$), the main classifier (h^1), the size of the windows (W), and the significance levels related to the warning state (α^w) and concept drift detection (α^d).

As new instances arrive (line 6), FCDD stores the last $2\times W$ instances associated with each label in two sliding windows: the first (ω^1) containing the most recent W instances; and the second (ω^2) containing the oldest W. Considering the iterators i, j and w, and knowing that ω^1 and ω^2 are three-dimensional matrices, we have that $i \in \{1, 2, ..., I\}$, $j \in \{1, 2, ..., J\}$ and $w \in \{1, 2, ..., W\}$, where I, J and W are the total labels of the dataset, the total attributes and the window size, respectively.

The lines 9 and 15 are responsible for adding the elements in ω^1 and ω^2. Initially, each instance associated with a y class will be inserted into ω^1 until the W limit is exceeded. The first time this occurs, the oldest instance of ω^1 will be moved to ω^2, making room for the current instance (W+1) to be inserted into the last ω^1 position. The same procedure will be performed for all instances, and when ω^2 reaches size W, its oldest instance will be discarded. Thus, ω^1 and ω^2 will store, for each y, only the last $2\times W$ instances of the stream $\mathcal{S}$.

As will be seen in detail later, the concept drift verification of FCDD will be performed individually on each of the attributes and on the hits and misses of different instances. For this reason, it is necessary to add the dimension of the attributes, justifying the iteration performed in the line 8. It is important to note that the initialization of the J dimension is increased by 1 (see line 3) because, in addition to the values of the attributes, the correct and incorrect predictions associated with y are also stored (line 15).

Given a dimension associated with the y label, only when ω^1_{yj} and ω^2_{yj} are filled will the drift and warning checks start to occur (line 18). The requirement for $4\times W$ instances ensures the window does not contain data from the start of the stream, which is usually unstable, making false positives less likely. Note that this verification is performed by class. Thus, a drift may be detected in a given class even if others have not reached the minimum number of instances.

Once the condition of line 18 is satisfied, the next step of FCDD iterates on the dimension of the attributes. The idea is to compare ω^1_{yj}, which stores the values of the j attribute of the last W instances associated with the y class, with ω^2_{yj}, which is similar, but with information from the last instances belonging to the range [W+1, $2\times W$]. In addition, the last iteration of the loop (J+1) compares hits (1) and misses (0) associated with y contained in ω^1 and ω^2. Tests on the attributes correspond to attempts to find drifts in $P(x \mid y)$ (virtual). On the other hand, checks on hits and misses aim to detect drifts in $P(y \mid x)$ (real).

Algorithm 1: Features and Classes Drift Detector (FCDD)

```
   Input: $\mathcal{S}$, $h^1$, $W$, $\alpha^d$, $\alpha^w$
1  $n_i$ = 1 $\forall i \in I$
2  $a_i$ = 0 $\forall i \in I$
3  $\omega^1_{ijw}$ = $\omega^2_{ijw}$ = 0 $\forall ijw \in I,\ J+1,\ W$
4  $s$ = $\varnothing$  ▷ Saved instances
5  $h^2$ = $h^1$
6  foreach $x$ in $\mathcal{S}$ do
7      $y$ = label($x$)  ▷ Get the correct label
8      for $j = 1$ to $J$ do
9          addCircularArray($\omega^1_{yj}$, $\omega^2_{yj}$, $x_j$)
10     $\hat{y}$ = $h^1(x)$  ▷ Predict instance
11     error = 0
12     if $\hat{y} \neq y$ then
13         error = 1
14     $j$ = $J$ + 1
15     addCircularArray($\omega^1_{yj}$, $\omega^2_{yj}$, error)
16     $n_y$ = $n_y$ + 1
17     drift = warning = 0
18     if $n_y > 4 \times W$ then
19         for $j = 1$ to $J+1$ do
20             p-value = hypothesisTest($\omega^1_{yj}$, $\omega^2_{yj}$)
21             if $p\text{-}value < \alpha^d$ then
22                 drift = $a_y$ = 1
23             else if $p\text{-}value < \alpha^w$ then
24                 warning = $a_y$ = 1
25     if drift then
26         if $s$ == $\varnothing$ then
27             $s$ = copy($\omega^1$, $\omega^2$)
28         $h^1$ = $h^2$
29         partiallyTrain($h^1$, $s$, $a$)
30         reset()
31     else if warning then
32         if $s$ == $\varnothing$ then
33             $s$ = copy($\omega^1$, $\omega^2$)
34         train($h^2$, $x$)
35     else
36         $s$ = $\varnothing$
37     train($h^1$, $x$)
```

To identify concept drifts, a hypothesis test is performed on the difference in means with known variance [19] (line 20), assuming that the instances are independent and identically distributed (IID). The null hypothesis (H_0) considers that there is no difference in the means of the values of ω^1_{yj} and ω^2_{yj}, i.e., they belong to the same distribution. The alternative hypothesis (H_1) indicates a difference in these means, suggesting a concept drift. So, considering that $\overline{x}_1$, $\overline{x}_2$, σ_1^2 and σ_2^2 are the mean and variance of ω^1_{yj} and ω^2_{yj}, the test statistic can be calculated by:

$$z = (\overline{x}_1 - \overline{x}_2)/\sqrt{\sigma_1^2/W + \sigma_2^2/W}$$
$$\text{p-value} = 2[1 - \Phi(|z|)]$$

where Φ corresponds to the cumulative distribution function of a standard normal random variable. Lines 21 and 23 reject the null hypothesis using two levels of significance. The lower confidence in rejection means the detector starts the warning state, while the higher confidence indicates a drift.

After performing the hypothesis test to find changes in $P(x \mid y)$ or $P(y \mid x)$, if a warning has occurred (line 31), the first action to be performed is to store all instances of ω^1 and ω^2 in s (if s is empty), considering all dimensions. Then, the alternative classifier is trained with the current instance (line 34). The warning state is expected to last for some time until the drift is confirmed, and the rationale of training h^2 is to avoid losing instances during this period.

When a concept drift occurs (line 25), assuming a warning was detected earlier, there will be two sets of instances: (1) those stored in s, backward from the point where the warning started; and (2) those trained in h^2, throughout the warning period. Thus, the main classifier assumes the instances trained by the alternative classifier (line 28) and later trains using the instances contained in s associated with classes that identified neither drift nor warning. This control is performed using the a array, which will have 1 in the y dimension if at least one rejection is identified in $P(x \mid y)$ or $P(y \mid x)$ – lines 22 and 24.

Sometimes, a drift is identified without previous warning. Thus, instead of containing the $2 \times W$ instances from the backward warning point, s will have instances from the backward drift point (line 27). Even so, FCDD can train with samples from unaffected classes. Finally, sometimes warnings are not followed by a drift: in this case, line 36 is useful to ensure that s does not store very old instances.

In terms of time complexity, for most cases FCDD will have asymptotic performance of $\Theta(J \times W)$, considering that the variance of the windows will be updated at each instance, useful for the hypothesis test calculation. However, its upper limit will be on the order of $\mathcal{O}(I \times J \times W)$. The worst case will happen after drift detection, where the two three-dimensional windows (ω^1 and ω^2) will be reset. This last information takes us to the memory complexity of FCDD, which is $\mathcal{O}(I \times J \times W)$.

IV. Experiment Setting

This section describes the set up of the experiments used to evaluate FCDD against other well-known concept drift detectors, namely Reactive Drift Detection Method (RDDM) [20], Drift Detection Method (DDM) [7], Drift Detection Methods based on Hoeffding's Bounds (HDDM) [21], and Wilcoxon Rank Sum Test Drift Detector (WSTD) [22]. They all used Hoeffding Anytime Tree (HATT) [23] as base classifier, a variation of Hoeffding Tree (HT) [24] with minor modifications. The motivation for the changes was to make

HATT more efficient in terms of both computational costs and accuracy.

To keep the comparison fair, only detectors with characteristics close to FCDD were included in the experiments. However, despite the existence of classifiers and ensembles focused on the imbalanced problem, no detector-level approach that deals with imbalance and multiclass datasets was found in the literature. Therefore, our selection criteria considered the most recent, efficient and/or popular detectors.

We chose the FCDD parameters default values following the methodology proposed by [25] that uses a Differential Evolution (DE) algorithm to find values that work reasonably well in many scenarios. The values obtained with this arrangement were: W = 331, α^d = 0.000824958, and α^w = 0.00170301. In the other methods, the used parameters were their default values proposed by their respective authors.

To simulate an imbalanced scenario with partial occurrences of concept drift, we implemented an artificial dataset based on the exclusive or (XOR) operation. Each instance ($x \in \mathcal{X}$) will be composed of two numeric attributes (a^1 and a^2) in the range [0, 32[. Knowing that $a^1 \oplus a^2$ will also have an output in the range [0, 32[, we defined six possible labels ($y \in \mathcal{Y}$), as can be seen in Table I.

TABLE I
XOR DATASET LABELS WITH THEIR RESPECTIVE RANGES.

label$_1$	label$_2$	label$_3$	label$_4$	label$_5$	label$_6$
[0, 5]	[6, 9]	[10, 15]	[16, 19]	[20, 25]	[26, 31]

Note that classes 2 and 4 have smaller ranges than the others, precisely to cause the imbalance effect. Besides, we explore the fact that the 32 possible numbers resulting from the XOR operation are equally likely to occur, as long as a^1 and a^2 come from a random source.

With these dataset characteristics, the problem consists of, given two attributes, the classifier must predict the correct label, with six possible results. If the prediction matches the label of the answer of $a^1 \oplus a^2$, the classification is correct, otherwise, it is incorrect.

A stream S was created containing 100K instances (x), where a^1 and a^2 were randomly generated. Several generator seeds were used in order to create multiple versions of the dataset and enable the calculation of intervals with 95% confidence. To produce concept drifts in the 30K and 60K instances, the responses of classes 1, 2, 3 and 4 were exchanged whereas classes 5 and 6 remained unchanged.

The tests were run on an Intel Core i7 10700 processor, 32GB of RAM, and an SSD, using Ubuntu Desktop 22.04 LTS 64 bits operating system.

All methods and codes related to the experiments were developed through the Massive Online Analysis (MOA) framework [26].

Considering the data imbalance and its multiclass characteristic, we evaluate the methods usig the metrics Kappa (κ) [27] and Prequential Multi-Class AUC (PMAUC) [28], both appropriate for this scenario.

TABLE II
SPECIFIC METRICS TO DETECTORS WITH 95% CONFIDENCE INTERVALS, USING HATT AS BASE CLASSIFIER.

	μDist$_1$	μDist$_2$	MCC
FCDD	246.80±26.80	236.60±52.44	**0.96±0.10**
RDDM	248.20±30.39	248.60±25.77	0.83±0.13
DDM	383.33±31.51	330.00±44.42	0.50±0.62
HDDM$_A$	57.60±86.37	34.00±31.10	0.81±0.19
HDDM$_W$	51.25±59.10	**19.33±14.01**	0.16±0.09
WSTD	**17.60±4.44**	23.00±7.17	0.33±0.13

TABLE III
GENERAL PERFORMANCE METRICS WITH 95% CONFIDENCE INTERVALS, USING HATT AS BASE CLASSIFIER.

	Kappa	PMAUC	Run-time	Memory
FCDD	**69.79±2.26**	**0.74±0.01**	25.01±2.14	2103±216
RDDM	68.76±0.53	**0.74±0.01**	15.05±1.62	75.02±7.10
DDM	68.97±2.78	**0.74±0.01**	14.93±1.87	42.51±6.34
HDDM$_A$	64.97±4.32	**0.74±0.01**	13.60±2.22	40.26±6.50
HDDM$_W$	28.56±2.26	0.71±0.01	**07.76±0.87**	**22.31±1.77**
WSTD	53.74±10.16	**0.74±0.02**	11.54±1.49	35.77±5.16

The first metric (κ) is defined by

$$\kappa = (p - p_{ran})/(1 - p_{ran})$$

where p represents the accuracy – measured by *Prequential* [29] – and p_{ran} represents the accuracy of a classifier that predicts labels randomly from the probability distribution of the predictions from a reference classifier. If the classification is perfectly correct, then $\kappa = 1$; otherwise, if the prediction matches that of p_{ran}, $\kappa = 0$.

The other metric intended to measure performance in this specific scenario (PMAUC) is defined by

$$\text{PMAUC} = \frac{2}{C \times (C-1)} \sum_{i<j} \hat{A}(i,j)$$

where C is the number of classes and $\hat{A}(i,j)$ is the probability that a randomly drawn member of class j will have a lower estimated probability of belonging to class i than a randomly drawn member of class i [28].

Assuming a dynamic and time-changing environment, the accuracy evaluation used the Prequential methodology with a sliding window of size 1000 as its forgetting mechanism [30]: each incoming instance is used initially for testing and subsequently for training.

In addition to κ and PMAUC, we compared the methods using run-time; memory usage in bytes per second (B/s); correctly (true positives, TP) and incorrectly (false positives, FP) detected drifts, considering a tolerance of 400 instances; mean detection distances to the exact drift points (μDist$_1$ and μDist$_2$), evaluating separately the two drifts points in the dataset; and, finally, the Matthews Correlation Coefficient (MCC). The latter is calculated as

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP+FP}) \times (\text{TP+FN}) \times (\text{TN+FP}) \times (\text{TN+FN})}}.$$

V. Results

Taking into account the scenario defined above, this section discusses the performance of FCDD against the other selected concept drift detectors. Table II shows details of the metrics related to the drift detections while Table III presents the results of the other more general metrics.

Observing the precision in the drift detections (Table II), it is possible to verify that FCDD was successful in all of them, and with a low amount of false positives. RDDM was the second best method, however, with more false positives and slightly longer periods to identify concept drifts, which is reflected in the values of μDist_1 and μDist_2.

On the other hand, HDDM_A, HDDM_W and WSTD were quicker in identifying the drifts, but this greater sensitivity meant more false positives were identified, especially in the case of HDDM_W.

Regarding class imbalance, DDM was one of the most affected methods. Because part of the region affected by the drifts is imbalanced, when more instances of the unaffected region arrive, the classifier is going to make fewer errors, which means its detections will take longer – observe the μDist_1 and μDist_2 values for DDM. Because FCDD can detect concept drifts associated with classes, it tends not to face many problems in this context and, as such, it successfully identified the changes in the minority classes.

Regarding the MCC metric, we could say it summarizes the performance of the detectors relating to TP, TN, FP and FN. Analyzing these results, it is clear that the performance of HDDM_W was deteriorated by its excessive false positives. The same can be said of WSTD, but to a lesser extent. On the other hand, the good stability of FCDD in the different seeds was also reflected in its MCC, putting it at an advantage over the other approaches.

Considering that the drifts affect only four of the six classes of the XOR dataset, discarding all previous instances from the point of drift identification may not be the best strategy. All the detectors used in the experiments follow this strategy, except for FCDD. Its characteristic of detecting drifts by class made it possible to also identify unaffected regions. Thus, the classifier used in the new concept will be provided with instances not affected by the change, giving it a possible advantage.

Changing the focus to Table III, specifically for the κ and PMAUC metrics which takes into account the multiclass and unbalanced characteristic of the dataset, we observed a distinct performance from FCDD. Specifically in κ, FCDD delivered the highest result, closely followed by DDM and RDDM, even though RDDM had a much smaller confidence interval. In the PMAUC, the results of most detectors were equivalent, the exception being HDDM_W, which was worse than the other methods.

In terms of run-time and memory usage, FCDD was the most expensive method, as expected. Its higher consumption is directly associated with the more training the classifier will receive, because instances of old concepts are reused. As already discussed in Algorithm 1, $2 \times W$ instances of each class are stored in sliding windows for detection and possible reuse by the classifiers – unaffected regions data.

Considering the $\mathcal{O}(I \times J \times W)$ complexity of FCDD in time and memory, as explained in Section III, this higher consumption in these two metrics becomes even clearer. When compared to the other approaches, almost all of them with constant time complexity, it would hardly be possible, in practice, an equivalence in these metrics.

On the other hand, this higher overhead is justified by the more specialized behavior of FCDD, both in dealing with imbalanced data and in taking advantage of instances not affected by changes in the probability distribution. These characteristics are completely ignored in the other detectors.

Finally, although this overhead is not so significant as to make the method unfeasible in practice, it is reasonable to analyze the context in which FCDD might be applied, paying particular attention to high-dimensional datasets.

Complementing the analysis of the reported results, a hypothesis test, namely the Student's t-test [19], was performed in order to verify in which metrics FCDD was statistically superior to the other approaches. The null hypothesis (H_0) states that the compared methods are equivalent, while the alternative hypothesis (H_1) states that there is a statistical difference. Table IV shows, for each metric, the detectors statistically inferior to FCDD, i.e., H_0 was rejected. Note that the run-time and memory metrics were not included in table IV because FCDD was not better than any of the other detectors in these metrics.

Finally, strictly observing the confidence intervals: in accuracy, FCDD was statistically superior to HDDM_A, HDDM_W and WSTD, and statistically equivalent to DDM and RDDM; in FP, FCDD was better than all other detectors; in the case of TP, it was statistically superior to DDM, HDDM_W and WSTD; regarding run-time, there was little difference between the methods but WSTD was the best; and, lastly, in memory consumption, FCDD was clearly worse than all the other approaches.

TABLE IV
Detectors statistically inferior to FCDD with 95% confidence in the different metrics.

Kappa	HDDM_A, HDDM_W, WSTD
PMAUC	HDDM_W
μ**Dist**$_1$	DDM
μ**Dist**$_2$	DDM
FP	RDDM, DDM, HDDM_A, HDDM_W, WSTD
TP	DDM, HDDM_W, WSTD
MCC	RDDM, DDM, HDDM_A, HDDM_W, WSTD

VI. Conclusions

This article proposes FCDD, a concept drift detector that performs per-class detections and is meant to deal with imbalanced problems. In scenarios where drifts occur partially, one of its main strategies is to avoid discarding the instances associated with unaffected regions, improving the overall performance of the classifier.

Experimentally, FCDD achieved expressive results in different metrics. Despite its higher memory and run-time consumption, its performance in metrics aimed at imbalanced and multiclass scenarios (κ and PMAUC) was superior or equivalent to the other approaches. Specifically in κ, FCDD was statistically superior to the HDDM_A, HDDM_W and WSTD detectors, while in PMAUC FCDD outperformed HDDM_W.

Regarding the metrics that evaluate the quality of the drift detections, FCDD proved to be very accurate and regular. In addition to achieving a relatively small delay in the detection of the existing concept drifts, this result was accompanied by low false positive rates. Moreover, this good performance was also reflected in the MCC metric, where FCDD was statistically superior to all the other detectors included in the comparison.

As future work, we propose to (a) use other statistical tests in concept drift detections and compare them; (b) experimentally combine FCDD with ensembles specifically targeted at imbalanced data (e.g. [31]); and (c) use some strategy of automatic parameter adjustment, modifying them according to different contexts.

Finally, it is important to note that all material used in this article will be freely available to other researchers.

Acknowledgment

Silas Santos was previously supported by post-doctorate grant number 88887.374884/2019-00 from CAPES. Roberto S. M. Barros is supported by research grant number 305539/2022-1 from CNPq.

Disclaimer

The views and opinions expressed in this research are solely those of the authors and do not necessarily reflect the official policy or position of the company.

References

[1] H.-L. Nguyen, Y.-K. Woon, and W.-K. Ng, "A survey on data stream clustering and classification," *Knowledge and Information Systems*, vol. 45, no. 3, pp. 535–569, Dec 2015.

[2] D. Brzezinski and J. Stefanowski, "Stream classification," in *Encyclopedia of Machine Learning*. Springer, 2016.

[3] S. Wang, L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4802–4821, 2018.

[4] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[5] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya, "A survey on addressing high-class imbalance in big data," *Journal of Big Data*, vol. 5, no. 1, Nov 2018.

[6] R. S. M. Barros and S. G. T. C. Santos, "A large-scale comparison of concept drift detectors," *Information Sciences*, vol. 451-452, pp. 348 – 370, 2018.

[7] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence: SBIA 2004*, ser. LNCS. Springer, 2004, vol. 3171, pp. 286–295.

[8] S. Wang, L. Minku, and X. Yao, "A learning framework for online class imbalance learning," in *2013 IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL)*, April 2013, pp. 36–45.

[9] J. Gama, *Knowledge Discovery from Data Streams*. Boca Raton, FL: Chapman&Hall/CRC, 2010.

[10] A. Bluman, *Elementary statistics: a step by step approach*, 9th ed. New York: McGraw-Hill, 2014.

[11] L. Minku and X. Yao, "DDD: a new ensemble approach for dealing with concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 4, pp. 619–633, 2012.

[12] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, Oct 2011.

[13] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[14] L. Korycki and B. Krawczyk, "Online oversampling for sparsely labeled imbalanced and non-stationary data streams," in *International Joint Conference on Neural Networks (IJCNN)*, 2020.

[15] A. Bernardo and E. D. Valle, "SMOTE-OB: Combining smote and online bagging for continuous rebalancing of evolving data streams," in *IEEE International Conference on Big Data*, 2021, pp. 5033–5042.

[16] Y. Lu, Y.-M. Cheung, and Y. Yan Tang, "Adaptive chunk-based dynamic weighted majority for imbalanced data streams with concept drift," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 8, pp. 2764–2778, 2020.

[17] A. Cano and B. Krawczyk, "Rose: Robust online self-adjusting ensemble for continual learning on imbalanced drifting data streams," *Machine Learning*, vol. 111, no. 7, pp. 2561—-2599, jul 2022.

[18] G. Aguiar, B. Krawczyk, and A. Cano, "A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework," 2022.

[19] D. C. Montgomery, *Applied Statistics and Probability for Engineers, 6th Edition*. Wiley, 2013.

[20] R. S. M. Barros, D. R. L. Cabral, P. M. Gonçalves Jr., and S. G. T. C. Santos, "RDDM: Reactive drift detection method," *Expert Systems with Applications*, vol. 90, pp. 344 – 355, 2017.

[21] I. Frías-Blanco, J. d. Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823, March 2015.

[22] R. S. M. Barros, J. I. G. Hidalgo, and D. R. L. Cabral, "Wilcoxon rank sum test drift detector," *Neurocomputing*, vol. 275, pp. 1954–1963, 2018.

[23] C. Manapragada, G. I. Webb, and M. Salehi, "Extremely fast decision tree," in *Proceedings of the 24th ACM SIGKDD Internat. Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1953–1962.

[24] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '00, New York, NY, USA, 2000, p. 71–80.

[25] S. G. T. C. Santos, R. S. M. Barros, and P. M. Gonçalves Jr., "A differential evolution based method for tuning concept drift detectors in data streams," *Information Sciences*, vol. 485, pp. 376–393, 2019.

[26] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive online analysis," *Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

[27] I. Žliobaitė, A. Bifet, J. Read, B. Pfahringer, and G. Holmes, "Evaluation methods and decision theory for classification of streaming data with temporal dependence," *Machine Learning*, vol. 98, no. 3, pp. 455–482, Mar 2015.

[28] S. Wang and L. Minku, "AUC estimation and concept drift detection for imbalanced data streams with multiple classes," in *International Joint Conference on Neural Networks (IJCNN)*, 2020.

[29] J. Gama, R. Sebastião, and P. Rodrigues, "On evaluating stream learning algorithms," *Machine Learning*, vol. 90, no. 3, pp. 317–346, 2013.

[30] J. I. G. Hidalgo, B. I. F. Maciel, and R. S. M. Barros, "Experimenting with prequential variations for data stream learning evaluation," *Computational Intelligence*, vol. 35, pp. 670–692, 2019.

[31] H. Du, Y. Zhang, K. Gang, L. Zhang, and Y.-C. Chen, "Online ensemble learning algorithm for imbalanced data stream," *Applied Soft Computing*, vol. 107, p. 107378, 2021.