Knowledge and Information Systems (2021) 63:1497–1527
https://doi.org/10.1007/s10115-021-01564-6

REGULAR PAPER

Efficient unsupervised drift detector for fast and high-dimensional data streams

Vinicius M. A. Souza[1,2] · Antonio R. S. Parmezan[3] · Farhan A. Chowdhury[2] · Abdullah Mueen[2]

Received: 6 November 2020 / Revised: 13 March 2021 / Accepted: 22 March 2021 / Published online: 9 April 2021

Abstract
Stream mining considers the online arrival of examples at high speed and the possibility of changes in its descriptive features or class definitions compared with past knowledge (i.e., concept drifts). The fast detection of drifts is essential to keep the predictive model updated and stable in changing environments. For many applications, such as those related to smart sensors, the high number of features is an additional challenge in terms of memory and time for stream processing. This paper presents an unsupervised and model-independent concept drift detector suitable for high-speed and high-dimensional data streams. We propose a straightforward two-dimensional data representation that allows the faster processing of datasets with a large number of examples and dimensions. We developed an 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 considering each feature individually. Our method achieves better performance measured by execution time and accuracy in classification problems for different types of drifts. The experimental evaluation considering synthetic and real data demonstrates the method's versatility in several domains, including entomology, medicine, and transportation systems.

Keywords Data stream · Concept drift · Unsupervised drift detector

✉ Vinicius M. A. Souza
vinicius@ppgia.pucpr.br

Antonio R. S. Parmezan
parmezan@usp.br

Farhan A. Chowdhury
fasifchowdhury@unm.edu

Abdullah Mueen
mueen@unm.edu

1 Pontifícia Universidade Católica do Paraná, Curitiba, PR, Brazil

2 University of New Mexico, Albuquerque, NM, USA

3 University of São Paulo, São Carlos, SP, Brazil

1 Introduction

The emergence of new technologies, such as those related to smart sensors, has made each individual and his daily activities a rich source of continuous data. For example, in a few days, a single person can produce a considerable volume of data from its electricity consumption measured by smart meters, physical activities using a smartwatch, trips recorded by GPS, among other daily activities. In this scenario, it is common to use machine learning models to understand the data source and their environment, aiming to provide intelligent applications with customized responses. The data mining community calls this kind of dynamic data generated continuously in real time and in an ordered way as data streams [11,15].

Among the challenges imposed by data streams, concept drift is one of the main and focus of active research in the last decades [15,41]. Concept drift represents the changes in data characteristics over time due to the source generation's evolving behavior. For instance, consider how impactful a job change is on the person's routine and its generated data. In this manner, when we employ a machine learning model to make decisions using streaming data, concept drift can drastically reduce its predictive performance. For this reason, data streams require efficient mechanisms to maintain the model updated according to the most recent data.

To handle concept drift, passive or active mechanisms can be used by the algorithms for adaptation [11]. Passive approaches are blind solutions where the model is continuously updated, independently if a drift effectively has occurred or not (e.g., [18,42,44]). On the other hand, active approaches explicitly identify changes in data by drift detectors for then retraining the model (e.g., [3,14]). In general, active approaches are preferred due to the reduced number of updates and label requests. Active approaches also provide additional information about the drift points in the stream, helping understand the process responsible for data generation. However, most drift detectors from the literature require labeled data to monitor performance indicators and verify when the model is outdated. Obtaining fully labeled data in an online fashion has a high cost for being dependent on experts to analyze fast arrival data. Since supervised detectors are limited or even impractical in some scenarios, researchers have sought unsupervised alternatives to perform such a task [40].

Most unsupervised concept drift detectors consider statistical tests that search for significant differences between the current data distribution and historical data [21,50]. Given the high computational complexity of multivariate tests like Hellinger distance [10] for density estimation, a common approach is to perform univariate tests for each data feature independently [49]. Examples of this approach are the seminal work of Kifer et al. [23] and recent works such as the unsupervised ensemble drift detection with feature subspaces [24]. Although it is a practical approach for drift detection, performing individual tests for each feature can be computationally costly for high-dimensional problems such as in the computer vision domain or text mining.

We present an unsupervised detector suitable for high-dimensional and high-speed data streams named *image-based drift detector* (IBDD). IBDD uses a simple two-dimensional visual data representation that allows detecting drifts efficiently based on pixel differences. This simplicity represents an essential speed advantage over the traditional methods when applied to high-speed data streams with thousands of features. We experimentally show on several domains that IBDD can accurately detect different types of drifts with time efficiency. We must emphasize that IBDD was first introduced in Souza et al. [38]. This paper gives more technical details on its implementation, properties, and performance, including new experiments considering variations in the visual representation, distance measures, evaluation

with synthetic data, and two new case studies of real-world problems regarding smart meters and asphalt condition monitoring.

The highlights of our proposal are summarized as follows:

- IBDD is simple to implement and fast to process high-dimensional streams at tens of observations per second, as required by many real-world applications;
- IBDD does not depend on labeled data to detect concept drifts;
- IBDD is adaptive to deal with various types of drifts;
- IBDD is model-independent. The detection is based exclusively on data, without the influence of the classifier, their outputs or performance. This characteristic allows employing IBDD in supervised or unsupervised mining tasks, as well as for analysis of evolving data;

This paper is organized as follows. In Sect. 2, we present the main concepts and definitions related to data streams and concept drift. Related works are discussed in Sect. 3. The proposed drift detector is introduced in Sect. 4. The experimental evaluation on synthetic and real data is reported in Sect. 5, while five different case studies are discussed in Sect. 6. Finally, our conclusions are presented in Sect. 7.

2 Background and definitions

A data stream is defined as a potentially infinite sequence of examples that arrive ordered by time, i.e., $\mathcal{DS} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_t, \mathbf{X}_\infty\}$, where $\mathbf{X}_t$ is a d-dimensional vector in the feature space that was observed at time t. In practice, $\mathbf{X}_t$ is a list with d descriptive features or attributes that represents the example as follows, $\mathbf{X}_t = \{a_1, a_2, \ldots, a_d\}$, where a_i can be qualitative (nominal, ordinal, or binary) or quantitative (discrete or continuous). When d is around hundreds or thousands of features, we have a high-dimensional data stream [48].

Among tasks such as clustering, regression, and anomaly detection, classification is probably the most prominent one. In classification, each example $\mathbf{X}_t \in X$ is associated with a class label $y_t \in Y$, generating a sequence of ordered pairs $\mathcal{DS} = \{(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \ldots, (\mathbf{X}_t, y_t), (\mathbf{X}_\infty, y_\infty)\}$. A model $\mathcal{L}_i$ induced from a historical data $\mathcal{DS}_i = \{(\mathbf{X}_1, y_1), \ldots, (\mathbf{X}_i, y_i)\}$ wants to predict the class label y_t of the example $\mathbf{X}_t$ observed in a future time t, where $t > i$.

Many real-world problems are dynamic with data generated under 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 [15]. 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 concept drifts into two groups according to the type of change [15]: (1) real and (2) virtual. 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.

Knowing the possible type of change that a given problem is subject to and the availability of labeled data is essential for choosing the appropriate detector. For example, in problems where it is impossible to continually monitor the predictive performance due to the high costs of obtaining labeled data, unsupervised detectors are required. However, unsupervised drift detectors can only detect changes when they are observable in the input features $P(X)$,

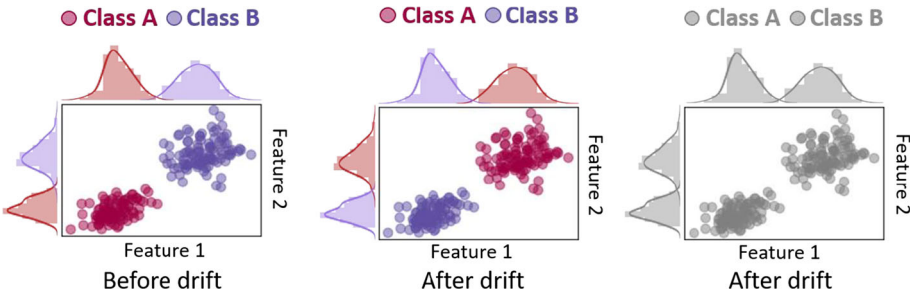


Fig. 1 Example of undetectable concept drift (real drift) by unsupervised detectors

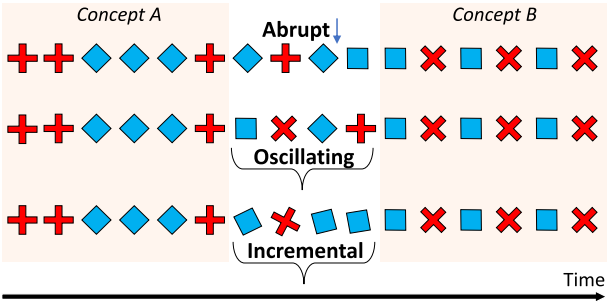


Fig. 2 Example of abrupt, oscillating, and incremental drifts in a two class problem. Initially (Concept A), the classes are defined as blue diamonds ◆ and red plus signs ✚. After the change (Concept B), the classes are then defined as blue squares ■ and red multiplier signs ✖ (color figure online)

since $P(Y|X)$ are not observable without labeled data [41,50]. Figure 1 illustrates a real concept drift in which a supervised detector can detect changes by monitoring the classifier's performance or by a simple comparison of the density estimates deviation for each class at different times. In contrast, this change is undetectable by unsupervised detectors that do not have access to labeled data. If the changes occur solely in the features $P(X)$ without affecting the decision boundary $P(Y|X)$ (i.e., virtual drift), a supervised detector based on performance monitoring will not identify it.

Concept drifts also can be categorized according to the change velocity/severity: (1) abrupt, (2) oscillating (or gradual), and (3) incremental. Let us consider Fig. 2, in which 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 ✖. 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 concept B. Incremental drifts occur over a time window with intermediate concepts between concepts A and B.

3 Related work

Concept drift detection and change-point detection are closely related tasks. While change-point detection has extensively been addressed in statistics since the seventies [19], concept drift detection has become the focus of intensive research by the data mining community in

the last decades [15,23]. The primary goal of change-point detection is to identify when an abrupt change occurs in the patterns of a sequence of real values representing the measurement of given phenomena (i.e., time series data). In turn, concept drift detection seeks to identify changes that affect the mapping from the input space (features) to the target concept (class) in data mining problems. Also, it is not restricted to time series and abrupt changes.

With the increasing popularity of streaming applications, concept drift detection has been a hot topic, with new detectors proposed every year by the data mining community. However, most of them, such as the state-of-the-art drift detection method (DDM) [14] and adaptive windowing (ADWIN) [3], are supervised methods that require that the ground-truth labels be available almost immediately after every prediction for performance monitoring. These methods follow the basic procedure of online adaptive learning [15]: (1) predict, (2) diagnose, and (3) update. The assumption of label availability after predictions required by the diagnose step is very optimistic and not realistic in many situations, restricting its use on real problems [40,44]. Given the high costs for obtaining labels in the stream setting due to the massive amount of data, the fast arrival time, and the dependence of experts, it is desirable to consider unsupervised detectors.

Unsupervised drift detectors focus on finding changes in data distribution without label information. These methods are usually based on a windowing scheme and statistical hypothesis tests performed on two data groups: a reference window with data drawn from the past and another window with the most recent ones. 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 data [23].

Some standard statistical tests for drift detection are the following univariate tests: Kolmogorov–Smirnov, two-sample T-test, Wilcoxon rank sum, Wilcoxon signed-rank, and Wald–Wolfowitz. Sobolewski and Woźniak [35] present a review of these tests, including a discussion of their sensitivity to drift detection. For the batch scenario, Cieslak and Chawla [7] present an interesting study comparing the Kruskal–Wallis, Kolmogorov–Smirnov, and Hellinger distance to identify distributional divergence in the feature space. Afterward, Ditzler and Polikar [10] proposed the use of Hellinger distance to detect drifts in data streams.

Žliobaitė [50] discusses three different information that can be used by statistical tests for unsupervised drift detection: (1) raw data, (2) classifier output, and (3) estimated class labels. Most of the works from the literature assume at least one of these approaches. In the following, we describe how each of them works and highlight their main advantages and drawbacks.

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 [15,23,31,35]. The drift is detected if one or a subset of features show statistical differences. This approach advantage is that it works independently of a classifier but can be costly on high-dimensional data. In this direction, Reis et al. [31] 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 multidimensional case, IKS shows a time complexity of $O(d\log N)$ for each window comparison. Besides the

computational costs, such an approach is sensitive to false alarms since a single feature can indicate a drift.

Recently, Korycki and Krawczyk [24] have proposed the unsupervised ensemble drift detection with feature subspaces (EDFS). EDFS combines the IKS and feature subspaces using an ensemble scheme. Instead of signalizing a drift whenever a single feature changes, EDFS creates a set of features subsets and performs the IKS test for each subset. A drift is then detected only if most tests detect drift in a subset using a voting scheme. The motivation of EDFS is to increase the robustness to false alarms and noisy features since the drifts are detected only when the changes occur on a substantial subset of features.

The second approach considers the classifier's output, such as the score or the probability of the examples to belong a class. The dependence of a classifier is a drawback of this approach since a wrong choice can drastically change the results. Besides, predicting well-calibrated probabilities is still a challenge for many supervised learning algorithms [30]. In some cases, the information provided by a classifier is used to detect drifts without statistical tests. An example is the margin density drift detection (MD3) algorithm [34], which monitors drifts verifying the number of examples in uncertainty regions. As uncertainty regions, MD3 considers the margin bounds produced by supervised models like SVM in the training phase.

In the latter approach, the statistical test considers the class labels predicted by a classifier. In this case, it is only possible to detect changes in class proportions $P(Y)$ over time, which typically has less impact on the accuracy. An efficient approach is to adjust only the decision thresholds instead update with new data [27].

4 IBDD: Image-based drift detector

IBBD is a model-independent and unsupervised drift detector for data streams that uses a visual representation to monitor distribution changes 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, textual data, and computer vision problems, which typically have thousands of features and need fast responses.

Our detector works independently of a classifier, which means that it can be used alone in tasks such as data analysis/online monitoring to provide warnings about significant changes in data, or it can be a trigger mechanism for stream classifiers to indicate the best time for the model update.

Figure 3 illustrates the general framework of IBDD in three main steps. In the first step (a), a reference image is generated from an initial data subset to compare the future data distributions. In this step, it is also estimated the thresholds for detection. In the second step (b), we generate a new image whenever an example arrives from the stream, considering a sliding window approach. In the last step (c), the method compares both images and detects a drift when the similarity between them is different from an expected range of values, previously estimated. We provide a detailed discussion regarding each step in what follows.

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

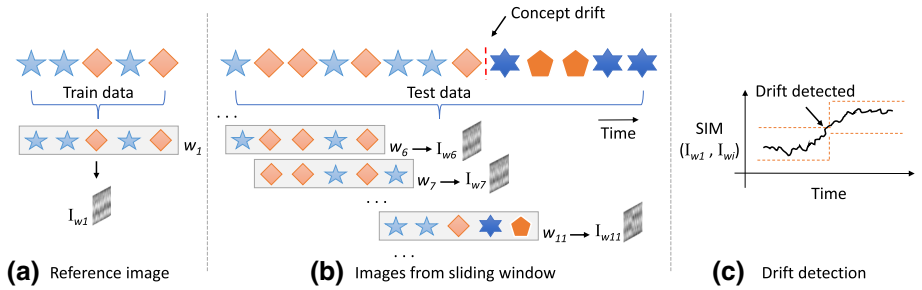


Fig. 3 General framework of the image-based drift detector (IBDD) in three main steps

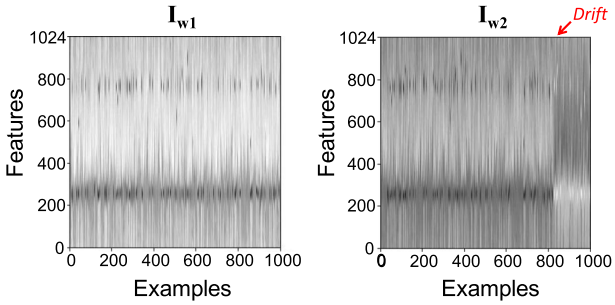


Fig. 4 Example of the visual representation used by IBDD given two data distributions drawn from different periods on the StarLightCurves dataset

reference window $w1$ is never updated, even after a drift is detected. Instead, we choose to update the thresholds that indicate a drift according to the data divergence.

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 visually represent a sequence of d-dimensional examples 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 illustrate the visual representation used by IBDD, let us consider the images I_{w1} and I_{w2} shown in Fig. 4. These images represent two data distributions drawn from different periods of one of our datasets (StarLightCurves). Each window has 1000 examples with 1024 features. On the right side of the figure, we indicate the start drift point in I_{w2}, which is visible only after the arrival of a sequence of examples from a different concept. 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 converting the windows $w1$ and $w2$ into the images I_{w1} and I_{w2}, we can measure the similarity between them to find significant changes. As $w2$ is a sliding window updated with each new arrival example from the stream, the images comparison are 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 dimensions with $p \times q$ pixels, according to the image's height and

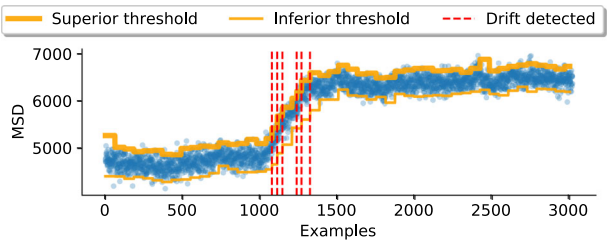


Fig. 5 Drifts detected by IBDD and updated thresholds (color figure online)

width, respectively.

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

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

According to the similarity value returned by MSD, we flag a drift when a sequence with m values is 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 confidence level to reject the null hypothesis that the two data groups are from the same distribution, the value returned by MSD does not have the same meaning, and its 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.

As previously stated, we estimate the initial values for the superior and inferior thresholds using the input data assuming they 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, set as 20 in all experiments.

In stable periods, the superior and inferior thresholds are updated dynamically 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.

The thresholds also update every 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. Figure 5 illustrates the detection of six drifts and the dynamic thresholds over the stream. Each blue circle represents the similarity between the windows measured by MSD. The example concerns the Yoga dataset, which will be introduced in Sect. 5.4.

Algorithm 1 presents the pseudo-code of IBDD coupled with a stream classifier. For its use only as a drift detector, we can ignore the input of a set of training labels (y_{train}) as well as the steps for model build (line 9), label prediction (line 11), labels request (lines 27 and 36), and model update (lines 28 and 38).

Algorithm 1 Image-Based Drift Detector and Stream Classifier

Require: Set of training examples ($\mathcal{X}_{train}$), set of training labels (y_{train}), unlabeled data stream ($\mathcal{X}_{test}$), window size (ws), number of consecutive MSD values above/below the threshold (m)
Ensure: Indexes of detected drifts ($\mathcal{CD}$), label predictions ($\hat{y}$)

1: $lastUpd \leftarrow 0$ ▷ Index position of last threshold update
2: $supThrd, infThrd \leftarrow initialEstimate(\mathcal{X}_{train}, ws)$ ▷ Thresholds estimate by permutation
3: $difThrd \leftarrow (supThrd - infThrd)$ ▷ Difference between thresholds
4: $w1 \leftarrow \mathcal{X}_{train}[end - ws : end]$ ▷ Gets the last ws training data
5: $w2 \leftarrow w1$
6: $I_{w1} \leftarrow imshow(w1)$ ▷ Generate an image from data $w1$
7: $\theta \leftarrow$ modelFit($\mathcal{X}_{train}, y_{train}$) ▷ Build an initial model
8: **for each** $\mathbf{x}_i \in \mathcal{X}_{test}$ **do** ▷ Starts to process the stream
9: $\hat{y} \leftarrow \hat{y} \cup \theta(\mathbf{x}_i)$ ▷ Predicts a label for $\mathbf{x}_i$
10: $w2 \leftarrow w2 \setminus w2[0]$ ▷ Drop the first element of the set
11: $w2 \leftarrow w2 \cup \mathbf{x}_i$ ▷ Append the current example $\mathbf{x}_i$
12: $I_{w2} \leftarrow imshow(w2)$ ▷ Generate an image from data $w2$
13: $sim \leftarrow sim \cup MSD(I_{w1}, I_{w2})$
14: **if** $(i - lastUpd > 50)$ **then** ▷ Update thresholds on stable periods
15: $supThrd \leftarrow \mu(sim[lastUpd : end]) + 2 \times \sigma(sim[lastUpd : end])$
16: $infThrd \leftarrow \mu(sim[lastUpd : end]) - 2 \times \sigma(sim[lastUpd : end])$
17: $difThrd \leftarrow difThrd \cup (supThrd - infThrd)$
18: $lastUpd \leftarrow i$
19: **end if**
20: **if** $(all(sim[end - m : end]) \geq supThrd)$ **then** ▷ Values above the superior threshold
21: $supThrd \leftarrow sim[end] + \sigma(sim[lastUpd : end])$ ▷ Update superior threshold
22: $infThrd \leftarrow sim[end] - \mu(difThrd)$ ▷ Update inferior threshold
23: $difThrd \leftarrow difThrd \cup (supThrd - infThrd)$ ▷ Update thresholds difference
24: $lastUpd \leftarrow i$
25: $y_{req} \leftarrow requestLabels(w2)$ ▷ Request labeled data
26: $\theta \leftarrow$ modelFit($w2, y_{req}$) ▷ Model update
27: $\mathcal{CD} \leftarrow \mathcal{CD} \cup i$ ▷ Update the list of drift positions with the index i
28: **else**
29: **if** $(all(sim[end - m : end]) \leq infThrd)$ **then** ▷ Values below the inferior threshold
30: $infThrd \leftarrow sim[end] - \sigma(sim[lastUpd : end])$ ▷ Update inferior threshold
31: $supThrd \leftarrow sim[end] + \mu(difThrd)$ ▷ Update superior threshold
32: $difThrd \leftarrow difThrd \cup (supThrd - infThrd)$ ▷ Update thresholds difference
33: $lastUpd \leftarrow i$
34: $y_{req} \leftarrow requestLabels(w2)$ ▷ Request labeled data
35: $\theta \leftarrow$ modelFit($w2, y_{req}$) ▷ Model update
36: $\mathcal{CD} \leftarrow \mathcal{CD} \cup i$ ▷ Update the list of drift positions with the index i
37: **end if**
38: **end if**
39: **end for**
40: **return** $\hat{y}, \mathcal{CD}$ ▷ Return predicted labels and drift positions

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 N stream example and have linear complexity according to the number of examples in the windows and the number of features, i.e., $O(p \times q)$, 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 in the sliding window $w2$. The storage of these examples is also essential to retrain the model after the drift detection.

5 Experimental evaluation

To support the reproducibility of our results, we have created a website[1] where we made available the codes and data evaluated in this work.

5.1 Setup

Ideally, the evaluation of a drift detector considers the exact change positions in the stream with measures such as the rate of missed detections, mean time to detections, and mean time between false alarms. However, given the lack of ground truth regarding the drifts' location in real-world data, it is usual to adopt the accuracy of a classifier updated according to a drift detector to measure its performance indirectly. Even in cases where we included changes artificially, we cannot guarantee that the dataset does not have unknown changes in other positions. Besides, for certain types of changes (e.g., incremental and oscillating), the changes occur over a time frame, and it is not easy to define the exact point they start or end to measure the delay. For these reasons, we chose to evaluate the detectors based on the classifier's results. 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 or with a small delay.

Unfortunately, the simplicity of this evaluation also leads to limitations. A drawback is that it does not consider the costs of unnecessary updates triggered by false detections. For this reason, we also consider the total time to process the entire stream. Thus, a classifier that performed a higher number of unnecessary updates is penalized with more time.

We compare the performance of a classifier updated with IBDD against two nonparametric statistical tests which use only the features information for unsupervised drift detection: (1) incremental Kolmogorov–Smirnov (IKS)[2] [31] and (2) Wilcoxon rank-sum (WRS) [35]. In all experiments, we adopt the same significance level employed by Reis et al. [31] ($\alpha = 0.001$). Two main reasons justify the choice of the statistical tests to compare our proposal: (1) given the simplicity and effectivity of this approach, it is a well-known and common choice for drift detection (e.g., [23,31,35,50]); and (2) for a fair comparison, this approach is the most similar with ours, being unsupervised and model-independent detectors.

Besides, we also take into account 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 example from the stream leading to continuous model updates. This approach is the closest strategy for maintaining a classifier updated with the most recent data over all the time, and the most costly.

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

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

[2] Python implementation provided by the authors.

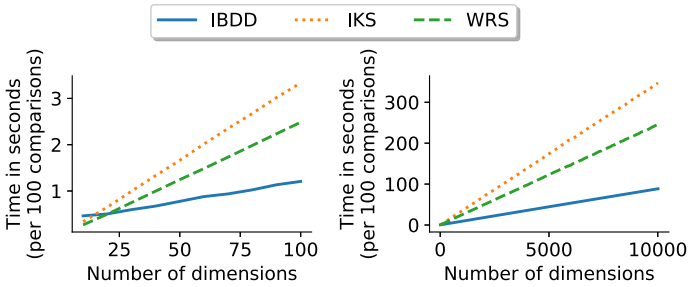


Fig. 6 Time spent in seconds by the detectors to perform 100 comparisons of windows with 1000 examples and dimensionality ranging from 10 to 10,000

5.2 Time costs

To evaluate the time efficiency of IBDD on high-dimensional data without the classifier's influence and the time for model updates, we measure the costs spent by different detectors for window comparisons varying the number of features in a data generated synthetically. Figure 6 shows the time results in seconds to perform 100 comparisons of windows with 1000 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 until 10,000 features.

IBDD becomes time competitive starting from 25 features. However, we recommend our detector on high-dimensional stream problems with more than 100 features. In such cases, we note a considerable time difference between the methods. In Sect. 5.4, we will discuss the time of IBDD when coupled with classifiers for model adaptation on real-world high-dimensionality data with concept drifts.

5.3 Results from synthetic datasets

Although the discussions throughout this article are mainly based on evaluations using real-world data, in some situations, simulated data help to verify specific questions such as the detector performance under different types of drifts. We consider synthetic datasets with controlled types of change. It is important to clarify that IBDD does not identify the type of drift a dataset experiences, but its detection works on data showing changes of different types. In the following, we describe the synthetic datasets.

Waveform [12]. In this dataset, the classification task is to differentiate between three different waveforms classes, each one generated from a combination of two or three base waves. The examples are described by 21 numeric features, all of which include noise. MOA framework [4] was used to generate versions with abrupt and oscillating changes (Waveform-A, Waveform-O, respectively). For both cases, we consider the occurrence of drifts at around example 500 in the test data;

Four class rotating (4CR) [44]. In this dataset, four bidimensional Gaussian clusters (each cluster is associated with a class) incrementally rotate clockwise around a common axis over time until performing a complete cycle of 360°. Figure 7 illustrates this dataset in four different moments, from T1 to T4.

Unimodal Gaussian two classes five dimensions (UG2C5D) [13]. In this dataset, two classes represented by Gaussian clusters with five features incrementally move in the space

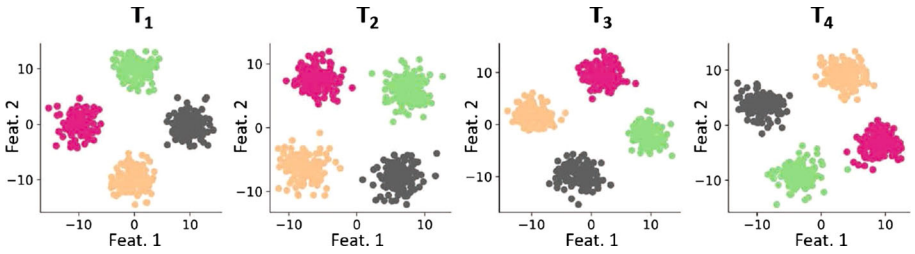


Fig. 7 Four different moments of the 4CR dataset with incremental changes

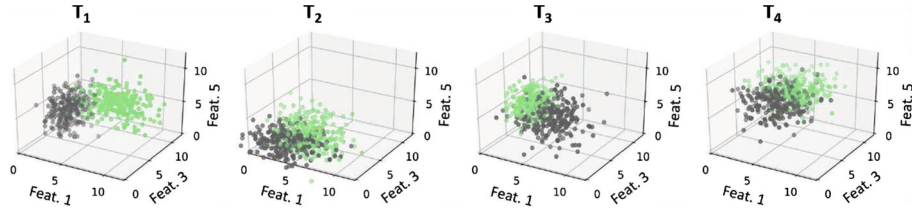


Fig. 8 Four different moments of the UG2C5D dataset with incremental changes

Table 1 Description of the synthetic datasets

Dataset	#Classes	#Examples	#Features	Train	Test	Drift
Waveform-A	3	5000	21	500	4500	A
Waveform-O	3	5000	21	500	4500	O
4CR	4	144,400	2	400	144,000	I
UG2C5D	2	200,000	5	1000	199,000	I

The drift types are *A* abrupt, *I* incremental, and *O* oscillating

in arbitrary directions. Figure 8 illustrates this dataset in four different moments (from T1 to T4), given three of five features. Unlike waveform, whose drifts start at a specific time, in UG2C5D, as well in 4CR, the incremental drifts are continuous over all the stream.

Table 1 summarizes the main characteristics of the synthetic datasets, such as the number of classes, examples, features, and size of the train and test sets.

The classification accuracy achieved by the baselines and by the classifiers using the detectors to trigger the model updates is shown in Table 2. This table also shows the *overall mean* achieved by the classifiers over all datasets. Given the three detectors, we highlighted the best result for each dataset. We note that IBDD provided accurate results for different types of drifts for most cases.

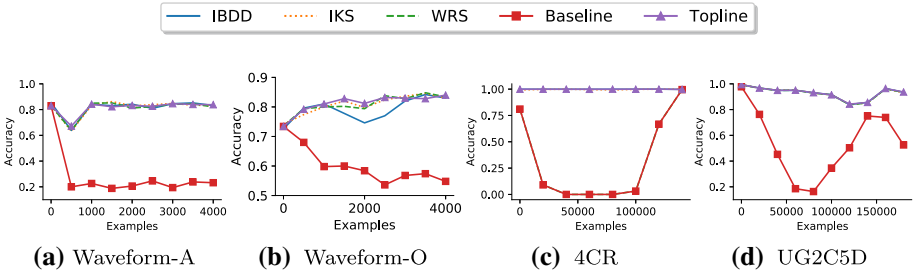
In Fig. 9, we show the accuracies over time. In all cases, we notice the drifts' impact on the Baseline classifier and how the detectors help the classifiers to deal with changes. For Waveform-Oscillating, the detector rapidly identifies the start point of the change but loses some accuracy around the example 2000 due to some false negative detections and, consequently, unnecessary updates.

It should be noted that the good result of a detector is directly related to its ability to identify the true changes as soon as possible and to ignore false alarms that noise or outliers examples can trigger. We hypothesize that IBDD is less susceptible to false alarms than traditional statistical tests since changes observed in a single variable hardly trigger our

Table 2 Accuracy results for synthetic datasets

Dataset	IBDD	IKS	WRS	Baseline	Topline
Waveform-A	**81.62**	81.56	81.24	28.44	81.64
Waveform-O	79.09	80.78	**80.82**	60.24	81.18
4CR	**99.89**	99.63	25.00	25.00	99.91
UG2C5D	**92.98**	92.90	92.87	54.06	93.06
Overall mean	88.40	88.72	69.98	41.94	88.95

The best result between the three drift detectors is highlighted in bold for each dataset



(a) Waveform-A **(b)** Waveform-O **(c)** 4CR **(d)** UG2C5D

Fig. 9 Accuracy over time for synthetic datasets with different types of drifts

detector. Besides, our parameter m provides some robustness in scenarios with noise and outliers examples that do not represent concept drifts. Furthermore, the image normalization (as previously illustrated in Fig. 4) also can be a factor that helps in the early detection since it makes the differences between the data distributions more evident.

5.4 Results from real-world datasets

We evaluated six real-world datasets from different domains, including healthcare, entomology, and astronomy. In some cases, we have modified the original data to include concept drifts. However, in all cases, the changes can be justified in practice. We describe the datasets in what follows.

Heartbeats [16,46]. This dataset contains ECG data from the Sudden Cardiac Death Holter Database (SDDB) of Physionet [16]. We consider the version previously evaluated by Ulanova et al. [46] for stream clustering, which uses the data from a specific patient (subject #51), a 67-year-old female. The dataset contains individual heartbeats extracted about once every 0.86 seconds of a stream 25 h long. Each example 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 [46]. The dataset's task is to identify the insects' species based on their wing-beat information measured by an optical sensor. When flying insects cross a light window in this sensor, their wings and body partially occlude the light for a very brief time, causing small variations captured by a set of phototransistors and recorded as an audio signal. By converting the sound to the spectrum of frequencies, it is possible to recognize different species based on features such as fundamental frequency and harmonics. In Fig. 10, we illustrate the audio signal generated by crossing an *Aedes aegypti* ♂ mosquito through the sensor and the spectrum of frequencies. In this example, the observed insect's wing-beat

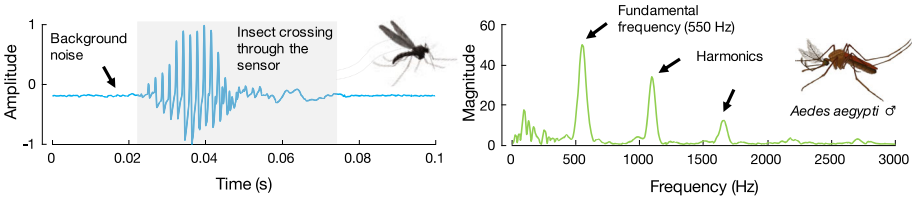
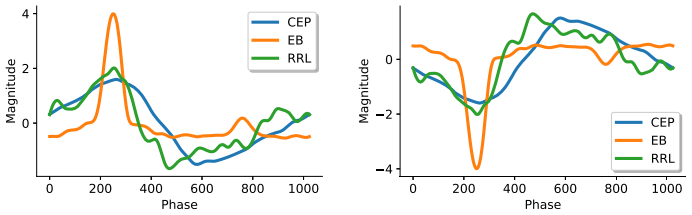


Fig. 10 (*left*) Audio signal generated by the optical sensor and (*right*) spectrum of frequencies



(a) Before concept drift **(b)** After concept drift

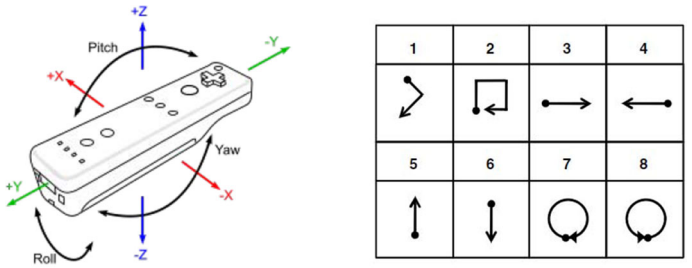
Fig. 11 Examples from the StarLightCurves dataset before and after the induced concept drift. Each curve represents an example from the three classes: Classical Type-I Cepheids (CEP), Eclipsing Binaries (EB), and RRab and RRc RRLyrae (RRL)

frequency is 550 Hz. The dataset has three classes: *Aedes aegypti* ♂, *Aedes aegypti* ♀, and *Culex quinquefasciatus* ♀. It is expected drifts on these data due to variations in environmental conditions (temperature and humidity), responsible for changing the flight characteristics of the insects and the measured data [43];

Sensor posture [22]. These data are from a multiagent system for the care of older people. The system monitors physical activities and reacts to critical situations calling for help in emergencies and issuing warnings if unusual behavior was detected. Each example contains the spatial localization of three wearing tags installed in the ankle left, ankle right, and chest of a person. The original data have 164,860 readings obtained from five persons and do not have drifts. In this work, we consider the modified version proposed by Reis et al. [31], where the ordering of the examples has sequences produced by the same person to induce drifts;

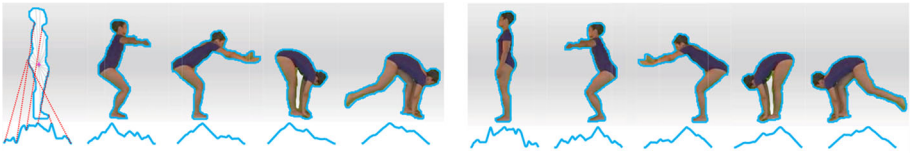
StarLightCurves [8]. This is a time series dataset where each example has 1024 features representing light measurements achieved from telescopic images regarding a celestial object as a function of time. The study of light curves in astronomy is associated with the study of the variability of sources. The dataset has predefined training and test sets with 1000 and 8236 examples, respectively. We introduce drifts in the test set by multiplying the values of each instance by -1, starting at the example 2000. We named this modified version StarLightCurves-YReversed. In practice, this change can be justified by a malfunction in the sensor responsible for the measurements. In Fig. 11, we show a representative example for each class, before and after the concept drift;

UWaveGestureLibraryY [8,26]. This dataset contains different gestures measured by an accelerometer device (Wii remote) on the y-axis. The dataset has predefined training and test sets with 896 and 3582 examples, respectively. We introduce incremental drifts in the test set, changing the sensor orientation around the x-axis six times in an angle of 30° each change until complete a move of 180°. We named this modified version as UWaveGestureLibY-Incremental. This kind of drift is justified in practice since motion devices such as a smartwatch on the user's wrist can slowly change their position over time. In Fig. 12a,



(a) Wii remote **(b)** Gesture movements

Fig. 12 Device used to measure the gesture movements of UWaveGestureLibraryY (**a**). The simulated drifts affect the device's pitch changing its inclination. The eight gestures performed in the dataset (**b**), where dot denotes the start and the arrow the end of each movement [26]



(a) Original images and their series **(b)** Flipped images and their series

Fig. 13 Image examples from the 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

we show the device used to measure the gesture data and its axes. The simulated drifts are responsible for changing the device's pitch. In Fig. 12b, we illustrate the eight classes of the dataset, where the dot denotes the start and the arrow the end of the gestures;

Yoga [8]. 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 predefined training and test sets with 300 and 3000 examples, respectively. We introduce drifts in the test set by flipping the *x*-axis of the series, starting on the example 1000. We named this modified version Yoga-XReversed. In practice, this drift represents a change in the orientation of the actors performing the yoga movements. To better illustrate, we show five examples in Fig. 13. In this figure, it is possible to see the image conversion to a time series and the effect on data when the actor changes their side.

In Table 3, we present the main characteristics of the datasets, where we can note that only Posture does not have a high number of features. We have included it to evaluate the performance of our proposal with low dimensional data. This table also indicates the size of the initial training data to build the first classification model. To standardize our experiments, we consider the same number of examples for the detectors' window size parameter.

In Table 4, we show the average accuracies over the data streams. Given the three detectors, we highlighted the best result for each dataset. Specifically for the Yoga dataset, IBDD slightly outperformed the Topline detector with a drastically lower number of model updates (6 times vs. 3000 times) and data labels requested (18.2% vs. 100%). Such points of updates can be seen in Fig. 5.

Table 3 Description of the real-world datasets

ID-Abbrev.	Dataset	#C	#E	#F	Train	Test	Drift
1-HBeats	SDDB heartbeats (#51)	8	77,904	280	500	77,404	O
2-Insects	Flying insects	3	86,400	200	1000	85,400	A
3-Posture	Person activity	11	164,860	3	2000	162,860	I
4-SLCur	StarLightCurves-YRev	3	9236	1024	1000	8236	A
5-UWave	UWaveGestLibY-Inc	8	4478	315	896	3582	I
6-Yoga	Yoga-XRev	2	3300	426	300	3000	A

The three drift types are *A* abrupt, *I* incremental, and *O* oscillating
C is the number of classes, *F* is the number of features, and *E* is the number of examples

Table 4 Accuracy results for real-world benchmark 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
Overall mean	79.30	77.51	77.59	50.49	79.13

The best result between the three drift detectors is highlighted in bold for each dataset

In Fig. 14, we present the accuracy over time for the six real-world benchmark datasets. Analyzing the accuracy decrease of the Baseline classifier, we can identify the point where the drift starts for each dataset and how the detections at different points impact the classifiers' recovery.

To better illustrate the time efficiency of IBDD on real data, Table 5 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 the Posture dataset, which has three features. It is interesting to note that the Baseline time represents only the classification costs. In contrast, the Topline times represent the classification times and the costs to model update for every example. Thus, for some datasets such as Heartbeats, StarLightCurves, and Yoga, it is more time expensive to use the IKS or WRS detectors than the naive approach employed by the Topline. The main practical restriction of such an approach is requesting the labels of all examples.

The relation between time and error is outlined in Fig. 15a. In this image, each point and ID represent a dataset, and different markers distinguish the detectors. The best results are located close to the origin (0, 0). As we can note, IBDD shows the best relation between time and accuracy for most of the datasets.

To statistically compare the accuracies provided by the detectors considering all the ten datasets (synthetic and real), we performed the Friedman test with a significance level of 5% (p value < 0.05) and the Nemenyi post hoc test [9]. This nonparametric test is indicated to compare multiple algorithms over multiple datasets. In our case, we compare the same algorithm using different detectors to decide the best time for updates, considering different datasets. We visually represent the results using a critical difference diagram. In this diagram, we sort the methods according to their average ranking outputted by the statistical test. The

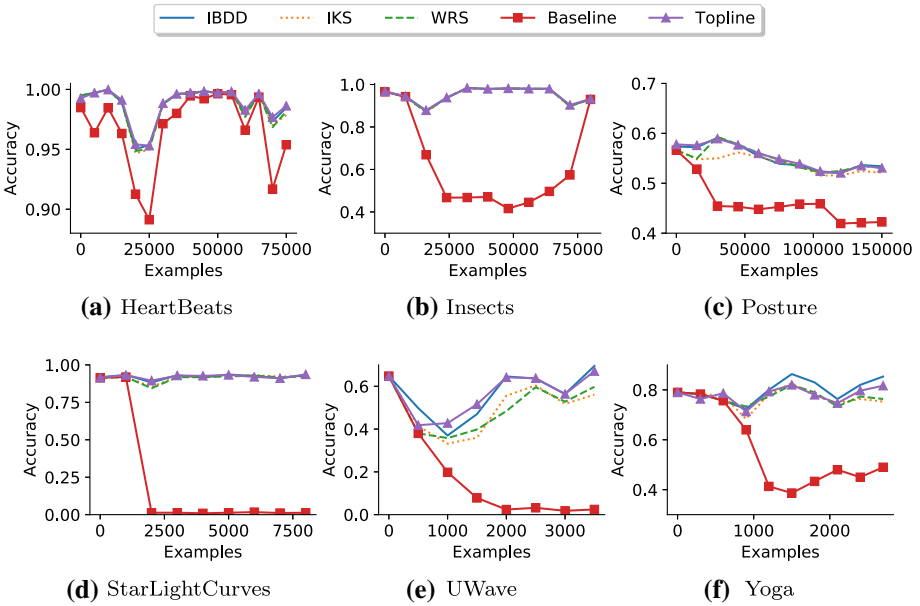


Fig. 14 Accuracy over time for real-world benchmark datasets

Table 5 Total time and time per example (in seconds) to process real-world datasets

Dataset	IBDD	IKS	WRS	Baseline	Topline
1-HBeats	**1650/0.021**	28,917/0.375	7703/0.100	349/0.004	6887/0.089
2-Insects	**2255/0.026**	18,327/0.215	7212/0.084	372/0.004	22,158/0.259
3-Posture	1633/0.010	**976/0.006**	1114/0.007	709/0.004	17,438/0.107
4-SLCur	**728/0.088**	7735/0.939	3345/0.406	36/0.004	3679/0.447
5-UWave	**96/0.027**	1125/0.314	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 best result between the three drift detectors is highlighted in bold for each dataset

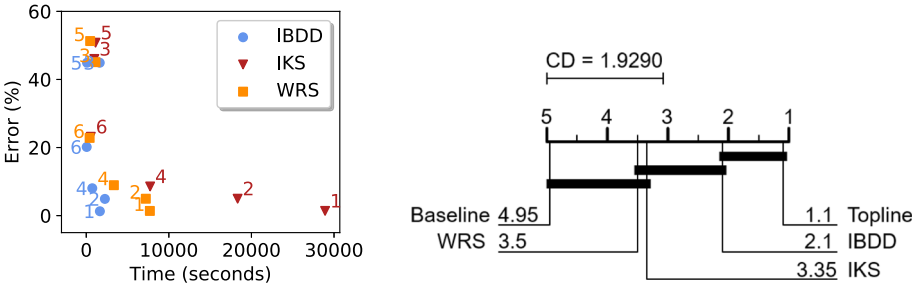


Fig. 15 **a** Time versus error results. Each point is a dataset, and the best results are located close to the origin (0, 0). **b** Critical difference diagram achieved by the Friedman test to statistically compare the detectors

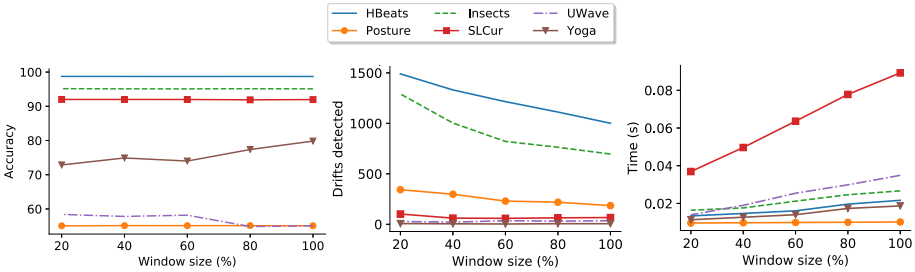


Fig. 16 Accuracy results (left), number of drifts detected (middle) and average time (right) of IBDD to process each example varying the window size parameter (ws)

methods connected by a line do not present statistically significant differences among them. Figure 15b exhibits the diagram. We can note IBDD with an average rank of 2.1 and without significant statistical difference regarding the Topline detector, which outputs a detection for every new example and maintains a classifier updated over all the stream.

5.5 Parameter sensitivity

IBDD has two parameters: (1) the number of examples in the windows and (2) 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 $m = 3$ for all datasets. To show the impact of such parameters, we perform an experiment varying their values.

For the window size parameter, we vary the values from 20 to 100% of the training set size, according to the *Train* column previously reported in Table 3, and set $m = 3$. In Fig. 16-(left), we show the accuracy results varying the window size. The most impacted datasets were Yoga and UWave, in which the accuracy has 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, it is interesting to observe that the best result was achieved with a small window containing 179 examples (20% of the training set). Although the window size slightly impacts the accuracy for most cases, this parameter affects the number of drifts detected and the runtime. In Fig. 16-(middle), we can see 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, as illustrated in Fig. 16-(right).

Concerning the number of consecutive MSD values above or below the drift threshold, we vary the values of m 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. 17, we show the impact of m on the accuracy and number of drifts detected. In most cases, $m = 3$ can avoid a higher number of detections.

5.6 Visual representation

One of the main steps of IBDD is the data transformation in a 2D representation. Given a set of multidimensional examples, such transformation is responsible for normalizing the data into a window in values between 0 and 255, represented by the image's pixels. In favor of simplicity, we represent the pixels values according to a linear scale of the gray color, where 0

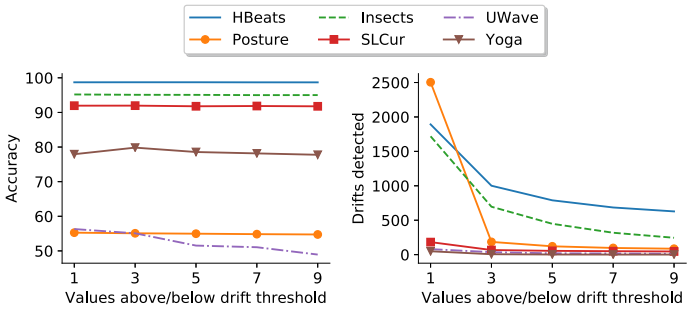


Fig. 17 Accuracy (left) and number of drifts detected (right) by IBDD given the variation of parameter m related to the MSD values above/below the threshold

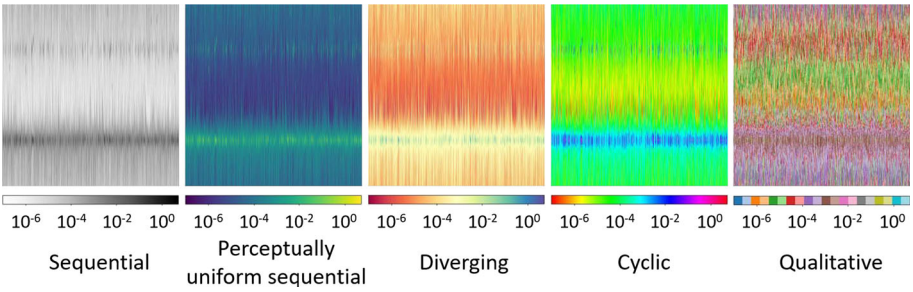


Fig. 18 Images obtained by different color maps given the StarLightCurves dataset

is represented by the black color and 255 by white. However, our visual representation allows employing any color map. A color map is a function responsible for mapping the individual values to specific colors according to a predefined color scale, and that varies according to the visualization purpose [2,29].

To verify the visual representation's influence, we evaluate the IBDD performance using different color maps to generate the images. Specifically, we consider the following categories based on their function [5]: sequential, perceptually uniform sequential, diverging, cyclic, and qualitative. Each category allows the generation of a vastitude of maps given different color combinations. For simplicity, we evaluate a specific color map from each category. Figure 18 illustrates the images obtained from the training set of StarLightCurves dataset. We also display a color bar below the images, representing the map function according to the data values.

Table 6 shows the accuracy results of IBDD employing different color maps. In general, the sequential color map using grayscale was responsible for presenting the best results. The exception was the UWave dataset, in which the best result was achieved by the diverging color map and the worst result by the sequential color map. The lack of a perceptual relationship between the color transitions of the qualitative color map led to a slight decrease in accuracy performance.

5.7 Distance measures

An essential part of our solution is the similarity measurement between the images representing the multidimensional data over different times in the stream. So far, we have shown

Table 6 Accuracy results of IBDD with different visual representations

Dataset	Sequential	Perceptually uniform	Diverging	Cycling	Qualitative
1-HBeats	**98.72**	98.58	98.61	98.57	98.51
2-Insects	**95.09**	94.95	94.95	95.03	95.07
3-Posture	**55.09**	52.48	52.52	52.51	52.48
4-SLCur	**91.96**	91.15	91.05	90.69	90.55
5-UWave	55.08	58.74	**60.30**	59.74	58.63
6-Yoga	**79.83**	78.60	77.87	79.37	77.53

The most accurate color map is highlighted in bold

that a simple approach like 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) [47] and CK-1 [6].

The SSIM is a well-known measure mainly employed to evaluate image quality by modeling image distortion as a combination of three factors: luminance distortion, contrast distortion, and loss of correlation. Formally, given two images x and y to be compared, SSIM considers the similarity $l(x, y)$ of the local patch luminance (brightness values), the similarity $c(x, y)$ of the local patch contrasts, and the similarity $s(x, y)$ of the local patch structures. These local similarities are simply combined according to Eq. 2.

$$\begin{aligned} \text{SSIM}(x, y) &= l(x, y) \times c(x, y) \times s(x, y) \\ &= \left(\frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}\right) \times \left(\frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}\right) \times \left(\frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}\right) \end{aligned} \tag{2}$$

In Eq. 2, μ_x and μ_y are the respective local sample means of the images x and y, and σ_x and σ_y are the respective local sample standard deviations of x and y, and σ_{xy} is the sample cross-correlation of x and y after removing their means. The symbols C_1, C_2, and C_3 are small positive constants that stabilize each term, so that near-zero sample means, variances or correlations do not lead to numerical instability. The similarity value returned by SSIM is bounded between -1 and 1, where $\text{SSIM}(x, y) = 1$ if and only if $x = y$.

CK-1 is a distance measure based on video compression algorithms to estimate the similarity between images. The primary theoretical basis of CK-1 is the concept of the Kolmogorov complexity. CK-1 extends the applicability of compression-based distances to images by using video compression. Given two images x and y, CK-1 is formalized according to Eq. 3.

$$\text{CK-1}(x, y) = \frac{C(x|y) + C(y|x)}{C(x|x) + C(y|y)} - 1 \tag{3}$$

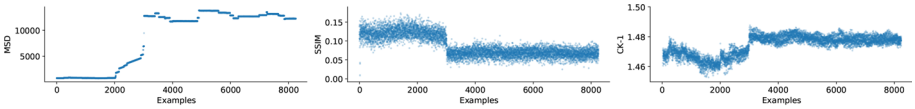
where $C(a|b)$ is the size in bytes of a synthetic MPEG-1 video composed by two frames $b \in \{x, y\}$ and $a \in \{x, y\}$, in this order.

MPEG-1 and most video encoding algorithms achieve compression by finding recurring patterns within a frame (intra-frame compression) and between frames (inter-frame compression). When x and y are similar images, the inter-frame compression step should exploit that to produce a smaller file size, which can be interpreted as a significant similarity. Although the original proposal of CK-1 employs the first generation of MPEG video codec for compression (MPEG-1), we consider a modern encoding algorithm (MPEG-4).

Table 7 Accuracy results 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[a]	**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

[a]SSIM and CK-1 require images with minimum sizes. As Posture has only three features, both measures could not compare the small images generated for this dataset
The most accurate measure is highlighted in bold



(a) MSD **(b)** SSIM **(c)** CK-1

Fig. 19 Similarities obtained by different measures for the StarLightCurves dataset

In Table 7, we show the accuracy results of our comparisons with different distance measures. We can note that MSD achieved the best results among the three distance measures except for the Insects dataset.

In Fig. 19, we compare the different measures for the StarLightCurves dataset. In this dataset, the drift starts at example 2,000. For SSIM and CK-1, we can note a delay of 1,000 examples to observe a significant change in the values, while MSD presents significant changes in their values close to the example 2,000.

For CK-1, we note that an important advantage of this measure for many image mining tasks is image rotation invariance. For example, given two images x and y, the value returned by CK-1(x, y) will be similar to CK-1(x, y'), where y' represents the original image y rotated in 180°. However, this characteristic is a drawback of CK-1 as a similarity distance for IBDD since rotations may indicate possible drifts in the data represented by the images.

6 Case studies

This section presents five case studies on different domains, such as public health, medical science, power management, transportation systems, and political science. These cases represent problems with abrupt, oscillating, and incremental changes in data with thousands of features.

6.1 Abrupt drift detection for streaming classification of Malaria vectors

Malaria is a disease transmitted by infected mosquitoes that kills more than 600,000 people yearly, the majority being African children under 5 years [32]. The identification of mosquito species is a crucial task for the estimation of Malaria transmission dynamics, surveillance, and evaluation of interventions [17]. Identifying species is challenging because some species can

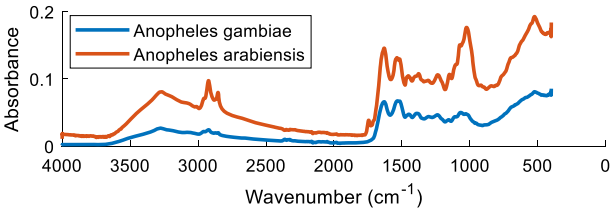


Fig. 20 Mid-infrared absorption spectra of two malaria mosquito species

Table 8 Accuracy and time results 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 (s)	**560**	16,722	4504	17	1611

The best result between the three drift detectors is highlighted in bold for each dataset

only be distinguished by molecular analysis, given their high morphological similarity [33]. 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 large-scale analysis [1].

Recently, Gonzalez-Jimenez et al. [17] have shown 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 (400–4000 cm^{-1}). In Fig. 20, we illustrate the mid-infrared absorption spectra for two examples of malaria mosquitoes from the species *Anopheles gambiae* and *Anopheles arabienses*.

To perform the species' automatic identification, we use a spectrum composed of 3600 observations as input features for supervised machine learning algorithms. Besides being 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 [17]. Thus, this problem requires an adaptive approach to provide accurate predictions under these changes. To benefits from the speed advantage provided by the mid-infrared spectroscopy, the drift detection and model update mechanisms need to be efficient.

Based on the public data provided by Gonzalez-Jimenez et al. [17], we use the meta-information related to the age of the mosquitoes to build a streaming dataset with abrupt changes. These data contain 4260 examples from malaria vector species *Anopheles gambiae* and *Anopheles arabienses* with age varying from 1 to 17 days. For our stream dataset, we adopt 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 age sequence for the arrival of the 3774 remaining examples: (17, 16, 3, 4, 15, 14, 5, 6, 13, 12, 1, 2, 11, 10, 7), where each color transition represents an abrupt concept drift.

A static classifier shows an accuracy of 69.61%, while the use of a drift detector can improve this result by about 12%. Table 8 exhibits detailed results achieved by the evaluated detectors, where IBDD ($m = 1$) presents the best accuracy result, spending about 8 × less time than WRS, and 29× less time than IKS. In Fig. 21, we show the accuracy performance of the detectors over different age groups, where it is possible to see that a classifier trained

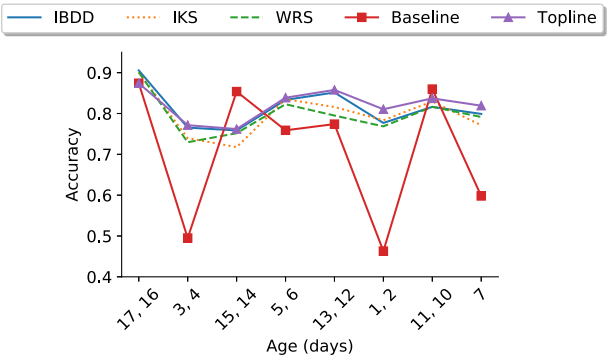


Fig. 21 Accuracy for malaria mosquitoes prediction considering different ages

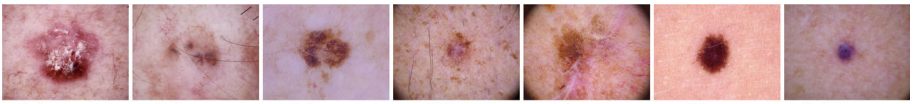


Fig. 22 Examples from the HAM10000 dataset. The diagnoses are: *Akiec actinic keratosis*, *Bcc* basal cell carcinoma, *Bkl* benign keratosis-like lesions, *Df* dermatofibroma, *Mel* melanoma, *Nv* melanocytic nevi, and *Vasc vascular lesions*

on 8- and 9- day-old mosquitoes have difficulty in identifying 1-, 2-, 3-, 4-, and 7-day-old mosquitoes.

6.2 Incremental drift detection for skin lesion classification

Skin cancer is the most common malignancy, affecting men and women of all ages and skin colors [25]. According to the World Health Organization, 2–3 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 [20]. Thus, skin diseases' automatic prediction is an essential and low-cost tool for preliminary diagnoses and early intervention.

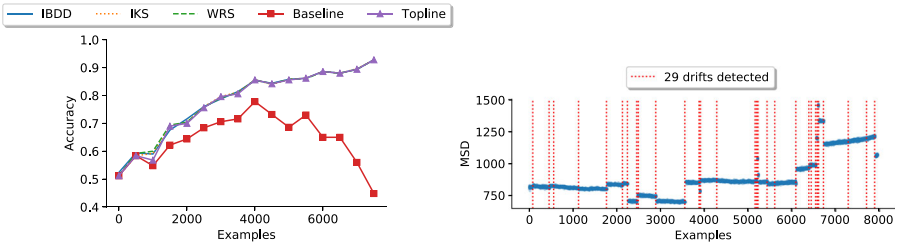
For this case study, we consider 10,015 dermoscopic lesion images from the public dataset HAM10000 [45]. This dataset contains data collected for 20 years. A feature vector with 2353 dimensions representing the mean pixel value of the images' RGB channels describes each example. The dataset has seven classes as illustrated in Fig. 22: actinic keratoses, basal cell carcinoma, benign keratinocyte lesions, dermatofibroma, melanoma, melanocytic nevus, and vascular lesions.

To build a streaming dataset with incremental drifts, we sort the arrival order of the examples according to the patient's age. Aging is directly responsible for changing the skin's characteristics, such as texture and color. Due to the accumulated sun exposure over the years, skin lesions' risk increases with age [28]. 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 of images from patients aged 75–85. The arrival sequence in the test set follows a decreasing trend in the age of 75–0.

Table 9 Accuracy and time results for the skin cancer case study

	IBDD	IKS	WRS	Baseline	Topline
Accuracy	**77.82**	77.74	77.80	64.20	77.49
Time (s)	**2131**	20,984	8199	38	9066

The best result between the three drift detectors is highlighted in bold for each dataset



(a) Accuracy **(b)** MSD values and drifts detected by IBDD

Fig. 23 Accuracy and drifts detected over time for the skin cancer dataset

For this problem, a static classifier shows 64.20% of accuracy, while the use of a drift detector can improve the result by about 13%. Table 9 and Fig. 23a present detailed results, where IBDD ($m = 3$) is the faster approach with the best accuracy. In Fig. 23b, we show the data similarities output by IBDD as well as the position of the 29 drifts detected. All the positions are uniformly distributed over the patients' age, together forming an incremental drift.

6.3 Identifying changes in the power consumption of residential customers

A smart meter records the customer's power consumption in time intervals and transmits the data to the electricity supplier for monitoring and billing. Mining such data allows performing different tasks, including identifying faults and anomalies, generating user profiles, identifying power usage patterns, and providing personalized recommendations for energy saving. There are many reasons to expect concept drifts on smart meter data due to the consumer behavior dynamics. For example, let us consider the consumption of a residential house in the last 5 years. Over this time, the residence could have been rented to different tenants, each with their power consumption patterns. Even for a same tenant, many factors can influence its consumption patterns over the years, such as a job change, divorce, weather conditions, or the simple electricity price update.

In this study, we evaluate the power consumption data recorded by smart meters provided by the Los Alamos Department of Public Utilities (LADPU) in New Mexico, USA. The data of nearly 1700 smart meters were recorded for 6 years. With the data recorded for every 15 min, we have 96 observations per day and a total of 226,272 observations for each smart meter in the period.

To evaluate our method, we build a dataset representing a classification problem to identify ten different residential customers in a single stream. We preprocessed the data to remove outliers and segment the historical time series from different customers into individual examples with 24 h of power consumption, which leads to 96 features per example. The arrival order of the examples in the stream follows the time-stamps, but we randomly shuffle the

Table 10 Accuracy and time results for the customer identification using power data

	IBDD	IKS	WRS	Baseline	Topline
Accuracy	**78.03**	73.43	71.70	31.79	80.05
Time (s)	**319**	1412	840	32	2181

The best result between the three drift detectors is highlighted in bold for each dataset

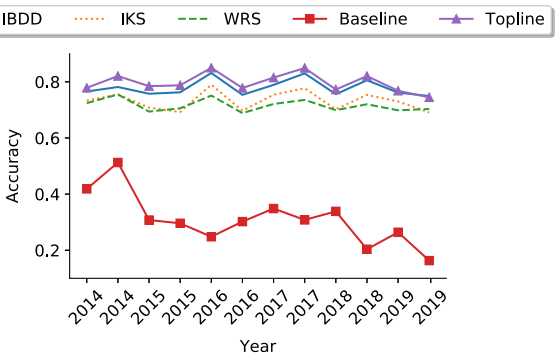


Fig. 24 Accuracy over time for the customer identification by power consumption data

customers to avoid temporal dependence [41]. From a total of 22,950 examples, we train a classifier using the first 300 examples corresponding to the first month of observations given ten customers and test with the 22,650 remaining data. The results of IBDD ($m = 3$) are shown in Table 10.

In Table 10, we note that IBDD improves the accuracy of a Baseline classifier by 46.24%. In Fig. 24, we show the results of different detectors over time, where the performance of IBDD is close to the Topline classifier. However, while the Topline spends about 36 min, IBDD spends only 5 min.

6.4 Monitoring asphalt conditions by vehicle vibration

The evaluation and monitoring of asphalt quality are essential for authorities to plan precise interventions and maintenance in the cities. This task is usually performed by vehicles adapted with complex and costly hardware or by walking profilers manually operated by experts to cover short distances.

In the last years, researchers have been working on new technologies to monitor the asphalt quality by low-cost sensors. An example is the Asfault system [36,37,39], which aims to provide safer and fast routes using the pavement quality information. Asfault collects real-time streaming data using the triaxial accelerometer of a smartphone placed in the vehicle's interior to sense its vibration under different road conditions. The system employs machine learning algorithms to classify the segments of pavement into five classes: (1) good, (2) average, (3) fair, (4) poor, and (5) obstacles. Figure 25 illustrates the classes. As we can see, the obstacles class comprises the occurrence of different events as speed bumps, raised pavement markers, elevated crosswalks, and ditches with greater depth.

Asfault uses two different classifiers to provide accurate responses: one trained in urban streets and other trained on highways. Currently, the user needs to choose it manually. To automate this decision and make Asfault adaptive for different scenarios, we evaluate the use

Fig. 25 Example of the classes considered by the Asfault system

Table 11 Accuracy and time results for the asphalt monitoring case study

	IBDD	IKS	WRS	Baseline	Topline
Accuracy	**87.64**	87.02	87.36	46.64	88.22
Time (s)	141	1426	255	37	1279

The best result between the three drift detectors is highlighted in bold for each dataset

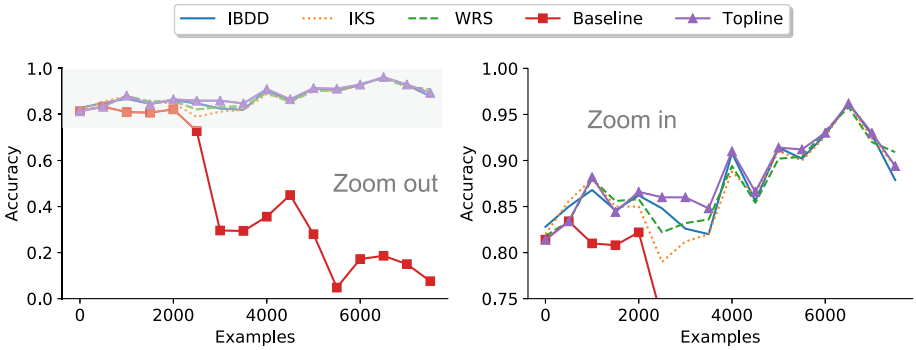


Fig. 26 Accuracies over time for the monitoring of asphalt conditions

of drift detectors to identify the changes between streets and highways, rebuilding the model when a change is observed.

We initially train a classifier with 1000 examples collected in urban streets. In the test phase, we included highways examples starting from example 2878 to 7566. The dataset has 62 features extracted from the accelerometer data, including zero-crossing rate, entropy, kurtosis, skewness, and short-time energy. Table 11 shows the results considering $m = 2$.

We note in Table 11 that IBDD provides the fastest and accurate classifier with 87.64%. Our detector is 10× faster than IKS and about 2 × faster than WRS. Compared with other case studies discussed, the gain in speed is lower due to the lower number of features. In Fig. 26, we show the results over time. For a better comparison, we detail the results in the gray area by reducing the y-axis scale.

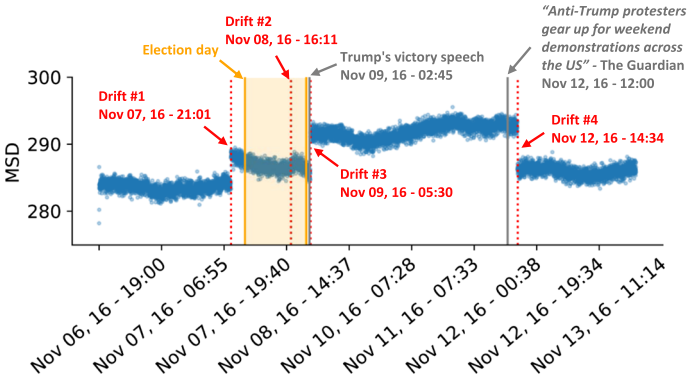


Fig. 27 Detected drifts in the Twitter data

6.5 Oscillating drift detection in Twitter streams

Twitter distributes tweets (i.e., short posts) via a streaming API. One can receive tweets as fast as fifty tweets-per-second. Drifts in 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 (e.g., earthquakes and terrorist attacks) happen.

In this case study, we consider a dataset with 10,841 political tweets related to the 2016 U.S. Presidential Election from November 4, 2016, to November 13, 2016. The election was held on November 8, 2016. The 200 most active users generated the collected data. Each tweet is represented in an embedding space pre-trained on 2B tweets of GLoVE[3] word-embedding, totaling a feature vector with 2000 dimensions and without class labels. To detect drifts in this unsupervised problem, we apply our algorithm with the following parameters: the reference window ($w1$) comprises 2251 tweets generated in the first 48 h, the sliding window ($w2$) has the same size as the reference window, and $m = 4$.

We identify four drift points (see Fig. 27). The drifts mark the evening before the election day, the time when voting was on, the time of the victory speech by President Donald Trump, and the time when thousands of protesters marched to Trump Tower in New York in a rally planned in the social networks.[4] 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 min, IKS spent 6 h and WRS took 2 h and 30 min.

7 Conclusions

Most time-efficient drift detectors have an unrealistic assumption of label availability after the prediction of each example. On the other hand, unsupervised detectors are time-costly in problems with hundreds or thousands of features since they perform statistical tests for each feature. Given the need for fast detections in high-speed streams and the vast presence of data

[3] https://nlp.stanford.edu/projects/glove/.

[4] https://bit.ly/39cy83O.

with high dimensionality in problems related to time series, text mining, and computer vision, there is a need for accurate and time-efficient methods for concept drift detection in such scenario. In this direction, we introduced IBDD. We demonstrated that our proposal is faster and accurate as well as state-of-the-art unsupervised detectors. We based our discussion on experiments with different types of drifts on real data from several domains. A mechanism to indicate the subset of features drifting and the drift type's characterization are subjects for future work.

Acknowledgements This material is based upon work supported by the National Science Foundation under Award #OIA-1757207 and the Brazilian National Council for Scientific and Technological Development under Grant No. 142050/2019-9.

References

1. Bass C, Williamson MS, Wilding CS, Donnelly MJ, Field LM (2007) Identification of the main malaria vectors in the *Anopheles gambiae* species complex using a TaqMan real-time PCR assay. Malar J 6(1):155
2. Bergman LD, Rogowitz BE, Treinish LA (1995) A rule-based tool for assisting colormap selection. In: Proceedings visualization. IEEE, pp 118–125
3. Bifet A, Gavalda R (2007) Learning from time-changing data with adaptive windowing. In: International conference on data mining (SDM). SIAM, pp 443–448
4. Bifet A, Holmes G, Kirkby R, Pfahringer B (2010) MOA: massive online analysis. J Mach Learn Res 11(May):1601–1604
5. Brewer C (2015) Designing better Maps: a guide for GIS users. ESRI Press
6. Campana BJL, Keogh E (2010) A compression based distance measure for texture. In: International conference on data mining (SDM). SIAM, pp 850–861
7. Cieslak DA, Chawla NV (2009) A framework for monitoring classifiers' performance: when and why failure occurs? Knowl Inf Syst 18(1):83–108
8. Dau HA, Keogh E, Kamgar K, Yeh CM, Zhu Y, Gharghabi S, Ratanamahatana CA, Yanping C, Hu B, Begum N, Bagnall A, Mueen A, Batista G (2018), Hexagon-ML: the UCR time series classification archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/
9. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
10. Ditzler G, Polikar R (2011) Hellinger distance based drift detection for nonstationary environments. In: Symposium on computational intelligence in dynamic and uncertain environments (CIDUE). IEEE, pp 41–48
11. Ditzler G, Roveri M, Alippi C, Polikar R (2015) Learning in nonstationary environments: a survey. IEEE Comput Intell Mag 10(4):12–25
12. Dua D, Graff C (2017) UCI machine learning repository. http://archive.ics.uci.edu/ml
13. Dyer KB, Capo R, Polikar R (2013) Compose: a semisupervised learning framework for initially labeled nonstationary streaming data. IEEE Trans Neural Netw Learn Syst 25(1):12–26
14. Gama J, Medas P, Castillo G, Rodrigues P (2004) Learning with drift detection. In: Brazilian symposium on artificial intelligence (SBIA), pp 286–295
15. Gama J, Žliobaitė I, Bifet A, Pechenizkiy M, Bouchachia A (2014) A survey on concept drift adaptation. ACM Comput Surv 46(4):44
16. Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PC, Mark RG, Mietus JE, Moody GB, Peng C, Stanley HE (2000) Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. Circulation 101(23):215–220
17. González-Jiménez M, Babayan SA, Khazaeli P, Doyle M, Walton F, Reddy E, Glew T, Viana M, Ranford-Cartwright L, Niang A (2019) Prediction of mosquito species and population age structure using mid-infrared spectroscopy and supervised machine learning. Wellcome Open Res 4
18. Guo LZ, Zhou Z, Li YF (2020) Record: resource constrained semi-supervised learning under distribution shift. In: International conference on knowledge discovery & data mining (KDD). ACM, pp 1636–1644
19. Hawkins DM (1976) Point estimation of the parameters of piecewise regression models. J R Stat Soc Ser C (Appl Stat) 25(1):51–57
20. Howlader N, Noone AM, Krapcho M, Garshell J, Miller D, Altekruse SF, Kosary CL, Yu M, Ruhl J, Tatalovich Z (2015) Seer cancer statistics review, 1975–2012. National Cancer Institute, Bethesda, MD

21. Hu H, Kantardzic M, Sethi TS (2019) No free lunch theorem for concept drift detection in streaming data classification: a review. Wiley Interdiscip Rev Data Min Knowl Discov 10:e1327
22. Kaluža B, Mirchevska V, Dovgan E, Luštrek M, Gams M (2010) An agent-based approach to care in independent living. In: International joint conference on ambient intelligence (AMI), pp 177–186
23. Kifer D, Ben-David S, Gehrke J (2004) Detecting change in data streams. In: International conference on very large data bases (VLDB), pp 180–191
24. Korycki L, Krawczyk B (2019) Unsupervised drift detector ensembles for data stream mining. In: International conference on data science and advanced analytics (DSAA). IEEE, pp 317–325
25. Laikova KV, Oberemok VV, Krasnodubets AM, Gal'chinsky NV, Useinov RZ, Novikov IA, Temirova ZZ, Gorlov MV, Shved NA, Kumeiko VV (2019) Advances in the understanding of skin cancer: ultraviolet radiation, mutations, and antisense oligonucleotides as anticancer drugs. Molecules 24(8):1516
26. Liu J, Zhong L, Wickramasuriya J, Vasudevan V (2009) uWave: accelerometer-based personalized gesture recognition and its applications. Pervasive Mobile Comput 5(6):657–675
27. Maletzke A, Reis D, Cherman E, Batista G (2018) On the need of class ratio insensitive drift tests for data streams. In: Second international workshop on learning with imbalanced domains: theory and applications, pp 110–124
28. Marks R (1995) An overview of skin cancers. Cancer 75(S2):607–612
29. Moreland K (2009) Diverging color maps for scientific visualization. In: International symposium on visual computing. Springer, pp 92–103
30. Niculescu-Mizil A, Caruana R (2005) Predicting good probabilities with supervised learning. In: International conference on machine learning (ICML), pp 625–632
31. Reis DM, Flach P, Matwin S, Batista G (2016) Fast unsupervised online drift detection using incremental Kolmogorov–Smirnov test. In: International conference on knowledge discovery and data mining (KDD). ACM, pp 1545–1554
32. Roth GA, Abate D, Abate KH, Abay SM, Abbafati C, Abbasi N, Abbastabar H, Abd-Allah F, Abdela J, Abdelalim A (2018) Global, regional, and national age-sex-specific mortality for 282 causes of death in 195 countries and territories, 1980–2017: a systematic analysis for the global burden of disease study 2017. The Lancet 392(10159):1736–1788
33. Santolamazza F, Mancini E, Simard F, Qi Y, Tu Z, della Torre A (2008) Insertion polymorphisms of SINE200 retrotransposons within speciation islands of *Anopheles gambiae* molecular forms. Malar J 7(1):163
34. Sethi TS, Kantardzic M (2017) On the reliable detection of concept drift from streaming unlabeled data. Expert Syst Appl 82:77–99
35. Sobolewski P, Woźniak M (2013) Comparable study of statistical tests for virtual concept drift detection. In: International conference on computer recognition systems (CORES), pp 329–337
36. Souza VMA (2018) Asphalt pavement classification using smartphone accelerometer and complexity invariant distance. Eng Appl Artif Intell 74:198–211
37. Souza VMA, Cherman EA, Rossi RG, Souza RA (2017) Towards automatic evaluation of asphalt irregularity using smartphone's sensors. In: International symposium on intelligent data analysis (IDA). Springer, pp 322–333
38. Souza VMA, Chowdhury FA, Mueen A (2020) Unsupervised drift detection on high-speed data streams. In: International conference on big data. IEEE, pp 102–111
39. Souza VMA, Giusti R, Batista AJL (2018) Asfault: a low-cost system to evaluate pavement conditions in real-time using smartphones and machine learning. Pervasive Mobile Comput 51:121–137
40. Souza VMA, Pinho T, Batista GEAPA (2018) Evaluating stream classifiers with delayed labels information. In: Brazilian conference on intelligent systems (BRACIS). IEEE, pp 408–413
41. Souza VMA, Reis DM, Maletzke AG, Batista G (2020) Challenges in benchmarking stream learning algorithms with real-world data. Data Min Knowl Discov 34:1805–1858
42. Souza VMA, Silva DF, Batista G, Gama J (2015) Classification of evolving data streams with infinitely delayed labels. In: International conference on machine learning and applications (ICMLA). IEEE, pp 214–219
43. Souza VMA, Silva DF, Batista GEAPA (2013) Classification of data streams applied to insect recognition: initial results. In: Brazilian conference on intelligent systems (BRACIS), pp 76–81
44. Souza VMA, Silva DF, Gama J, Batista GEAPA (2015) Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: International conference on data mining (SDM). SIAM, pp 873–881
45. Tschandl P, Rosendahl C, Kittler H (2018) The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Sci Data 5:180161
46. Ulanova L, Begum N, Shokoohi-Yekta M, Keogh E (2016) Clustering in the face of fast changing streams. In: International conference on data mining (SDM). SIAM, pp 1–9

47. Wang Z, Bovik AC (2009) Mean squared error: love it or leave it? a new look at signal fidelity measures. IEEE Signal Process Mag 26(1):98–117
48. Yu L, Liu H (2003) Feature selection for high-dimensional data: a fast correlation-based filter solution. In: International conference on machine learning (ICML), pp 856–863
49. Yu S, Wang X, Príncipe JC (2018) Request-and-reverify: hierarchical hypothesis testing for concept drift detection with expensive labels. In: International joint conference on artificial intelligence (IJCAI), pp 3033–3039
50. Žliobaite I (2010) Change with delayed labeling: when is it detectable? In: International conference on data mining workshops (ICDMW). IEEE, pp 843–850

Vinicius M. A. Souza is an Associate Professor with the Graduate Program in Informatics (PPGIa) of the Pontifical Catholic University of Paraná (PUCPR), Brazil. Earlier, he was a Postdoctoral Research Fellow at the University of New Mexico, USA. He is Ph.D. in Computer Science and Computational Mathematics (2016) at the University of São Paulo, Brazil. He has authored over 40 articles in peer-reviewed conferences and journals, including Data Mining and Knowledge Discovery, Information Sciences, Engineering Applications of Artificial Intelligence, IEEE-ICDM, IEEE-BigData, and SIAM-SDM. His research interests include data mining, evolving data streams, and time series mining in applications involving smart sensors, public health, entomology, and agriculture.

Antonio R. S. Parmezan is a Ph.D. candidate of Computer Science and Computational Mathematics at the University of São Paulo (USP), Brazil. He holds an M.Sc. degree in Computer Science and Computational Mathematics at USP (2016) and a B.Sc. in Computer Science from Western Paraná State University (2012), Brazil. His research interests include machine learning, time series processing and analysis, data mining, data streams, feature selection, and metalearning.

Farhan A. Chowdhury is a Ph.D. student of Computer Science at the University of New Mexico, USA. He holds a B.Sc. in Electrical and Electronic Engineering from the Bangladesh University of Engineering and Technology (2016). He has articles in conferences such as ACM-SIGKDD, WWW, and IEEE-BigData. His research interests include temporal data mining, social network analysis, event, and anomaly detection.

Abdullah Mueen is an Assistant Professor in Computer Science at the University of New Mexico since 2013. Earlier, he was a Scientist in the Cloud and Information Science Laboratory at Microsoft Corporation. His major interest is in temporal data mining, focusing on two unique types of signals: social networks and electrical sensors. He has been actively publishing in the data mining conferences including ACM-SIGKDD, IEEE-ICDM and SIAM-SDM and journals including Data Mining and Knowledge Discovery and Knowledge and Information Systems. He has received a runner-up award in the Doctoral Dissertation Contest in ACM-SIGKDD 2012. He has won the best paper award at the same conference. Earlier, he earned Ph.D. degree at the University of California at Riverside and B.Sc. degree at Bangladesh University of Engineering and Technology.