Unsupervised Drift Detection on High-speed Data Streams

Vinicius M. A. Souza
University of New Mexico, USA
vinicius@unm.edu

Farhan A. Chowdhury
University of New Mexico, USA
fasifchowdhury@unm.edu

Abdullah Mueen
University of New Mexico, USA
mueen@unm.edu

***Abstract*—Changes in data distribution of streaming data (i.e., concept drifts), constitute a central issue in online data mining. The main reason is that these changes are responsible for outdating stream learning models, reducing their predictive performance over time. A common approach adopted by real-time adaptive systems to deal with concept drifts is to employ detectors that indicate the best time for updates. However, an unrealistic assumption of most detectors is that the labels become available immediately after data arrives. In this paper, we introduce an unsupervised and model-independent concept drift detector suitable for high-speed and high-dimensional data streams in realistic scenarios with the scarcity of labels. We propose a straightforward two-dimensional representation of the data aiming faster processing for detection. We develop a simple adaptive drift detector on this visual representation that is efficient for fast streams with thousands of features and is accurate as existing costly methods that perform various statistical tests. Our method achieves better performance measured by execution time and accuracy in classification problems for different types of drifts, including abrupt, oscillating, and incremental. Experimental evaluation demonstrates the versatility of the method in several domains, including astronomy, entomology, public health, political science, and medical science.**

***Index Terms*—Data streams, concept drift, drift detection**

I. Introduction

Concept drifts generally represent changes in streaming data caused by underlying changes in the dynamics of data generation. For example, a job change can drastically modify the patterns of power consumption of a given customer measured by smart meters. Consider Fig. 1, two classes - blue diamonds ◆ and red plus signs ✚ - are appearing in a stream. A drift changes the concept of "orientation" of the objects by a 45° rotation, changing them to blue squares ■ and red multiplier signs ✖. Unsupervised concept drift detection is useful in extracting optimal classification performance on streaming data by smartly requesting labels from annotators and retraining soon after or along with the drift. Besides, concept drifts may indicate significant events, and thus provide domain experts with actionable insight. For example, a concept drift in political tweets points to the sway in public opinion.

Initial research on concept drift detection restrictively assumes labeled data are always available, and exploits class labels in detecting drifts [1] - if classifier performance degrades, a drift must have occurred. Recent research work realistically lifts the restriction to detect drifts in an unsupervised manner, and request labels only when retraining is necessary deemed by the detection [2], [3]. Typical unsupervised methods use statistical tests that are rigid to specific confidence levels and maintain required statistics over time [4]–[6]. Such methods are generally costly, mainly on high-dimensional data, since a test is performed for each feature. To elaborate, in order to detect drifts in a Twitter stream with thousands of features, an algorithm has to react as fast as in 20 msec of receiving a tweet. In contrast, a statistical test on such data considering a window of ten minutes takes about 12 sec, which is unacceptable for high-speed data streams. Similar contrast exists in other domains, including agriculture, Internet-of-Things (IoT), and manufacturing, where streams of data objects are produced at 50Hz or more. Therefore, we need techniques that perform a more straightforward computation to identify drifts without lagging behind the stream of data.

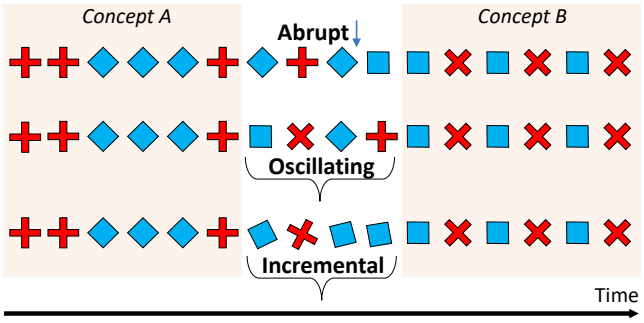


Fig. 1. Three types of drifts over time given two classes represented by geometric figures. *Concept A* (✚, ◆) represents the data before the drift, and *Concept B* (✖, ■) represents the data after the drift.

Concept drifts can be categorized into three kinds (see Fig. 1): i) abrupt, ii) oscillating (also called gradual), and iii) incremental. Abrupt drifts are instantaneous, i.e., an initial concept A sudden changes to a new concept B. Oscillating drifts occur over a window where both concepts A and B are seen at the same time until the concept A is replaced by the new concept B. Incremental drifts occur over a window of time with intermediate concepts between the concepts A and B. Most existing algorithms are designed for one specific kind of drift and are generally less adaptive. For example, recently, Barros et al. [7] experimentally showed that some detectors work best when the drift is abrupt and delays in detecting oscillating drift. Unfortunately, in most real-world problems, the kind of drift is unknown ahead of time, and a single stream can observe all kinds of drifting behavior; hence, we need a single algorithm that can detect them all.

Why the problem is difficult? Keeping up with a high-speed stream while performing data mining tasks is a hard problem. The main reason is that algorithms must be optimized for worst-case scenarios instead of optimizing for average-case scenarios in case of offline mining tasks. Because a series of unfortunate worst-case scenarios can easily accumulate into a lag for which the available memory/buffer would become insufficient. Hence, drift detection on high-speed streams must operate at a fixed worst-case cost per update. In addition, detecting all kinds of drifts is challenging because the algorithm needs to adapt to diverse conditions on the stream. Most data-adaptive hypothesis tests are expensive and are hard to generalize to high-dimensional data. Hence, we need an adaptive algorithm suitable for high-dimensional data.

We propose a technique named *Image-based Drift Detector* (IBDD) to estimate the difference between recent and historical data on a stream. By using a visual 2D representation, we can efficiently detect a drift performing a single image comparison based on pixels differences instead of comparing the differences between data distributions for each feature individually. This simplicity represents an essential speed advantage over the traditional methods when applied to high-speed streams with thousands of features. We empirically demonstrate our findings using real-world datasets on several domains with different kinds of drifts. Also, we present three case studies in public health, political science, and medical science. To support the reproducibility of this work, we made publicly available all our code and the new datasets.

We summarize the main properties of our proposal below:

- IBDD is simple to implement and fast enough to process high-dimensional streams at tens of objects per second;
- IBDD does not depend on labeled data to detect drifts;
- IBDD is adaptive to detect various kinds of drifts, such as abrupt, oscillating, and incremental;
- IBDD is model-independent. The detection is based exclusively on data, without any influence of the predictive algorithm choice and their outputs or performance. It allows us to employ IBDD in supervised or unsupervised mining tasks, as well as for change analysis.

This paper is organized as follows. In Section II, we present definitions of data streams. In Section III, we discuss related works. In Section IV, we introduce IBDD. In Section V, we present the experiments conducted on real-world data. In Section VI, we discuss the performance of IBDD in three case studies. Finally, Section VII concludes this work.

II. Background

A data stream can be defined as an infinite sequence of examples ordered by time $\mathcal{DS} = \{\vec{X}_1, \vec{X}_2, \ldots, \vec{X}_t, \ldots\}$, where $\vec{X}_t$ is a d-dimensional vector in the feature space that was observed at time t. In practice, $\vec{X}_t$ is a list with d descriptive features that represent the observation being made [8]. When d is around hundreds or thousands of features, we have a high-dimensional data streams.

In classification problems, each example $\vec{X}_t \in X$ is associated to a class label $y_t \in Y$. In this task, the stream is a sequence of pairs $\mathcal{DS} = \{(\vec{X}_1, y_1), (\vec{X}_2, y_2), \ldots, (\vec{X}_t, y_t), \ldots\}$. The goal of a model $\mathcal{L}_i$ induced from a historical data $\mathcal{DS}_i = \{(\vec{X}_1, y_1), \ldots, (\vec{X}_i, y_i)\}$ is to predict the class label y_t of the example $\vec{X}_t$ observed in a future time t, where $t > i$.

Many real-world problems are dynamic with data generated on evolving environments which leads to changes in the underlying distribution of the stream. This change is named *concept drift*, where the data distribution in a given moment is called *concept* and the *drift* represents a change in such distribution in a given time [6]. Formally, a concept drift occur between the times t and $t + \Delta$, if $P_t(X, Y) \neq P_{t+\Delta}(X, Y)$, where P_t refers to the joint distribution at time t given a set of examples (X) and their class labels (Y).

We can categorize the concept drifts into two groups according to the type of change: i) real and ii) virtual [6]. In real drift, also called class drift or prior probability shift, the change affects the class-conditional likelihoods $P(Y|X)$, while the features $P(X)$ can change or not. In virtual drift, also called feature change or covariate shift, the distribution of the features $P(X)$ changes over time, while the boundaries among the classes remain unchanged. Thus, in classification problems, virtual drifts do not affect the classifier's predictive performance. Unsupervised drift detectors, as proposed in this work, can detect virtual drifts and real drifts. However, for real drifts, it is needed that the changes also be observable in the input features $P(X)$. When the changes are exclusively in $P(Y|X)$, it is not possible to catch them without labeled data [4]. At first glance, this limitation may seem very restrictive. However, this type of change is common in several problems, as discussed in our experimental evaluation with real data and case studies from different domains.

Drifts constitute a central issue on stream learning since it is responsible for the outdating of predictive models, leading to decreases in their accuracy. To deal with this problem is common for stream classifiers, the use of a drift detector to indicate the best time for the model update with recent data, as proposed in this work. There is a vast body of works about drift detection [4]–[6], [9], [10], in Section III, we discuss the related ones of our proposal.

III. Related work

Novel drift detectors are proposed every year. However, most of them, such as the state-of-the-art DDM [1] and ADWIN [10], are supervised methods which require that the ground-truth be available almost immediately after every prediction for monitoring the error rate and verify when the classifier is outdated. The assumption of label availability after predictions is very optimistic and not realistic in many situations due to the high cost of manual labeling a massive amount of fast arrival data, restricting the use of these detectors in real problems [11], [12]. Also, these detectors are model dependent, which means that their results can vary according to the chosen predictive algorithm and their outputs.

Although semi-supervised learning is a popular approach, using partially labeled data is still poorly explored for drift detection. An example is the SUN algorithm [13]. The general

idea is to build a growing decision tree incrementally and generates concept clusters in the leaves using labeled information. The unlabeled data are then labeled according to the majority-class of their nearest cluster. For drift detection, SUN considers the difference between old and new concept clusters.

Unsupervised drift detectors focus on finding changes without any label information. In general, such methods are based on the use of statistical hypothesis tests performed on two groups of data, one drawn from the past in a reference window and another with the most recent data in a sliding window. If the test rejects the null hypothesis that both groups are from the same distribution, a drift is detected. After the detection, the reference window is updated with recent data, and new comparisons are made with the new incoming streaming data [9]. Common choices for statistical tests are the univariate tests Kolmogorov-Smirnov, Two-sample T-test, Wilcoxon Rank Sum, Wilcoxon Signed-rank, and Wald-Wolfowitz [5].

In Žliobaitė [4], three different information are used by statistical tests for unsupervised drift detection: i) raw data, ii) classifier output, and iii) estimated class labels. The first approach of monitoring raw data is the most usual technique, where the test is performed on each feature of the recent data compared with the reference data to determine whether both come from the same population [2], [5], [6], [9]. Concept drift is detected if one or a subset of features show statistical differences. The advantage of this approach is that it works independently of a classifier but can be time costly on high-dimensional data. In this direction, Reis et al. [2] proposed the Incremental Kolmogorov-Smirnov test (IKS). IKS is a speedup of the original Kolmogorov-Smirnov test for data streams, which reduces the complexity of $O(N\log N)$ to $O(\log N)$ for the comparison of two samples with N unidimensional examples. The general idea is employing a randomized tree for the insertion and removal of data during the recomputing of statistics in an incremental manner taking advantage of previously computed values. However, the test still needs to be performed individually for each d feature, which can be costly when we have thousands of features.

In the second approach discussed by Žliobaitė [4], the statistical test considers the output data of the classifier, such as the score or the probability of the examples to belonging a class. The dependence of a classifier is a drawback of this approach since the wrong choice of a learner can drastically change the results. Besides, predicting well-calibrated probabilities is still a challenge for many supervised learning algorithms [14]. In the latter approach, the statistical test considers the class labels outputted by a classifier. In this case, it is only possible to detect changes in the proportions of the classes $P(Y)$ over time, which typically has less impact on the accuracy.

For multidimensional tests, we can mention the Kullback-Leibler distance to detect drifts [15]. However, such a test considers labeled data to compare the probability distribution for each class. Moreover, Alippi et al. [16] have experimentally shown that the detectability worsens when the number of dimensions increases, even at reasonably low-dimensions (e.g., ten features).

IV. IBDD: Image-Based Drift Detector

IBBD is a model-independent and unsupervised drift detector for data streams that uses a visual representation to monitor changes in data distributions over time. While traditional statistical tests need to compare the differences between the distributions of each feature, the visual representation employed by IBDD allows performing such comparison efficiently in a single pass procedure. It represents an important advantage over the traditional methods when applied to high-dimensional data such as time series, text, and computer vision problems which typically has thousands of features.

Our detector is simple to implement and works independently of a classifier, which means that it can be used in supervised tasks such as classification or in tasks with fully unlabeled data such as data analysis and online monitoring to provide warnings about significant changes in data. IBDD is adequate as a trigger mechanism to indicate the best time for model updates, aiming to maintain predictive performance in evolving environments. The detections are based exclusively on the data, without any influence of the predictive algorithm choice. In classification, the detector can identify both virtual and real drifts when the changes are observable in the features.

To detect drifts, IBDD uses two data windows ($w1$ and $w2$) with the same size w, where $w1$ is a reference window which contains data of a known and stable concept (in classification problems, it represents the training data used to build an initial model), and $w2$ is a sliding window updated continuously with each new example that arrives from the stream over time.

The d-dimensional data from $w1$ and $w2$ are then converted into two different 2-dimensional gray-scale images. In this image, the pixels' intensities represent the feature values of each example into the window. Each row of the image represents a feature, while an entire column represents an example with d features. In this manner, it is possible to represent a sequence of d-dimensional examples visually in two dimensions with height p equal to the number of features and width q equal to the number of examples into the window.

To better illustrate the visual representation used by IBDD, consider the images I_{w1} and I_{w2} shown in Fig. 2.

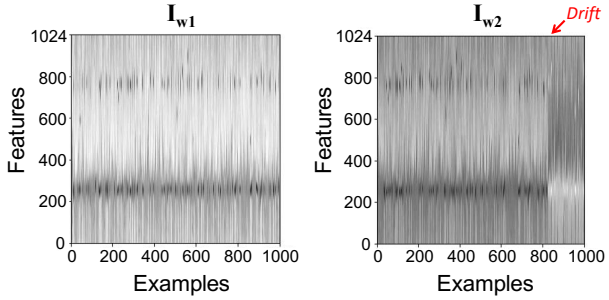


Fig. 2. Example of the visual representation used by IBDD given two data distributions drawn from different periods on the StarLightCurves dataset.

The images of Fig. 2 represent two data distributions drawn from a different time of one of the evaluated datasets (StarLightCurves). The windows have 1,000 examples with

1,024 features each one. Thus, the images have a height of 1,024 pixels and a width of 1,000 pixels. We also indicate the drift point in I_{w2}. It is interesting to note the contrast change in the entire image I_{w2} in addition to the change in the pixel intensities after the indicated drift. This contrast change occurs due to the normalization in the sliding window data, and it helps to identify the differences in the compared distributions.

After the conversion of windows $w1$ and $w2$ into the images I_{w1} and I_{w2}, we can measure the similarity between them to find significant changes in data distribution. As $w2$ is a sliding window updated with each new arrival example from the stream, the comparison between the images is performed continuously.

To measure the similarity between I_{w1} and I_{w2}, we employ the simple and straightforward metric Mean-Squared Deviation (MSD), as defined in Eq. 1. Both images need to have the same dimension with $p \times q$ pixels, according to the height and width, respectively.

$$MSD(I_{w1}, I_{w2}) = \frac{1}{p \times q} \sum_{i=1}^{p} \sum_{j=1}^{q} (I_{w1_{(i,j)}} - I_{w2_{(i,j)}})^2 \quad (1)$$

The general idea of MSD is to measure the difference between the pixel intensities of two images. The MSD has the property of non-negativity, and a value of 0 indicates perfect similarity between the compared images. MSD is symmetric, which means that $MSD(I_{w1}, I_{w2}) = MSD(I_{w2}, I_{w1})$.

According to the similarity value returned by MSD, we flag a drift when a sequence with m values are above a superior threshold or below an inferior threshold. To avoid false alarms due to noise, we recommend $m > 1$. While statistical tests return a probability value (p-value), which represents the level of confidence to reject the null hypothesis that two data are from the same distribution, the value returned by MSD does not have the same meaning and their values can drastically change according to the dataset. For this reason, IBDD dynamically updates its thresholds on stable periods and when a drift is detected.

Before processing the stream examples, we estimate the initial values for the superior and inferior thresholds using the data provided as input, assuming that such data are stationary. We compare the image I_{w1} generated from the initial data against a set of different images obtained by k random permutations of the same data. The superior threshold is defined by the mean of the observed values plus two times the standard deviation, while the inferior threshold is defined by the mean of the observed values minus two times the standard deviation. The number of permutations k is a hyperparameter of IBDD, which was fixed on 20 in all experiments. For stationary data, the variability of the values is small, and a reduced number of iterations is sufficient for a reasonable estimate.

In stable periods, the superior and inferior thresholds are dynamically updated following the same rule employed in the initial estimate, but considering the mean and standard deviation of the past MSD values since the last update. We consider stable periods when no drifts were detected in the last 50 consecutive examples.

IBDD also updates both thresholds ever time a new drift is detected. The update is needed because the reference image I_{w1} is the same over the entire stream, and after a drift, the similarity relation between the reference data and the new concept changes. In this case, if we observe a sequence with m values above the superior threshold, it is updated with the last MSD value, which flagged the drift, plus the standard deviation of the last values since the last update. To update the inferior threshold, we consider the last MSD value minus the mean of differences between superior and inferior thresholds in the past. This updating procedure is analogs when the drift occurs due to the values below the inferior threshold.

Fig. 3 illustrates the detection of six drifts and the dynamic thresholds. Each blue circle represents the similarity between the windows measured by MSD. The example concerns the Yoga dataset, which will be introduced in the next section.

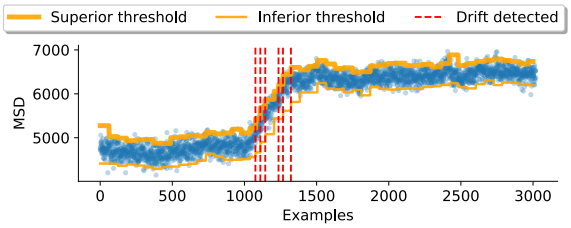


Fig. 3. Similarity values, drifts detected by IBDD and thresholds updates.

V. Experimental Evaluation

To support the reproducibility of our work, we have built a website[1] where we made available the codes and data, as well as additional results.

A. Setup

In general, the evaluation of drift detectors is carried out considering synthetic data, where the generation follows some specific distribution, and the type and location of drifts are known beforehand. Synthetic data allows the use of measures that consider the rate of true/false alarms and the delay of detections [17]. Although the evaluation is facilitated, a clear limitation is that real data and changes are more complex. Thus, the performance of detectors evaluated on synthetic data can differ in practice. In this direction, we prioritize the use of real-world data in our evaluation. Given the lack of ground truth regarding the drifts' location, we consider the accuracy of a classifier updated according to a drift detector as an indirect performance measure. The assumption is that more precise detectors help the classifiers to achieve stable results under changes if the model is updated at the right times. It is essential to note that we use labeled data only to evaluate our method, and the detection does not depend on any label.

We compare the performance of a classifier updated with IBDD against two nonparametric statistical tests which use

[1]Supporting website: https://sites.google.com/view/ibdd-paper

only the features information for unsupervised drift detection: i) Incremental Kolmogorov-Smirnov (IKS)[2] [2] and ii) Wilcoxon Rank-Sum (WRS) [5]. In all experiments, we consider the same significance level employed by [2] of $\alpha = 0.001$. Two main reasons justify the choice of the statistical tests to compare our proposal: i) given the simplicity and effectivity of this approach, it is a well-known and common choice for drift detection (e.g., [2], [4], [5], [9]); and ii) for a fair comparison, this approach is the most similar with ours, being unsupervised and model-independent detectors.

Besides, we also consider the following strategies that represent the worst and best case for model adaptation:

- ***Baseline***: a static classifier that does not have a detector and never updates its model. This naive classifier helps to understand drifts' impact on predictive performance when no action is taken to deal with changes;
- ***Topline***: a dynamic classifier that uses a persistent detector that identifies drift for every stream example leading to continuous model updates. This approach is the closest strategy to maintain a classifier updated with the most recent data over all the time, as well as the most costly.

We consider Random Forest as the base classifier for all detectors given their high learning performance and the low number of parameters. The experiments were performed in an Intel Core i7-9700 @ 3.00GHz, 32GB of memory, running the Windows 10 64-bit. All algorithms were coded in Python 3.

B. Time efficiency and complexity

To evaluate the time efficiency of IBDD on high-dimensional data, we measure the costs varying the number of features on synthetic data. Fig. 4 shows the time results in seconds to perform 100 comparisons of windows with 1,000 examples and dimensionality ranging from 10 to 10,000. On the left side of the figure, we detail the results until 100 features, while on the right side, we have a more general view.

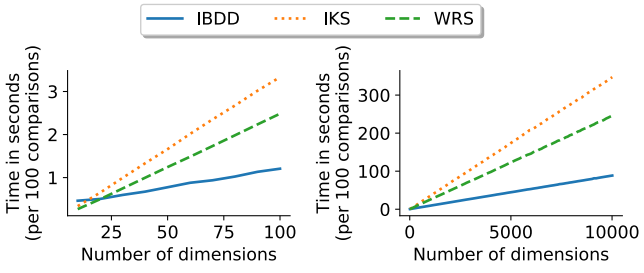


Fig. 4. Time spent in seconds by the detectors to perform 100 comparisons of windows with 1,000 examples and dimensionality ranging from 10 to 10,000.

IBDD becomes time competitive starting from 25 features. However, we recommend our detector on high-dimensional problems with more than 50 features. In such cases, we can note a considerable advantage of IBDD over the methods.

In terms of time complexity, the two main steps of IBDD are the image generation from the sliding window and the comparison with the reference image from the training data. Both steps are performed for each stream example and have linear complexity according to the number of examples into the windows and the number of features or $O(p \times q)$, where p and q are the height and width of the images. For space complexity, IBDD spends $O(w \times d)$ to store the most recent d-dimensional examples into the sliding window $w2$.

[2]Python implementation provided by the authors.

C. Results on real-world datasets

We compare the performance of our method on six real-world benchmark datasets from different domains, prioritizing problems with high dimensionality. In some cases, we modified the original data to include concept drifts. However, in all cases, the changes can be justified in practice. On the following, we describe the datasets.

Heartbeats [18], [19]. This dataset contains ECG data from the Sudden Cardiac Death Holter Database (SDDB) of Physionet [18]. We consider the version previously evaluated by Ulanova et al. [19] for stream clustering, which uses the data from a specific patient (subject #51) with high variability. The dataset contains individual heartbeats extracted about once every 0.86 seconds of a stream 25 hours long. Each example of this dataset has 280 values and eight possible classes, such as normal beat, paced beat by the patient's pacemaker, premature ventricular contraction, supraventricular premature beat, and different beats fusions;

Flying Insects [19]. The task is identifying the species of flying insects based on the wing-beat information measured by optical sensors. The data generated by the sensor are audios recorded when the insects fly through a laser light. It is possible to recognize different species based on frequency and harmonics position in the spectrum by converting the audios to the frequency domain. The dataset has three classes: *Ae. aegypti* ♂, *Ae. aegypti* ♀, and *Cx. quinquefasciatus* ♀. It is expected drifts due to variations in environmental conditions, responsible for changing the flight characteristics of the insects [8], [20], [21] (e.g., see the impact of temperature in Fig. 5). Also, some erratic flights can generate noise examples;

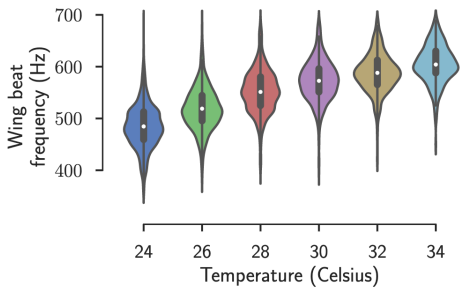


Fig. 5. Wing-beat frequency of *Ae. aegypti* ♂ according to temperature [22].

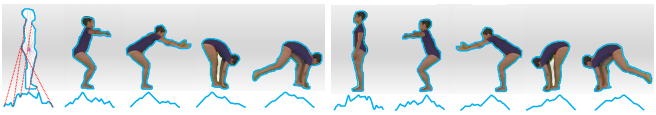
Sensor Posture [23]. These data are from a multi-agent system for the care of older people. The system provides flexible activity monitoring, reacting to critical situations as calling for help in the case of an emergency, and issuing warnings if unusual behavior was detected. Each example contains the localization of three wearing tags installed in the

ankle left, ankle right, and chest of a person. The original data has 164,860 readings obtained from five persons and does not have drifts. In this work, we consider the modified version proposed by [2], where the ordering of the examples has sequences produced by the same person to induce drifts;

StarLightCurves [24]. This is a time series dataset where each example has 1,024 features representing the brightness of a celestial object as a function of time. The dataset has pre-defined training and test sets with 1,000 and 8,236 examples, respectively. We introduce drifts in the test set multiplying the values of each instance by -1, starting on the example 2,000. We named this modified version as StarLightCurves-YReversed. In practice, this change can be justified by a malfunction in the sensor responsible for the measurements;

UWaveGestureLibraryY [24]. This data contains a set of eight gestures measured by an accelerometer device in the y-axis. The dataset has pre-defined training and test sets with 896 and 3,582 examples, respectively. We introduce incremental drifts changing the sensor orientation around the x-axis six times in an angle of 30° each change. We named this version as UWaveGestureLibY-Incremental. This kind of drift is justified in practice since motion devices such as a smartwatch in the user's wrist can slowly change their position;

Yoga [24]. This is a time series dataset generated from images with actors performing yoga movements. The problem is to discriminate between one actor (male) and another (female). Each image was converted to a one-dimensional series through the distances among the actor's contour to the image center. The dataset has pre-defined training and test sets with 300 and 3,000 examples, respectively. We introduce drifts by flipping the x-axis of the series, starting on the example 1,000. We named this modified version as Yoga-XReversed. In practice, this drift represents a change in the orientation of the actors performing the yoga movements. To better illustrate, we show in Fig. 6, five examples from this dataset. In this figure, it is possible to see the image conversion to time series, and the effect on data when the actor changes their side.



(a) Yoga movements and series (b) Flipped images

Fig. 6. Examples of images from Yoga dataset and corresponding time series obtained by the distances among the actor's contour and the image center. A horizontal flip in the image generates a flip in the x-axis of the time series.

Table I shows the datasets characteristics, where only Posture does not have a high number of features. We have included it to evaluate the performance of IBDD with low dimensional data. We also indicate the size of the initial training data to build the first model. To standardize our experiments, we consider the same number of training examples for the parameter concerning the window size and $m = 3$ for all cases.

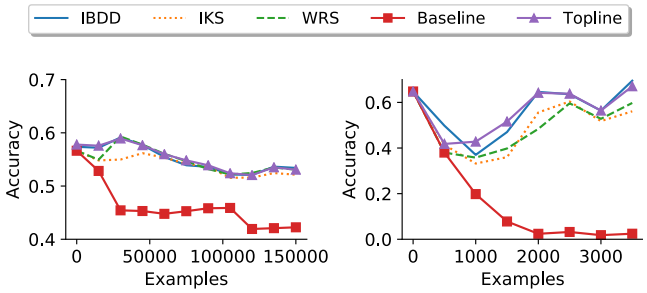
TABLE I
DESCRIPTION OF THE REAL-WORLD DATASETS. DRIFT KINDS ARE ABRUPT (A), INCREMENTAL (I), AND OSCILLATING (O).

ID-Abbrev.	Dataset	Class	Examples	Feat.	Train	Drift
1-HBeats	SDDB Heartbeats (#51)	8	77,904	280	500	O
2-Insects	Flying Insects	3	86,400	200	1,000	A
3-Posture	Person Activity	11	164,860	3	2,000	I
4-SLCur	StarLightCurves-YRev	3	9,236	1,024	1,000	A
5-UWave	UWaveGestLibY-Inc	8	4,478	315	896	I
6-Yoga	Yoga-XRev	2	3,300	426	300	A

In Table II, we show the average accuracies of the classifiers updated according to detectors and the average ranking provided by the Friedman test. Given the three detectors, we highlighted the best result for each dataset. IBDD is ranked in the first position among the competitors and similar to the supervised Topline. Specifically for the Yoga dataset, IBDD slightly outperformed the Topline detector with a drastically lower number of model updates (6 times $vs.$ 3,000 times) and labels requested for model update (18.2% $vs.$ 100%). Such points of updates can be seen in Fig. 3. In Fig. 7, we present the accuracy over time for Posture and UWave.

TABLE II
ACCURACY RESULTS FOR REAL-WORLD DATASETS.

Dataset	IBDD	IKS	WRS	Baseline	Topline
1-HBeats	**98.72**	98.62	98.62	96.67	98.79
2-Insects	**95.09**	95.03	95.04	61.29	95.22
3-Posture	**55.09**	53.97	54.91	46.26	55.28
4-SLCur	**91.96**	91.44	91.06	23.22	92.06
5-UWave	**55.08**	49.19	48.72	19.29	55.33
6-Yoga	**79.83**	76.83	77.17	56.23	78.07
Avg. rank	**1.833**	3.583	3.416	5	1.166



(a) Posture (b) UWave

Fig. 7. Accuracy for Posture and UWave datasets.

We show in Table II that IBDD can provide results slightly superior to the state-of-the-art unsupervised detectors or even to the label costly Topline approach which considers continuous updates. However, the main advantage of IBDD is time efficiency accompanied by accurate results. To better illustrate the time efficiency of IBDD on real data, Table III shows the time in seconds spent by the classifier coupled with a given drift detector to process a whole dataset, as well as the average time to process each example. As expected, IBDD loses in time only for Posture dataset, which has three features. For high-dimensional datasets, such as StartLightCurves with 1,024 features, IBDD was about 10× faster than IKS.

TABLE III
TIME RESULTS IN SECONDS (TOTAL/PER EXAMPLE).

Dataset	IBDD	IKS	WRS	Baseline	Topline
1-HBeats	**1,650/0.021**	28,917/0.375	7,703/0.100	349/0.004	6,887/0.089
2-Insects	**2,255/0.026**	18,327/0.215	7,212/0.084	372/0.004	22,158/0.259
3-Posture	1,633/0.010	**976/0.006**	1,114/0.007	709/0.004	17,438/0.107
4-SLCur	**728/0.088**	7,735/0.939	3,345/0.406	36/0.004	3,679/0.447
5-UWave	**96/0.027**	1,125/0.314	473/0.132	16/0.004	972/0.271
6-Yoga	**56/0.019**	551/0.183	410/0.137	13/0.004	363/0.121

The relation between time and error is illustrated in Fig. 8. In this image, each point represents a dataset (also represented by their ID), different markers distinguish the detectors. The best results are located close to the origin $(0, 0)$.

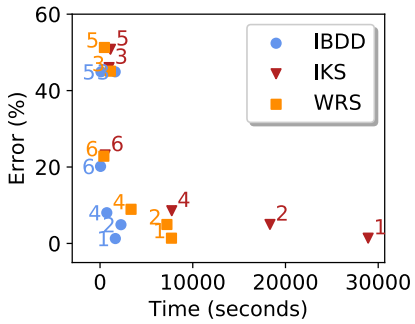


Fig. 8. Relation between time and error. Each point is a dataset, markers represent detectors and the best results are located close to the origin $(0, 0)$.

D. Parameter sensitivity

IBDD has two parameters: i) the number of examples into the windows to perform the image comparisons and ii) the number of MSD values above/below the threshold to indicate a drift (m). In our experiments, we set the window size with the same size as the initial training set for each dataset and $m = 3$. To show the impact of such parameters, we perform an experiment varying their values.

For the window size parameter, we were varying the values from 20% to 100% of the training set size, according to the $|Train|$ previously reported in Table I, and set $m = 3$. In Fig. 9, we show the accuracy results varying the window size.

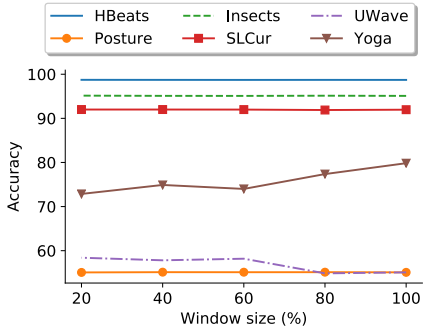


Fig. 9. Accuracy results varying the window parameter.

The most impacted datasets were Yoga and UWave, in which the accuracy had changed from 72.87% to 79.83% and from 54.86% to 58.40%, respectively. For Yoga, the worst accuracy was observed in small windows with only 60 examples. For UWave, the best result was achieved with a small window containing 179 examples.

Although the window size slightly impacts the accuracy for most cases, this parameter affects the number of drifts detected and the runtime. In Fig. 10, we can note that the number of drifts detected, and consequently, the number of false-positive detections, are reduced with larger windows, while the average time to process each example increases.

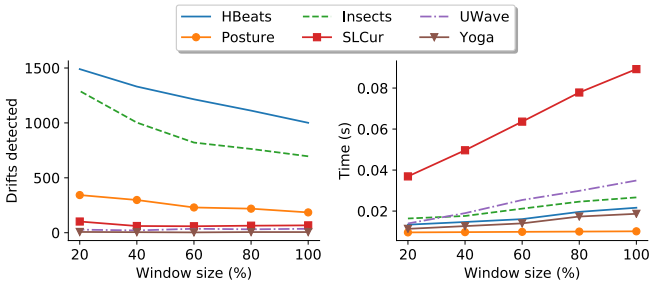


Fig. 10. Number of drifts detected and average time of IBDD to process each example varying the window size.

Concerning the number of consecutive MSD values above or below the drift threshold, we were varying the values of ϵ from 1 to 9. In general, a small value allows a higher number of false drifts detected due to the occurrence of noise or outliers. On the other hand, a higher value can delay true drifts detection. In Fig. 11, we show the impact of ϵ on the accuracy and number of drifts detected. For most cases, $\epsilon = 3$ can avoid a higher number of detections.

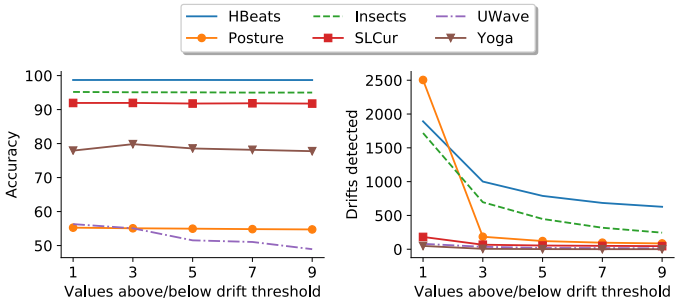


Fig. 11. Accuracy and number of drifts detected by IBDD given the variation of parameter m.

E. Distance measures

An essential part of our solution is the similarity measurement between the two-dimensional images that represent the multidimensional data over the stream. At this point, we have shown that a simple approach such as MSD can be a fast and accurate measure for IBDD. However, the employment of a different measure is an open choice of our proposal.

In this section, we present the performance of IBDD using different measures. Specifically, we compare MSD with two other measures: Structural Similarity Index (SSIM) [25] and CK-1 [26]. The SSIM is a well-known measure mainly employed for evaluation of image quality by modeling image distortion as a combination of three factors: loss of correlation, luminance distortion, and contrast distortion. CK-1 is a Kolmogorov complexity-based distance that uses video compression algorithms to estimate image similarity. Although the original proposal of CK-1 employs the first generation of MPEG video codec for compression (MPEG-1), we consider the fourth generation (MPEG-4) in our experiments. In Table IV, we show the accuracy results of this experiment.

TABLE IV
ACCURACY OF IBDD WITH DIFFERENT DISTANCE MEASURES.

Dataset	MSD	SSIM	CK-1
1-HBeats	**98.72**	98.68	98.60
2-Insects	95.09	**95.11**	95.05
3-Posture[3]	**55.09**	– –	– –
4-SLCur	**91.96**	80.85	80.84
5-UWave	**55.08**	37.41	52.79
6-Yoga	**79.83**	57.86	56.23

VI. CASE STUDIES

In this section, we present three case studies on public health, political science, and medical science. These domains represent problems with abrupt, oscillating, and incremental changes in data with thousands of features.

A. Abrupt drift detection for classification of Malaria vectors

Malaria is a disease transmitted by infected mosquitoes which kills more than 600,000 people yearly, the majority being African children under 5 years of age [27]. The identification of mosquito species is a crucial task for the estimation of Malaria transmission dynamics, surveillance, and evaluation of interventions [28]. Identifying species is challenging because some species can only be distinguished by molecular analysis, given their high morphological similarity [29]. For example, *Anopheles gambiae* and *Anopheles funestus* are two very similar species of mosquitoes that constitute the bulk of malaria transmission in Africa. An accurate method to distinguish them is the Polymerase Chain Reaction (PCR), which is time-consuming and expensive, being unpractical in field conditions and for large-scale analysis [30].

Recently, Gonzalez-Jimenez et al. [28] showed that mid-infrared spectroscopy (MIRS) is a fast (few seconds per mosquito), practical, and cost-effective method for mosquito analysis without any molecular identification. In this spectroscopy technique, the light interacts with the fundamental vibrations of the biomolecules present in the mosquito's cuticle, creating an absorption spectrum of discrete well-delineated bands in the mid-infrared region (400-4,000 cm^{-1}).

[3]SSIM and CK-1 require images with a minimum size for all dimensions. As Posture has only three features, both measures were not able to compare the small generated images for this dataset.

In Fig. 12, we illustrate the mid-infrared absorption spectra for two examples of malaria mosquitoes from the species *Anopheles gambiae* and *Anopheles arabiensis*.

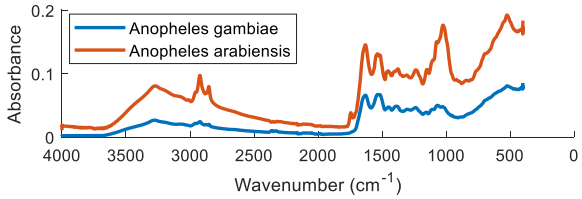


Fig. 12. Mid-infrared absorption spectra of two malaria mosquito species.

To perform the automatic identification of the species, we use a spectrum composed of 3,600 observations as input for a classifier. Besides being a high-dimensional data, the mid-infrared spectral bands are affected in non-trivial ways by the development of the mosquito and the changing composition of the cuticle over time [28]. Thus, this problem requires an adaptive approach to provide accurate predictions under these changes. To maintain the speed advantage provided by the mid-infrared spectroscopy, the mechanism for drift detection and model update need to be efficient.

Based on the public data provided by Gonzalez-Jimenez et al. [28], we use the meta-information related to the age of the mosquitoes to build a streaming dataset with abrupt changes. This data contains 4,260 examples from malaria vector species *Anopheles gambiae* and *Anopheles arabiensis* with age varying from 1 to 17 days. In our stream dataset, we consider an initial training set composed only by mosquitoes with age between 8 and 9 days (486 examples). In the test set, we consider the following sequence of age for the arrival of the 3,774 remaining examples from the stream: $(17, 16, 3, 4, 15, 14, 5, 6, 13, 12, 1, 2, 11, 10, 7)$, where each color transition represents an abrupt concept drift.

For this data with abrupt changes, a static classifier shows an accuracy of 69.61%, while the use of a drift detector can improve this result in about 12%. Table V shows detailed results achieved by the evaluated detectors, where IBDD ($m = 1$) presents the best accuracy result, spending about $8\times$ less time than WRS, and $29\times$ less time than IKS.

TABLE V
ACCURACY AND TIME FOR THE MALARIA MOSQUITO SPECIES PREDICTION CASE STUDY.

	IBDD	IKS	WRS	Baseline	Topline
Accuracy	**81.37**	80.23	79.70	69.61	82.33
Time (sec.)	**560**	16,722	4,504	17	1,611

In Fig. 13, we show the accuracy over different age groups, where is possible to see that a classifier trained on 8- and 9-day-old mosquitoes have difficulty in identifying 1-, 2-, 3-, 4-, and 7- day-old mosquitoes.

B. Oscillating drift detection on tweets

Twitter distributes short posts via a streaming API. One can receive tweets as fast as fifty tweets-per-second. Drifts in

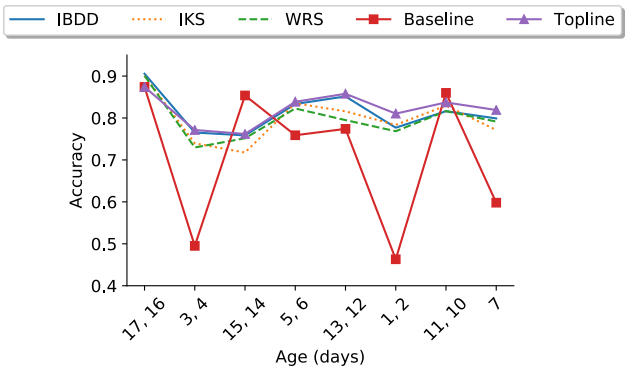


Fig. 13. Accuracy for malaria mosquitoes prediction according to age.

tweets can indicate a sudden or gradual change in the trending topics and change in the type (e.g., political orientation) of posters. Detecting such drifts can be time-critical when sudden events as an earthquake or terrorist attack happen.

In this case study, we consider 10,841 political tweets related to the 2016 U.S. Presidential Election from November 4 to November 13, 2016 [31]. The election was held on November 8. The 200 most active users generated the collected data. Each tweet is represented in an embedding space pre-trained on 2B tweets of GLoVE[4] word-embedding, totaling a feature vector with 2,000 dimensions and without class labels. To detect drifts on this unsupervised problem, we apply IBDD with the following parameters: the reference window ($w1$) comprises 2,251 tweets generated on the first 48 hours, the sliding window ($w2$) has the same size of the reference window, and $m = 4$. Since the data is unlabeled, this case study is an example where a model-independent detector is required to identify changes. Fig. 14 shows the drifts points.

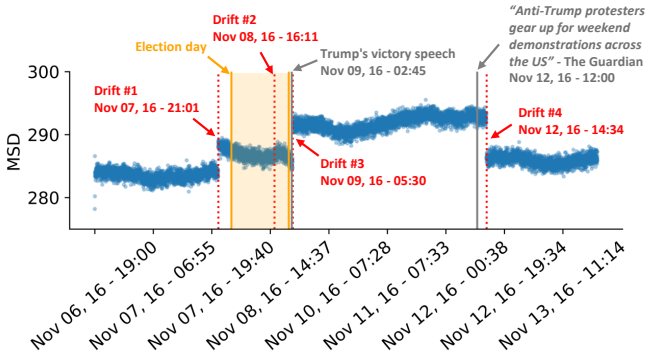


Fig. 14. Four drifts detected in the Twitter data and their description.

We identify four drift points. The drifts mark i) the evening before the election day, ii) the time when voting was on, iii) the time of the victory speech by President Donald Trump, and iv) the time when thousands of protesters marched to Trump Tower in New York in a rally planned in the social networks[5]. The drifts identify changes in discussion topics from campaigns to exit polls to election results to protest - an oscillating drift that captures the oscillation in the political orientation of the tweets on discussion topics. For comparison, IKS and WRS also have found drifts in similar positions. However, while IBDD spent 53 minutes, IKS spent 6 hours and WRS 2 hours and 30 minutes.

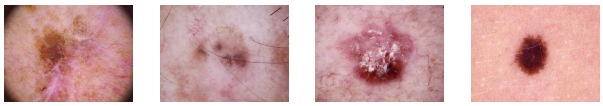
[4] https://nlp.stanford.edu/projects/glove/
[5] https://bit.ly/39cy83O

C. Incremental drift detection for skin lesion classification

Skin cancer is the most common malignancy in the world, affecting men and women of all ages and skin colors [32]. According to the World Health Organization, 2 to 3 million cases of non-melanoma and more than 132,000 cases of melanoma are diagnosed each year.

When detected early, the 5-year survival rate for melanoma is 98%, but 23% for late stage detection [33]. In this direction, the automatic prediction of skin diseases using dermoscopic images represents an important and low-cost tool for preliminary diagnoses and early intervention.

In this case study, we consider 10,015 dermatoscopic images of pigmented lesions from a public dataset HAM10000 [34]. This dataset contains data collected over a period of 20 years from different sites. A feature vector with 2,353 dimensions representing the mean pixel value of the RGB channels of the images describes each example. The dataset has seven classes according to the disease categories: actinic keratoses, basal cell carcinoma, benign keratinocyte lesions, dermatofibroma, melanoma, melanocytic nevus, and vascular lesions. Fig. 15 illustrates four real examples from the dataset.



(a) Mel (b) Bcc (c) Akiec (d) Nv

Fig. 15. Examples from the HAM10000 dataset. The diagnoses are: Mel (Melanoma), Bcc (Basal cell carcinoma), Akiec (Actinic keratosis), and Nv (Melanocytic nevi).

To build a streaming data with incremental drifts over time, we sort the arrival order of the examples according to the patient's age. Aging is directly responsible for changing the characteristics of the skin, such as texture, color, among others. The risk of skin cancer due to the accumulated sun exposure over the years increases with age [35]. Thus, the classes are impacted by the age of the patient as well as the age of the disease. The dataset has images from patients with ages ranging from 0 to 85 years. In our stream dataset, we consider an initial training set composed only by images from patients with age between 75 to 85 years (1,312 examples). In the test set, the arrival sequence follows a decreasing trend in the age from 75 to 0 (7,944 examples).

For this data with incremental age changes, a static classifier shows 64.20% of accuracy, while the use of a drift detector can improve the result in about 13%. Table VI shows detailed results. Besides the best accuracy, IBDD ($m = 3$) is $10\times$ faster than IKS and $4\times$ faster than WRS.

In Fig. 16, we show the data similarities output by IBDD as well as the position of the 29 drifts detected by our method.