# Concept Drift Detection Delay Index

Anjin Liu, Jie Lu, *Fellow, IEEE*, Yiliao Song, *Member, IEEE*, Junyu Xuan, and Guangquan Zhang

**Abstract**—Data streams may encounter data distribution changes, which can significantly impair the accuracy of models. Concept drift detection tracks data distribution changes and signals when to update models. Many drift detection methods apply thresholds to distinguish between drift or non-drift streams and to claim their method outperforms others with non-aligned drift thresholds. We consider that selecting a proper drift threshold could be more important than developing a new drift detection algorithm, and different drift detection algorithms may end up with very similar performance with aligned drift thresholds. To better understand this process, we propose a novel threshold selection algorithm to align the drift thresholds of a set of algorithms so that they are all at the same sensitivity level. Based on comprehensive experiment evaluations, we observed that several state-of-the-art drift detection algorithms could achieve similar results by aligning their thresholds, providing a novel insight to explain how drift detection algorithms contribute to data stream learning. We noticed that a higher detection sensitivity improves accuracy for data streams with frequent distribution change. The evaluation results are showing that drift thresholds should not be fixed during stream learning. Rather, they should adjust dynamically based on the prevailing conditions of the data stream.

**Index Terms**—data stream, data mining, concept drift, machine learning

## 1 Introduction

### 1.1 Motivation

Conventional machine learning models assume the training data represents the learning tasks to be completed adequately and that the training data follow the same distribution as the production data [1]–[3]. However, in real-world scenarios this assumption may not hold, especially for applications involving data stream analysis [1], [3], [4]. For example, in industrial applications the data collected from manufacturing processes tend to be inherently nonstationary. Sensors develop faults, they age, and the precision of their measurements can change in operating conditions [5]. With enough drift from one concept to another, the model becomes so inaccurate as to be effectively useless. What drift detection mechanisms do is to identify such a drift and to tell learning models how significant the drift is.

In the literature, performance-based drift handling algorithms form the largest category of concept drift detection [1], [4]. These algorithms focus on tracking changes in learning model performance through error rates or other quantitative indicators [6], [7]. If the increase in errors is significant enough, it triggers a model update. The most popular model of this type was designed by Gama [8]. It consists of both a warning threshold and a drift threshold. An independent performance monitoring system continuously tracks the prediction results of the model. If the warning threshold is reached, the system starts setting aside newly arriving observations to form a new training set. Should the drift threshold subsequently be reached, the system tells the model to update itself with the new training set.

However, the best way to define warning and drift thresholds, so as to maximize performance, has never been fully explored. Currently, the best practice is to define and set them using expert knowledge [1], [9]. As a result, today's drift detection algorithms all have different drift thresholds with different value ranges.

The idea that an algorithm's drift detection threshold contributes just as much to performance as the detection mechanism itself raises a string of questions. How do we know whether an algorithm is actually effective, or whether the threshold simply needs adjusting? Is the current thinking that drift thresholds should be fixed during data stream learning correct? Might it be better if they could change dynamically to suit the prevailing condition of the data stream? With the goal of answering all these questions, we developed a basic idea that if we could somehow quantify the influence of the drift threshold on a drift detection algorithm, we could align multiple drift detection algorithms. One intuitive drift detection robustness indicator is the false alarm rate. If we could set the drift thresholds for two drift detection algorithms $H_1$ and $H_2$ so that they have the same false alarm rate when there is no drift, then we could compare them in the presence of drift on a more equal footing.

TABLE 1
The different recommended drift thresholds of various drift detection algorithms implemented in skmultiflow

[10].

| Detection algorithm | Default threshold | Threshold range |
|---|---|---|
| ADWIN | $delta = 0.002$ | $[0, +\infty)$ |
| DDM [8] | $out\_countrol\_level = 3$ | $[0, +\infty)$ |
| EDDM [11] | $fddm\_outcontrol = 0.9$ | $[0, 1]$ |
| HDDM-A [12] | $drift\_confidence = 0.001$ | $[0, 1]$ |
| HDDM-W [12] | $drift\_confidence = 0.001$ | $[0, 1]$ |
| Page-Hinkley [13] | $Threshold = 50$ | $[0, +\infty)$ |
| KSWIN [14] | $alpha = 0.005$ | $[0, 1]$ |

Another critical issue is how to find the best drift threshold for a drift detection algorithm during learning under dynamic data stream. As discussed in a recent survey [1], a properly set drift threshold is critical to stream learning results, and the default drift threshold of a drift detection algorithm may not provide the best possible results with ev-

ery stream. Most often, a manual tuning process is required to find the optimal setting. The different threshold range problem raises an additional challenge related to this issue. For example, the threshold range of ADWIN [15], DDM [8] is $[0, +\infty)$ but for EDDM [11], HDDM-A, HDDM-W [12] it is $[0, 1]$, as summarized in Table 1. If these thresholds cannot be aligned to remove their impact as a factor, it is difficult to know which algorithm is the best choice for the task at hand. Without repeated testing and turning, we cannot know whether bad performance is the result of an incorrectly set threshold or a poorly suited detection mechanism.

### 1.2 Challenges

In summary, without a way to align the thresholds of drift detection algorithms, we are facing two challenges issues: 1) The comparison result may bias for drift detection algorithms with non-aligned drift thresholds. 2) It hinders researchers from developing adaptive drift threshold selection algorithms. When considering that a drift detection algorithm has two major components – the detection mechanism and threshold selection – our ability to design a better drift detection solution rest on our understanding of how detection mechanisms perform with different thresholds. The lack of a systematic way to manipulate drift thresholds across a set of detection algorithms is hindering research progress.

### 1.3 Contributions

Hence, in this paper we propose a novel schema for comparing concept drift detection algorithms that consider learning accuracy with aligned drift detection threshold. The main components of the schema include:

- The concept drift Detection Delay Index (DD Index) – a novel robustness scale for quantitatively comparing and evaluating algorithms that detect concept drift based on error rates. The DD Index helps explain whether the difference between two drift detection algorithms is caused by their drift thresholds or their detection mechanisms.
- The robustness alignment algorithm – an automatic drift threshold aligning algorithm. The proposed algorithm effectively removes threshold tuning from the process of evaluating a set of drift detection algorithms, allowing researchers to focus solely on how each algorithm's robustness/sensitivity impacts the learning process.

From a series of experiments with and without the DD Index and threshold alignment, we find that model validation errors and stream drift frequency are two critical factors affecting drift threshold selection and that proper threshold selection is critical to overall learning. In general, high detection robustness performs well on data steams where drifts are infrequent and vice versa. We also find strong correlations between learning accuracy, robustness levels, drift severity, and drift frequency, which inspires us to believe that adaptive drift thresholds could improve learning quality under concept drift. Hence, we contend that drift thresholds should not be fixed, but rather should be adjusted dynamically based on the prevailing conditions in the current data stream.

The rest of this paper is organised as follows. In Section 2, the problem of concept drift detection and related works are discussed. Sections 3 and 4 introduce the DD Index and the recursive drift threshold selection algorithm. Section 5 demonstrates how the DD Index works and evaluates the drift threshold searching algorithm. Section 6 concludes this study with a discussion of future work.

## 2 Literature Review

This section introduces the definition of concept drift, followed by a review of learner-based drift detection algorithms. At last, we discuss two common metrics to evaluate the performance of a drift detection algorihtm.

### 2.1 Concept Drift Definition

Concept drift occurs when the distribution of a model's training set no longer matches the distribution of the data currently being analysed. Most common with streaming data or nonstationary learning environments [16]–[19]. Formally, concept drift is defined as follows. Consider a feature space, denoted as $\mathcal{X} \subseteq \mathbb{R}^n$, where $n$ is the dimensionality of the feature space. A data instance $d_t = (X, y)$ is a pair of feature vectors $X$ and a label $y$, where $y \in \{y_1, \ldots, y_c\}$ and $c$ is the number of classes. A data stream can be represented as an infinite sequence of data instances. A concept drift occurs at time $t$ if the joint probability of $X$ and $y$ changes, i.e., $p_t(X, y) \neq p_{t+1}(X, y)$ [4], [20], [21]. Further decomposing $p(X, y)$, we have $p(X, y) = p(y|X) \times p(X)$. If we only consider problems that use $X$ to infer $y$, denoted as $X \rightarrow y$, concept drift can be divided into two research streams: covariate shift and concept shift [17].

- Covariate shift focuses on the drift in $p(X)$, while $p(y|X)$ remains unchanged. This is more commonly known as $virtual\ drift$ [3], [4], [22].
- Concept shift focuses on the drift in $p(y|X)$, while $p(X)$ remains unchanged. This is standard concept drift, also called $actual\ drift$ or $decision\ boundary\ drift$ [3], [4], [22].

To the best of our knowledge, no study has proven any relationship of necessity or sufficiency between virtual drift and actual drift without prior knowledge. However, in practice, changes in $p(X)$ are often strongly correlated to $p(y|X)$. Moreover, detecting and adapting to virtual drift (i.e., changes in $p(X)$) without presumptions does not necessarily improve learning performance. Therefore, to avoid unnecessary drift detection for questionable benefit, we have solely focused on actual drift.

### 2.2 Learner-based Algorithms

Drift detection algorithms based on error rates are dedicated to capturing actual drift. The intuition behind this type of algorithms is to monitor fluctuations in the error rate [23], [24]. Gama et al. [4] categorised them as change detection techniques, further dividing them into four subcategories: sequential analysis, control charts, monitoring two distributions, and contextual. In a more recent survey, Yu and

TABLE 2
We merged DDM and DELM into one row because they are using the same drift detection algorithm. The difference between is in the drift adaption strategy. HDDM detects drift in an online mode, while STEPD is batch mode.

| Algorithm | abrupt | gradual | regional | noise | statistic |
|---|---|---|---|---|---|
| DDM, DELM | ✓ | | | | ✓ |
| EDDM | ✓ | ✓ | | | ✓ |
| FWDDM | ✓ | ✓ | | ✓ | |
| LLDD | ✓ | ✓ | ✓ | | |
| HDDM | ✓ | ✓ | | ✓ | ✓ |
| STEPD | ✓ | ✓ | | ✓ | ✓ |

Abraham [25] added further performance indicators beyond the error rate to the list of considerations, including the rates of true positives and negatives, and the positive and negative predicted values. Most drift thresholds have unique definitions making them difficult to compare directly.

One of the top referenced concept drift detection algorithms is the DDM [8], which was the first algorithm to contain defined warning and drift levels for signalling concept drift. The alarms are straightforward, based on simple increases in the error rate within a given time window. If the error rate reaches the warning level, DDM starts building a new learner, but continues using the old learner for predictions until the error rate reaches the drift level. Once the error rate exceeds the drift level, the old learner is replaced with the new learner, as illustrated in Fig. 1.

Many subsequent algorithms have adopted a similar implementation, e.g., LLDD [26], EDDM [11], HDDM [12], FWDDM [27], DELM [28]. However, each defines the warning and drift level thresholds differently. For instance, ECDD's definitions are based on a modified EWMA chart that uses a dynamic mean instead of the conventional static mean. The method outlined in [29], called the Statistical Test of Equal Proportions Detection (STEPD) detects changes in the error rate through a statistical significance test of the difference in the error rate in the most recent time window with the overall error rate. The test statistic that quantifies the change in error rate between the two windows has been proven to conform to a standard normal distribution. STEPD suggests a warning level of $\alpha_{warn} = 0.05$ and a drift level of $\alpha_{drift} = 0.003$. We summarize the characteristics of the most popular error-rate based drift detection algorithms from five dimensions. 1) sensitivity to abrupt drift; 2) Sensitivity to gradual drift; 3) Sensitivity to regional drift; 4) Robust to noise; 5) Drift threshold can be defined by statistic significant level, as shown in Table 2

Although most drift thresholds are based on some version of a statistical hypothesis test, their recommended significance levels are different. Additionally, there are few drift thresholds are defined by expert knowledge. We contend that a more generalized drift threshold selection process would help users better understand the correlations between the drift detection mechanism, the drift thresholds and stream learning performance.

### 2.3 Statistical Hypothesis Testing

A statistical hypothesis test means to test a claim by modelling a set of observed values as a collection of random

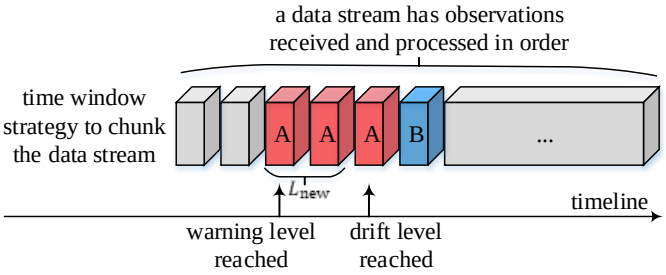



Fig. 1. An illustration of the warn/drift-level concept drift handling framework. When a true label or feedback is available, the monitoring system evaluates the severity of the change. If learner performance drops moderately, the system will start buffering new observations. Should performance subsequently return to normal, the system will reset the buffer. However, if performance continues to deteriorate such that it reaches the drift threshold, a new learner $L_{new}$ will be built from the buffered data.

variables with a joint probability distribution in some set of possible joint distributions and then infer whether, according to a threshold of probability, i.e., a significance level, the values observed are likely or unlikely to have followed the given probability distribution [30]–[33].

This form of testing has been widely used to determine whether a model has become outdated due to concept drift. The general premise is to assume a null hypothesis that the model is working as expected and then accept or reject the null case by comparing the model's most recent predictions with a set of its historical predictions. Typically, the significance level, i.e., grounds for rejecting the null case, is defined as a maximal allowed "false positive rate". And, because controlling the risk of incorrectly rejecting a true null hypothesis is usually the priority, Type-I and Type-II errors are the two most commonly used metrics for evaluating the efficacy of a hypothesis testing method [30], [31], [34].

- A Type I error indicates that a null hypothesis was rejected when it was, in fact, true. In other words, this metric shows the number of false alarms raised when there was no concept drift, but the algorithm said there was.
- A Type II error indicates that the null hypothesis was not rejected when it should have been. Often simply called missing rate, this error tells us when concept drift had occurred, but the algorithm failed to detect.

Much of statistical theory revolves around minimising of one or both of these errors, although completely eliminating either is a statistical impossibility for non-deterministic algorithms. The quality of a hypothesis test can be increased by selecting a low threshold (cut-off) value and modifying the p-value.

## 3 Concept Drift Detection Delay Index

This section introduces the concept drift DD Index, which quantifies the impact of the detection threshold on error rate-based drift detection algorithms. In Section 3.1, we explain that error rate-based drift detection is closely related to the problem of binomial distribution estimation and that, by changing the parameters of the distribution, binomial

TABLE 3
Notation summary

| Notation | Description |
|---|---|
| $W, w$ | time window |
| $\delta$ | concept drift margin |
| $\theta$ | a drift threshold |
| $H$ | a drift detection algorithm or a hypothesis test |
| $\epsilon$ | the mean of observed error rates |
| $\varepsilon$ | the expectation of error rate |
| $\omega$ | a user defined drift detection robustness level |
| $\Omega(H, \theta)$ | a function to measure the drift detection robustness level for a drift detection algorithm $H$ with a given threshold $\theta$ |

distributions can be used to simulate changes in error rates. Section 3.2 outlines how to empirically estimate DD Index as a uniform criterion for drift threshold selection with different error rate-based detection methods. The notations are summarized in Table 3

### 3.1 Error Rate-based Concept Drift Detection and Binomial Distribution

In this section, we first state the relationship between the error rate and Binomial distribution. Then we introduce the intuition behind DD Index and our drift threshold alignment algorithm.

The problem setting is as follows. Assume we have two chunks of data contained in two disjoint time windows $W_1$ and $W_2$, where

$$W_i = (X_t, y_{t+0.5})_{t \in w_i},$$

and $w_i$ is the time interval of $W_i$. The class label $y_{t+0.5}$ indicates that the true label will be received after receiving the feature $X_t$ but before receiving feature $X_{t+1}$ with the next observation. If a learning model $L$ can correctly predict any observation $(X_t, y_{t+0.5})$, the error is counted as 0, and 1 otherwise, or simply denoted as

$$\epsilon_t = |y_{t+0.5} - L(X_t)|.$$

Therefore, the problem of error rate-based drift detection can be stated as follows:

**Problem 1**. (Error Rate-based Drift Detection) Detecting concept drift via error rate is a task of measuring the difference in prediction accuracy for model $L$ between two time windows $W_1$ and $W_2$. It can also be thought of as a two-sample hypothesis test to accept or reject whether $\{\epsilon_t\}_{t \in w_1}$ and $\{\epsilon_t\}_{t \in w_2}$ are drawn from the same population. Formally, this hypothesis is expressed as

$$\mathbb{E}_{P_{w_1}(X,y)} \left( |y - L(x)| \right) \neq \mathbb{E}_{P_{w_2}(X,y)} \left( |y - L(x)| \right),$$

or shorten as $\varepsilon_{w_i} \neq \varepsilon_{w_{i+1}}$. If the prediction errors $\{\epsilon_t\}_{t \in w}$ are i.i.d., the errors of $L$ can be described as a sequence of binomial trails, that is,

$$\epsilon \sim Binomial\left(1,\ \varepsilon\right),$$

where $\varepsilon$ is the expectation of the error rate. In practice, $\varepsilon$ can be estimated by the sequence of errors in a time window $w$, i.e., $\hat{\varepsilon}_w = \frac{1}{|w|} \sum_{t=1}^{|w|} \epsilon_t$.

From the drift detection perspective, a concept drift detection algorithm should be able to identify drifts in the error sequence. If we could generate a sequence of independent and identically distributed binomial trails, a valid concept drift detection algorithm must be able to identify the drift points in the sequence unbiased. Examples of how to simulate error rate drift through a binomial distribution follows.

**Example 1**. (Error Rate Drift Simulation) Generate two sequence of binomial trails: one of length $n_{valid}$ with a success rate $(1-\varepsilon)$ denoted as $\{\epsilon_t\}_{t=1}^{n_{valid}}$ and one of length $n_{test}$ with a success rate of $(1-\varepsilon')$ denoted as $\{\epsilon'_t\}_{t=1}^{n_{test}}$. Concatenating $\{\epsilon_t\}_{t=1}^{n_{valid}}$ and $\{\epsilon'_t\}_{t=1}^{n_{test}}$ gives an error sequence with a drift margin equal to

$$\delta = |\varepsilon - \varepsilon'|$$

and a drift point located at $n_{valid}$ out of $(n_{valid} + n_{test})$.

This simulated error sequence can be used to evaluate the robustness of a drift detection algorithm $H$ and to search for the best drift threshold. More details are shown in Examples 2 and 3.

**Example 2**. (Drift Detection Robustness Measurement) Consider a drift detection algorithm $H_1$ and a default drift threshold of $\theta_{1,0}$. Setting $\varepsilon = \varepsilon'$ such that no drift exists. Using $H_1$ with $\theta_{1,0}$ to detect the drift point $t'$. A drift point will be found some where $n_{valid} \leq t' \leq n_{valid} + n_{test}$.

Scaling the detected drift point by $\frac{t' - n_{valid}}{n_{test}}$, we could have a drift delay indicator, denoted as

$$\Theta_{\varepsilon,\varepsilon'}\left(H_1,\ \theta_{1,0},\ n_{valid},\ n_{test}\right) \in [0,\ 1]\,.$$

The closer the $\Theta_{\varepsilon,\varepsilon'}\left(H_1,\ \theta_1,\ n_{valid},\ n_{test}\right)$ is to 1, the more robust $H_1,\ \theta_{1,0}$ are.

**Example 3**. (Drift Detection Robustness Alignment) Again considering algorithm $H_1$, let us set $\Theta_{\varepsilon,\varepsilon'}\left(H_1,\ \theta_{1,i},\ n_{valid},\ n_{test}\right)$=0.99 and the search space of the drift threshold

$$\theta_{1,i} \in \{0.001,\ 0.002, \ldots, 1\}\,.$$

By iteration, we can find a $\theta_{1,i}$ that satisfies

$$\Theta_{\varepsilon,\varepsilon'}\left(H_1,\ \theta_{1,i},\ n_{valid},\ n_{test}\right) \geq 0.99,$$

and this process can be repeated for a set of drift detection algorithms

$$\mathcal{H} = \{H_1, \ldots,\ H_j\}$$

such that

$$\Theta_{\varepsilon,\varepsilon'}\left(H_j,\ \theta_{j,i},\ n_{valid},\ n_{test}\right) \geq 0.99$$

for all $H_j \in \mathcal{H}$. Thus, all algorithms in the set $\mathcal{H}$ are aligned at a robustness level of 0.99 with their unique thresholds.

In general, there are two major approaches to selecting a drift threshold in the literature. Approach one is statistical significance level-based threshold selection. For example, setting drift threshold based on p-value or $\alpha$, such as setting $\alpha = 0.05$ as warning level and $\alpha = 0.01$ as drift level. Approach two is a heuristic method that sets the thresholds based on cross-validation results or based on expert knowledge. By contrast, we propose a Monte Carlo drift threshold selection solution, that is, setting a desired false alarm rate and running the given drift detection algorithm on $n$ synthetic binomial data streams. Then we adjust the drift threshold recursively based on the observed false alarm rate.

### 3.2 The Concept Drift DD Index

In Section 3.1, we discussed how to leverage binomial distribution to simulate concept drift. In this section, we propose a scalar measurement, called the Detection Delay Index (DD Index), to quantify the efficacy of drift detection methods on simulate drifts.

Concept drift detection and statistical hypothesis testing have both overlapped research objectives and evaluation metrics. The Type-I and Type-II errors widely used in hypothesis testing are also commonly used to evaluate drift detection algorithms. One difference between drift detection and hypothesis testing, however, is that drift detection deals with observations received in a time span, which raises issues over the time lag between when a drift occurs and the time it takes to detect it. This problem, known as the detection delay problem, is not measured by Type-I and -II errors. As an example, consider two sets of observations $W_1$ and $W_2$ of equal size at 100 samples but a slight difference in distribution. With this information alone, algorithm $H_1$ could not detect a drift between $W_1$ and $W_2$. However, as $W_2$ receives more observations and the number of samples increases to 150, $H_1$ returns a drift alarm. Neither a Type-I nor a Type-II error can reflect the 50-sample delay between when the distributions began to deviate and when the alarm triggered.

An efficient way of evaluating detection delays is an open problem. Currently, the best method is a rudimentary observation count, denoted as $n_{count}$. But, beyond its obvious tedium, this type of approach can only be used if the user knows there is a drift. Hence, in any pre-emptive strategy, if no drift occurs, the count becomes infinite and incomparable. Additionally, counts and rates, like Type-I and Type-II errors, cannot be integrated as one measurement, further complicating the evaluation process.

The intuition of the DD Index is to resolve the shortcomings with counting the number of delayed observations by predefining an upper limit for the counting process to stop. This upper limit is denoted as $n_{test}$. Further, as discussed in Example 2, the delay between when the drift occurs and when it is detected can be counted in terms of the number of observations. Accordingly, $n_{count}$ can be scaled in the interval $[0, 1]$ by $\frac{n_{count}}{n_{test}}$ to create a scaled drift delay indicator, denoted as $\Theta_{\varepsilon,\varepsilon'}(H_i,\ \theta_{i,j},\ n_{valid},\ n_{test})$. A value of 0 indicates that the drift was detected at the exact time the drift occurred. The indicator moves toward 1 with each additional observation it takes to detect the drift, reaching 1 if no drift has been detected after $n_{test}$ observations. Since $n_{valid}$ and $n_{test}$ are both hyperparameters and all drift detection algorithms use them when calculating $\Theta_{\varepsilon,\varepsilon'}(H_i,\ \theta_{i,j},\ n_{valid},\ n_{test})$, we have shortened the notation to $\Theta_{\varepsilon,\varepsilon'}(H_i,\ \theta_{i,j})$. More details on the impacts of $n_{valid}$ and $n_{test}$ on the algorithms are evaluated and discussed in the Experiments section.

Notably, estimating the scaled delay indicator only once may carry a high bias. Nor may one estimation be enough to accurately reflect the Type-I and -II error rates. Therefore, according to the law of large numbers, the drift detection process is repeated a sufficiently large number of times to ensure the average of the scaled delay indicators comes close to the expected value. The settings of the simulated error sequences stay the same in each repetition, i.e., drift point: $n_{valid} + n_{test}$ and drift severity $|\varepsilon - \varepsilon'|$; only the random seed changes. Thus, the formal definition of DD index is:

**Definition 1**. (Detection Delay Index) Given a drift detection algorithm $H_i$ and a drift threshold $\theta_{i,j}$, the DD Index of the change in error rate from $\varepsilon$ to $\varepsilon'$ is the expectation of the scaled delay indicator, denoted as

$$\Omega_{\varepsilon,\varepsilon'}(H_i, \theta_{i,j}) = \mathbb{E}\left(\Theta_{\varepsilon,\varepsilon'}(H_i, \theta_{i,j})\right).$$

Since the DD Index is a measure of robustness, for simplicity, we use the same notation $\Omega$ to represent both the DD Index and robustness levels. Admittedly, the DD Index does not entirely replace Type-I and Type-II errors as an evaluation metric, but what it does provide some extra and necessary information regarding false alarm rates and missing detection rates. For example, if a drift detection algorithm $H_i$ has a high Type-I error with a threshold of $\theta_{i,j}$, which means it has a high false alarm, the DD Index $\Omega_{\varepsilon,\varepsilon'}(H_i,\ \theta_{i,j})$ will be close to 0. In this case, setting $|\varepsilon - \varepsilon'| = 0$ will simulate a stationary error sequence. Whereas, if $H_i$ has high Type-II errors with the threshold $\theta_{i,j}$, i.e., a high missing rate, the DD Index $\Omega_{\varepsilon,\varepsilon'}(H_i,\ \theta_{i,j})$ will be close to 1. Here, $|\varepsilon - \varepsilon'|$ should be set to $> 0$ to simulate a nonstationary error sequence. According to the strong law of large numbers (Kolmogorov's law), we have the sample average converges almost surely to the expected value

$$\lim_{n_{run}\to\infty} \frac{1}{n_{run}} \sum_{k=1}^{n_{run}} \Theta^k_{\varepsilon,\varepsilon'}(H_i, \theta_{i,j}) \xrightarrow{a.s.} \mathbb{E}\left(\Theta_{\varepsilon,\varepsilon'}(H_i,\ \theta_{i,j})\right),$$

where $n_{run}$ is a sufficiently large number. The DD Index and the law of large numbers are the core components of our robustness alignment algorithm. The implementation algorithm to calculate DD Index is shown in Algorithm 1.

The required parameters for Algorithm 1 are the drift detection algorithm $H$ and its drift threshold $\theta$. The initial error rate $\varepsilon$ simulates the performance of a learning model. The drift error rate $\varepsilon'$ simulates its performance after concept drift. The initial lengths of the error sequence $n_{valid}$ and drift error sequence $n_{test}$ control the length of the error sequence before and after concept drift. The number of repeating tests $n_{run}$ tells the algorithm how many times to run the detection process to compute the average DD Index. Following the law of large numbers, $n_{run}$ should be a sufficiently large number. Considering the runtime complexity of the algorithm, we set $n_{run} = 5000$ as a default.

The basic workflow is to use a binomial distribution to simulate a sequence of prediction errors made by a learning model (Lines 3, 4). The probability of the model making a successful prediction is then changed to simulate a change in the model's error rate. The number of observations required to detect the drift is then counted and normalised against the total number of available observations (Lines 5-13). This counting process is then repeated a sufficiently large number of times to provide a relatively accurate average estimate of the DD Index (Lines 1, 2, 14-17).

**Algorithm 1:** Concept Drift DD Index

**input :** 1. drift detection algorithm, $H$
2. drift threshold, $\theta$
3. validation error rate, $\varepsilon$
4. test error rate, $\varepsilon'$
5. validation error sequence size, $n_{valid} = 80$
6. test error sequence size, $n_{test} = 200$
7. number of repeating tests, $n_{run} = 5000$

**output:** 1. DD Index value, $\Omega_{\varepsilon,\varepsilon'}(H,\theta)$

Initialise the DD Index, $\Theta_{\varepsilon,\varepsilon'}(H,\theta) = 0$;
**for** *k in range* ($n_{run}$) **do**
  Generate validation sequence, $\{\epsilon_i\}_{i=1}^{n_{valid}}$, that $\epsilon_i \sim Binomial(1,\varepsilon)$;
  Generate test sequence, $\{\epsilon'_j\}_{j=1}^{n_{test}}$, that $\epsilon'_j \sim Binomial(1,\varepsilon')$;
  Input $\{\varepsilon_i\}_{i=1}^{n_{valid}}$ to $H$;
  Initialise $n_{count} = n_{test}$;
  **for** *j in range* ($n_{test}$) **do**
    Input $\epsilon'_j$ to $H$;
    **if** $H$ *is in drift status* **then**
      delay counter $n_{count} = j$;
      break;
    **end**
  **end**
  $\Theta_{\varepsilon,\varepsilon'}(H,\theta) = \Theta_{\varepsilon,\varepsilon'}(H,\theta) + \frac{n_{count}}{n_{test}}$;
**end**
$\Omega_{\varepsilon,\varepsilon'}(H,\theta) = \frac{1}{n_{run}}\Theta_{\varepsilon,\varepsilon'}(H,\theta)$;
**Return** $\Omega_{\varepsilon,\varepsilon'}(H,\theta)$

## 4 Drift Threshold Alignment

### 4.1 The Recursive Drift Threshold Alignment Algorithm

The intuition of robustness alignment is to iteratively search a drift threshold $\theta_{i,j}$ for a drift detection algorithm $H_i$ so that $\Omega_{\varepsilon,\varepsilon'}(H_i,\theta_{i,j})$ equals a desired robustness level $\omega$. However, implementing this idea in reality is not quite so straightforward. Hence, in this section, we discuss some of the problems and solutions to robustness alignment with the DD Index.

The first issue that needs to be addressed is how to define the search space of a drift threshold $\theta_{i,j}$. Since a search space for $\theta_{i,j}$ could be infinite, e.g., $\theta_{i,j} \in \mathbb{R}_{[0,\,1]}$, continuous domains need to be converted into discrete sets. There are three parameters that can be used for this conversion. They are: the maximum robustness threshold value $\theta_{maxRob}$, the minimum robustness threshold value $\theta_{minRob}$, and the threshold search gap $\theta_{gap}$. The parameters $\theta_{maxRob}$, $\theta_{minRob}$ control the upper and lower bounds of the search space. To define them more specifically, consider a drift threshold value of $\theta_{i,j} \in \mathbb{R}_{[0,\,1]}$. If a drift detection algorithm $H_i$ has a maximum robustness of $\theta_{i,j} = 1$, i.e., no drift should be reported under any circumstances, then $\theta_{maxRob} = 1$. By the same token, $\theta_{minRob} = 0$ means that when $\theta_{i,j} = 0$, the drift detection algorithm $H_i$ will report drift even no drift existed. The threshold searching gap $\theta_{gap}$ controls the discretisation level of the drift threshold $\theta_{i,j}$. A simple example of using $\theta_{maxRob}$, $\theta_{minRob}$ and $\theta_{gap}$ to generate a finite discrete set is

$$\{\theta_{minRob}, \theta_{minRob} + \theta_{gap}, \theta_{minRob} + 2 \times \theta_{gap}, \ldots, \theta_{maxRob}\},$$

which would give a threshold searching set of size

$$n_\theta = \left\lceil \frac{\theta_{maxRob} - \theta_{minRob}}{\theta_{gap}} \right\rceil.$$

To optimise the searching process, a recursive searching algorithm is designed to leverage the monotonicity of the drift thresholds and detection robustness. As such, the algorithm only needs to compute $\Omega_{\varepsilon,\varepsilon'}(H_i,\theta_{i,j})$ for $\log_2(n_\theta)$ times rather than $n_\theta$ times. The idea behind the recursion is to devise a method of solving a problem where the solution depends on solutions to smaller instances of the same problem. Such issues can generally be solved by iteration, but this requires both identifying and indexing the smaller instances at the time of programming.

The detection robustness of a drift detection algorithm in terms of its drift threshold can be represented as a monotonic function. Given two threshold values $\theta_{i,\,1}$ and $\theta_{i,\,2}$ for $H_i$, if $\theta_{i,\,1} \leq \theta_{i,\,2}$, then $H_i$ must have either

$$\Omega_{\varepsilon,\varepsilon'}(H_i,\theta_{i,1}) \leq \Omega_{\varepsilon,\varepsilon'}(H_i,\theta_{i,2})$$

or

$$\Omega_{\varepsilon,\varepsilon'}(H_i,\theta_{i,1}) \geq \Omega_{\varepsilon,\varepsilon'}(H_i,\theta_{i,2}).$$

The monotonically increasing function in our proposed recursive threshold searching algorithm is described below. For brevity, we have spared a duplicate description of the decreasing case, which simply reverses the values of $\theta_{minRob},\ \theta_{maxRob}$.

With the discretisation strategy and assumption of monotonicity, the search task can be completed with a divide-and-conquer approach:

$$Searching\ for\ \theta_{i,j}\ in\ (\theta_{minRob}, \theta_{maxRob}, \theta_{gap})$$
$$such\ that\ \Omega_{\varepsilon,\varepsilon'}(H_i,\theta_{i,j}) = \omega$$

by searching for $\theta_{i,j}$ in

$$\left(\theta_{minRob}, \frac{\theta_{maxRob} + \theta_{minRob}}{2}, \theta_{gap}\right),$$

if $\Omega_{\varepsilon,\varepsilon'}(H_i,\theta_{minRob}) < \omega < \Omega_{\varepsilon,\varepsilon'}\left(H_i, \frac{\theta_{maxRob}+\theta_{minRob}}{2}\right)$. Otherwise, searching for $\theta_{i,j}$ in

$$\left(\frac{\theta_{maxRob} + \theta_{minRob}}{2}, \theta_{maxRob}, \theta_{gap}\right).$$

Note that the above subproblems have the same formulation as the original search problem. As a result, only $\Omega_{\varepsilon,\varepsilon'}\left(H_i, \frac{\theta_{maxRob}+\theta_{minRob}}{2}\right)$ needs to be computed and only $\theta_{maxRob}$ to $\frac{\theta_{maxRob}+\theta_{minRob}}{2}$ (or $\theta_{minRob}$ to $\frac{\theta_{maxRob}+\theta_{minRob}}{2}$) needs to be updated in each recursion. As a result, we have the recursion stop conditions are:

**Condition 1 – Desired threshold found:**

$$\Omega_{\varepsilon,\varepsilon'}(H_i,\theta_{minRob}) = \omega\ or\ \Omega_{\varepsilon,\varepsilon'}(H_i,\theta_{maxRob}) = \omega$$

**Condition 2 – Search space limitation reached:**

$$\theta_{maxRob} - \theta_{minRob} \leq \theta_{gap}$$

**Condition 3 – Search space out of range:**

$$\Omega_{\varepsilon,\varepsilon'}(H_i,\theta_{minRob}) > \omega\ or\ \Omega_{\varepsilon,\varepsilon'}(H_i,\theta_{maxRob}) < \omega$$

If the threshold is found (Condition 1), the algorithm returns $\theta_{minRob}$ if $\Omega_{\varepsilon,\varepsilon'}(H_i,\theta_{minRob}) = \omega$, and $\theta_{maxRob}$ otherwise.

**Algorithm 2:** Recursive Threshold Searching for Robustness Alignment

**input :** 1. default parameters for Alg. 1 ($H$, $\varepsilon$, $\varepsilon'$, $n_{valid}$, $n_test$, $n_{run}$)
2. threshold searching space, $\{\theta_{minRob},\ \theta_{maxRob},\ \theta_{gap}\}$
3. desired robustness level, $\omega$

**output:** 1. the threshold $\theta_{i,j}$, so that $\Omega_{\varepsilon,\varepsilon'}(H_i, \theta_{i,j}) = \omega$

**if** *recursion stop condition 1-3* **then**
  return corresponding $\theta_{i,j}$;
**end**
**if** $\Omega_{\varepsilon,\varepsilon'}\left(H_i, \frac{\theta_{maxRob}+\theta_{minRob}}{2}\right) > \omega$ **then**
  **Return** recursive search $\theta_{i,j}$ in $\left(\theta_{minRob},\ \frac{\theta_{maxRob}+\theta_{minRob}}{2},\ \theta_{gap}\right)$;
**else**
  **Return** recursive search $\theta_{i,j}$ in $\left(\frac{\theta_{maxRob}+\theta_{minRob}}{2},\ \theta_{maxRob}, \theta_{gap}\right)$;
**end**

If the space has been fully searched (Condition 2), the algorithm returns $\theta_{maxRob}$. Out of range errors (Condition 3) raise a range of warnings. The pseudocode is given in Algorithm 2.

## 5 Experiments and Analysis

To evaluate the DD Index and the robustness alignment algorithm, we conducted four experiments. The first two experiments in Section 5.1 were designed to analyse the effectiveness of the DD Index in selecting appropriate drift thresholds for a range of different drift detection algorithms so as to align their robustness. The third experiment in Section 5.2 tests how different drift detection algorithms perform under different concept drift conditions in terms of different robustness level. The fourth experiment in Section 5.3 tests the usefulness of the robustness alignment algorithm for evaluating the strengths and weaknesses of several drift detection algorithms on benchmark datasets. All the implementation codes and datasets are available online [1].

**Dataset Setting**: We evaluated the proposed DD index on three groups of datasets (4 toy datasets, 5 Synthetic benchmarks and 4 real-world benchmarks) **Toy Datasets** include a Bernoulli trail sequence dataset with vary success rate, a moon shape drifting dataset, a circle shape drifting dataset and a blob shape drifting dataset. **Synthetic Benchmarks** are SEA generator [35], Rotating Hyperplane generator [36], AGR (AGRAWAL) [37] generator, RTG (Random Tree Generator) and RBF generator. **Real-world Benchmarks** are Electricity Price Dataset (Elec), NAOO Weather Dataset (Weather), Spam Filtering Dataset (Spam), Airline Delay Dataset (Airline).

**Algorithm Setting**: The compared algorithms were ADWIN, DDM, EDDM, HDDM-A, HDDM-W, Page-Hinkley, KSWIN implemented in skmultiflow with their default drift thresholds [10]. We chose these algorithms because they are all implemented in the Python package skmultiflow [10] (version 0.5.3) removing programming quality as a factor in the results. It also makes the experiments easier to reproduce the drift detection accuracy.

1. https://github.com/Anjin-Liu/TKDE2020-DDIndex

### 5.1 Illustration and Demonstration of DD Index

**Experiment 1. Drift Detection on Bernoulli Trail Sequence with Default Drift Threshold**. This experiment was designed to evaluate how various drift detection algorithms perform on binomial trails with their default parameters. First, we generated several pairs of binomial trail sequences $\{\epsilon_i\}_{i=1}^{n_{valid}} \sim Bseq(1,\ 1-\varepsilon,\ n_{valid})$ and $\{\epsilon_j\}_{j=1}^{n_{test}} \sim Bseq(1,\ 1-\varepsilon',\ n_{test})$ to simulate the prequential prediction errors of a learning model, where 0 represents a correct prediction, 1 represents an incorrect prediction, and $\varepsilon$ denotes the expected error rate. Initially, we set $\varepsilon = \varepsilon'$ to measure the general performance and robustness of each algorithm. Then we slightly increased the difference between $\varepsilon$ and $\varepsilon'$ to evaluate detection sensitivity. Thus, the results illustrate the sensitivity and robustness of each drift detection mechanisms in terms of the drift margin, i.e., $|\varepsilon - \varepsilon'|$. When the drift margin is low, i.e., when the drift is hard to detect, we would expect to see the DD Index value at close to 1, and close to 0 when the drift margin is high, i.e., the drift is easy to detect.

To evaluate drift detection algorithms with different sizes of the validation errors and to examine how the estimation of the model's error rate impact the drift detection accuracy. we fixed the validation error rate $\varepsilon$ at 0.15 and changed the test error rate within the predefined error grid $\varepsilon' \in \{0.15, 0.25, \ldots, 0.95\}$. $n_{test}$ was fixed to 200 and $n_{valid}$ was changed within the set $n_{valid} \in \{0, 30, 80, 130, 180\}$.
**Findings and Discussion**. The results of Experiment 1 are plotted in Fig. 2. All algorithms were able to identify the simulated changes in the Bernoulli trail sequences and, overall, the general trends were in line with expectations, in that the DD Index scores decreased as the severity of the drift increased. Further, each algorithm, with their default settings, performed very differently. Two interesting points are worth mentioning.

- Fig. 2 shows the unmatched robustness levels of the different algorithms with their default drift thresholds. With the exception of DDM and EDDM, all other algorithms have a robustness level of 1. However, we do not believe this means they all have the same robustness. Rather, it is more likely that $n_{test} = 200$ is not a large enough number of observations to distinguish a nuanced difference in performance. Another way of thinking about this is that the drift threshold may be set too high/low to detect very frequent drifts. As an example, if a data stream has two slightly different concepts, $A$,$B$ and these two concepts switch approximately every 150 time points (i.e., 1-150 is concept $A$, 151-300 is concept $B$, 301-450 is concept $A$ and so on), an overly robust threshold will mean the detection mechanism cannot identify the drifts.
- Also, from Fig. 2, we find that the validation size must not be 0. Fig. 2 (a) clearly illustrates that ADWIN, HDDM, Page-Hinkley and KSWIN could not detect a drift no matter how large the drift margin is. The fluctuating error rates of DDM and EDDM

indicate some kind of reaction, but, frankly, it is hard to explain their performance. This tells us that if we do not validate a given model $L$ on its training data and then input the validation errors into the drift detection algorithm $H$, the algorithm may lose its power.

**Experiment 2. Drift Detection on Binomial Trail Sequence with DD Index**. In this experiment, we first leverage the DD Index to find the drift thresholds for all drift detection algorithm in terms of a given desired robustness level $\omega$. Intuitively, high sensitiveness on drift could be accompanied by high false alarms. An ideal drift detection has to meet the desired false alarms rates first then as sensitive as possible.

In the experiment result, we would expect to see the robustness level of all algorithms are close to a predefined target value, that is, the robustness level is the same at $\varepsilon' = \varepsilon$. Then we would be able to compare their drift detection sensitivity.

We conducted the experiment according to the control variate method. The objectives, the fixed parameters, the varied parameters and pointers to the results are shown in Table 4. Next, we aligned the robustness levels of each of the algorithms using the search spaces shown in Table 5.

**Findings and Discussion**. The results for Scenarios 1-3 are shown in Fig. 3 and Tables 6, 7, from which we made the following observations:

- Fig. 3 shows the aligned robustness for all evaluated drift detection algorithms. From the results, it is clear that the drift thresholds selected via Algorithm 2 perform as intended. HDDM-W and Page-Hinkley emerge with the most competitive drift detection sensitivity.
- Tables 6 and 7 show us the drift threshold values corresponding to the different DD Index (robustness levels) – the difference between the two being the validation error rates ($\varepsilon = 0.15$ for Table 6 and $\varepsilon = 0.35$ for Table 7). Both tables show a trend that the detection DD Index scores hold true to their indications – that is that less robust thresholds score lower on the scale. ADWIN, for instance, with a DDI of $\omega = 0.99$ scores a $delta = 0.611$ but a $delta = 2.257$ at a DDI of $\omega = 0.8$. The negative correlation between the robustness level and threshold value of DDM and Page-Hinkley sees a lower threshold value for these two methods, which is perfectly in keeping with design because the larger the threshold value, the lower the robustness.
- The last worthy mention is that, as the validation error rates increased from 0.15 to 0.35, the robustness levels decreased for ADWIN, the HDDM family, Page-Hinkley and KSWIN, yet increased for DDM and EDDM. Looking at Table 6 and 7, ADWIN with a $delta = 0.611$ reached a robustness level of $\omega = 0.99$ at $\varepsilon = 0.15$, but only $0.95 \leq \omega \leq 0.99$ at $delta = 0.611$ and $\varepsilon = 0.35$. This phenomenon suggests that error rates can influence the way a drift detection algorithm will react, once again reinforcing our argument that fixing the drift threshold in stone is not the soundest strategy.

### 5.2 Stream Learning on Toy Datasets with Simulated Concept Drift

**Experiment 3. Varying Detection Parameters and Drift Scenarios with Toy Streams**. In this experiment, we focus on investigating how the robustness and sensitivity of drift detection algorithms impacts learning performance. The intuition is to generate a set of data streams with predefined drift frequencies and severities, and then observe the results when incremental learning with drift detection is applied to those streams. To avoid the influence of warning levels, we set both the warning and drift thresholds to the same value so that no warnings are triggered.

We expect to see that algorithms with fixed drift thresholds perform differently on data streams with different drift frequencies and severities. To this end, this experiment has two goals: first, to affirm our contention that drift thresholds should be updated dynamically; second, to gather insights into the conditions and signals that should trigger thresholds to change.

We chose GaussianNB imported from sklearn as the default learning model because it has a fast processing speed and a partial fit option. The minimum training window size $w_{min}$ was set to 200. The same drift detection algorithms were evaluated as in Experiments 1 and 2.

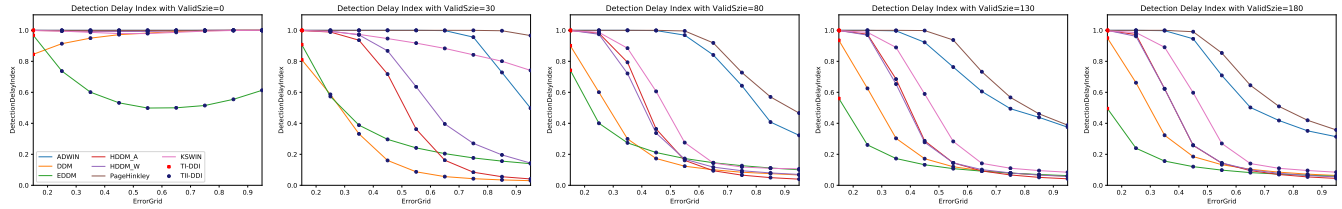
Once the training buffer reached $w_{min}$, we performed cross-validation to generate validation errors and randomly selected $n_{valid} = 80$ errors as initial inputs for the drift detection algorithms. The drift thresholds were selected by the robustness alignment algorithm from within $\omega \in \{0.99, 0.95, \ldots, 0.8\}$.

Concept drifts were simulated via $p(X)$ shifting. A parameter $\delta$ drawn from $\delta \in \{0, 0, 0.2, 0.4, 0.6, 0.8, 1\}$ controlled the severity of the drifts and to simulate different drift frequencies we generated 50 streams of different concept sizes (the length of a stable concept) from $\{300, 800, 1300, 1800, 2300\}$ for each dataset and setting of $\delta$. The mean results are reported for evaluation. Drift frequency is the reciprocal of the concept size.

An example of the toy datasets is shown in Fig. 4. Each has 22 dimensions – 2 informative dimensions and 20 noisy – to increase the learning difficulty by increasing the number of data instances required for the learning model to converge. We did this because if a model can quickly and easily learn a dataset with only a small number of samples, the best option for maintaining good performance would be to simply retrain the model whenever a new data batch, rendering drift adaptation somewhat moot.

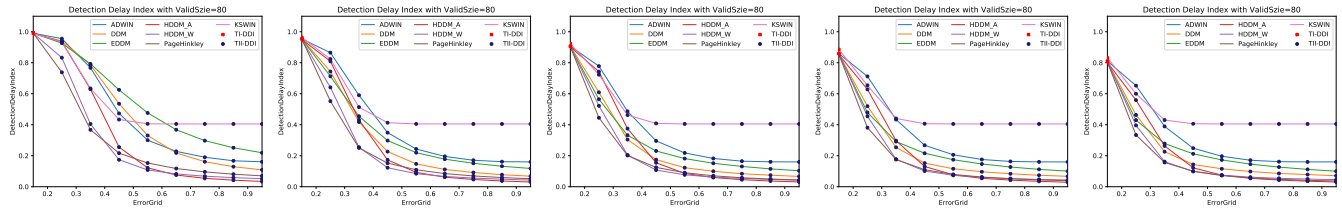
**Findings and Discussion**. The results, shown in Figs. 5-7, are means from 150 runs (50 runs with each of the 3 datasets). Overall, the results provide strong evidence to support our conjecture that a fixed drift threshold performs differently in different drift situations. We had two major findings from this experiment.

- The relationship between drift frequency and accuracy is not neglectable. Fig. 5 shows how remarkably the accuracy of different algorithm changes at different concept sizes. For example, EDDM was the best performer with very frequent drifts, and yet the worst with less frequent drifts.




(a) (b) (c) (d) (e)

Fig. 2. Robustness evaluation on ADWIN, DDM, EDDM, HDDM-A, HDDM-W, Page-Hinkley, KSWIN with their default drift thresholds.




(a) Aligned to 0.99 (b) Aligned to 0.95 (c) Aligned to 0.90 (d) Aligned to 0.85 (e) Aligned to 0.80

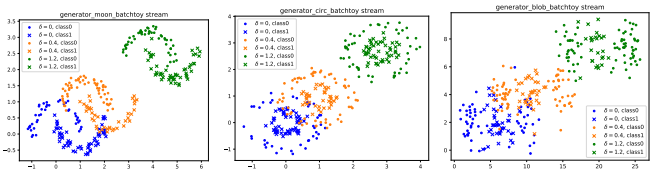
Fig. 3. Robustness evaluation on ADWIN, DDM, EDDM, HDDM-A, HDDM-W, Page-Hinkley, KSWIN with aligned drift threshold.

TABLE 4
Parameter setting scenarios for Experiment 2.

| Objectives | Fixed Parameters | Varying Parameters | Results |
|---|---|---|---|
| Setting 1: To show the efficacy of aligning the robustness of the algorithms with DD Index, and to show the sensitivity of each algorithm once the robustness has been aligned. | $\varepsilon = 0.15$, $n_{valid} = 80$, $n_{test} = 200$ | $\varepsilon' \in \{0.15, \ldots, 0.95\}$, $\omega \in \{0.99, \ldots, 0.8\}$ | Fig. 3 |
| Setting 2: To show the drift threshold values for each algorithm with aligned robustness levels. | $\varepsilon = \varepsilon' = 0.15$, $n_{valid} = 80$, $n_{test} = 200$ | $\omega \in \{0.99, \ldots, 0.8\}$ | Table. 6 |
| Setting 3: To show the drift threshold values of each algorithm with aligned robustness levels and to show how the validation and test error rates affect the threshold selection. | $\varepsilon = \varepsilon' = 0.35$, $n_{valid} = 80$, $n_{test} = 200$ | $\omega \in \{0.99, \ldots, 0.8\}$ | Table. 7 |

TABLE 5
The parameter search space for each algorithm. All search gaps were $\theta_{gap} = 0.001$.

| Algorithm | Parameter Name | $\theta_{MaxRob}$ | $\theta_{MinRob}$ |
|---|---|---|---|
| ADWIN | delta | 0.001 | 10 |
| DDM | out_control_level | 10 | 0.001 |
| EDDM | fddm_outcontrol | 0.001 | 10 |
| HDDM-A | drift_confidence | 0.001 | 1 |
| HDDM-W | drift_confidence | 0.001 | 1 |
| Page-Hinkley | Threshold | 100 | 0.001 |
| KSWIN | alpha | 0.001 | 1 |




(a) (b) (c)

Fig. 4. The toy datasets with simulated concept drift. The feature space is shifted at each time point through a drift severity parameter $\delta$. Subfigure (a) is the moon shape dataset, (b) is the circle shapes, and (c) is the blob shapes.

- For some algorithms, robustness levels have the opposite effect on classification accuracy depending on the drift frequency. Figs. 5 (b) and (e) show that increasing the robustness level caused classification accuracy to decrease in streams with frequent drifts, but to increase in streams with only occasional drifts. ADWIN, the HDDM family, Page-Hinkley and KSWIN all show this same pattern. However, DDM and EDDM do not, which may be because the drift thresholds in these methods consider the real-time standard deviation of the error sequence. This behaviour is highly interesting as a basic proxy of dynamic threshold selection.

Fig. 6 shows the number of concept drifts detected. As the results show, at the same robustness level, the false alarm rate increases as the concept size increases, i.e., as drifts become less frequent. Thus, performance would improve if the drift threshold could adapt to the prevailing drift frequency.

Fig. 7 plots MinMaxScaled classification accuracy at different concept sizes versus different robustness levels. The results show classification accuracy tends to peak along a sliding scale from low robustness to high robustness as the concept size increases. For example, at a concept size of 300, the highest (mean) accuracy among all the algorithm occurred at $\omega = 0.85$, while, with a concept size of 800,

TABLE 6
Robustness alignment with a validation and test error rate of $\varepsilon = \varepsilon' = 0.15$. All algorithms had a DD Index of $> 0.99$ with their default drift threshold, except EDDM $< 0.8$ and DDM between $0.85$ and $0.90$.

| | ADWIN | DDM | EDDM | HDDM-A | HDDM-W | Page-Hinkley | KSWIN |
|---|---|---|---|---|---|---|---|
| Default | 0.002 | 3.000 | 0.900 | 0.001 | 0.001 | 50.000 | 0.005 |
| $\omega = 0.99$ | 0.611 | 5.415 | 0.495 | 0.004 | 0.007 | 11.639 | 0.013 |
| $\omega = 0.95$ | 1.160 | 3.580 | 0.652 | 0.019 | 0.018 | 9.055 | 0.053 |
| $\omega = 0.90$ | 1.601 | 3.026 | 0.740 | 0.029 | 0.028 | 7.797 | 0.120 |
| $\omega = 0.85$ | 1.932 | 2.740 | 0.802 | 0.053 | 0.036 | 6.999 | 0.173 |
| $\omega = 0.80$ | 2.257 | 2.560 | 0.853 | 0.076 | 0.045 | 6.358 | 0.242 |

TABLE 7
Robustness alignment result with validation, test error rates $\varepsilon = \varepsilon' = 0.35$. The purpose of this experiment is to show the impact of the drift threshold on the validation and test error rates. ADWIN, HDDM-A, HDDM-W, Page-Hinkley and KSWIN became less robust as the validation error rates increased, while DDM and EDDM grew more robust.

| | ADWIN | DDM | EDDM | HDDM-A | HDDM-W | Page-Hinkley | KSWIN |
|---|---|---|---|---|---|---|---|
| Default | 0.002 | 3.000 | 0.900 | 0.001 | 0.001 | 50.000 | 0.005 |
| $\omega = 0.99$ | 0.413 | 3.821 | 0.528 | 0.001 | 0.001 | 16.514 | 0.001 |
| $\omega = 0.95$ | 0.892 | 2.947 | 0.631 | 0.008 | 0.002 | 12.854 | 0.004 |
| $\omega = 0.90$ | 1.286 | 2.595 | 0.687 | 0.018 | 0.006 | 11.003 | 0.013 |
| $\omega = 0.85$ | 1.614 | 2.410 | 0.725 | 0.027 | 0.009 | 9.783 | 0.033 |
| $\omega = 0.80$ | 1.910 | 2.279 | 0.753 | 0.037 | 0.013 | 8.842 | 0.054 |

accuracy peaked at $\omega = 0.9$, and at $\omega = 0.95$ with a concept size of 2300. Notably, with large concept sizes, accuracy begins to decrease after a certain robustness level. As Fig. 7 (a) with a concept size of 2300 shows, classification accuracy decreased significantly at $\omega = 0.99$ after peaking at $\omega = 0.95$.

### 5.3 Stream Learning on Benchmark datasets

**Experiment 4. Varying Drift Detection Parameters with Benchmark Datasets**. With this experiment, we investigated whether toy datasets and the benchmark concept drift evaluation datasets share the same accuracy vs. robustness patterns. The basic settings are the same as Experiment 3. The main difference is that we used two groups of benchmark datasets – one group of synthetic datasets with synthetic drift (SynthData-SynthDrift) and one group of real-world datasets with unknown drift (RealData-UnknDrift). We would expect to see similar accuracy vs. robustness patterns with the synthetic datasets as the ones we found in Experiment 3. Additionally, we would like to confirm whether there is always a drift threshold that outperforms the default drift threshold.

**Findings and Discussion**. The learning accuracies with the default and the best drift thresholds appear in Tables 8 and 9. We were not able to discern the accuracy vs. robustness patterns for small concept sizes, since all the synthetic datasets have a large fixed concept size. However, we can see the DD Index scores of the best results vary from dataset to dataset. Although the results do not show exactly what the correlation is, they do show that one exists. And, when comparing the accuracy values for the Default and Best columns, we find that varying the drift thresholds improves the classification results, which proves our contention that the drift threshold should be dynamically adjusted based on the historical frequency and severity of drifts in data stream rather than using a fixed threshold the whole time.

## 6 Conclusions and Future Work

With the DD Index, we can align the drift thresholds across a set of algorithms so that they all have the same robustness level. As a result, threshold selection could be removed as a factor from the comparison, leaving a clear space to reveal the problems or, benefits of the drift detection mechanism itself. A series of experiments confirm that adaptively select drift thresholds is better than using a fixed threshold. Additionally, both validation errors and the frequency of drifts in a stream must be considered for a good threshold selection. These findings pave the way to a more intelligent drift threshold selection paradigm, where drift thresholds dynamically adjust in line with learning performance on the current data stream.

Our next step with the DD Index is to leverage historical drift information in the threshold update process. For example, a learning model built on a small training set may have a very high variance, and, as a result, a drift detection algorithm might raise false alarms more frequently in the history. Efforts to further understand the nature of these relationships will feature in our continued investigations of the correlations between learning models and drift detection algorithms.

## Acknowledgments

This work was supported by the Australian Research Council through the Discovery Project under Grant DP190101733.

## References

[1] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, 2018.
[2] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification," *ACM Computing Surveys*, vol. 50, no. 2, pp. 1–36, 2017.
[3] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, pp. 12–25, 2015.

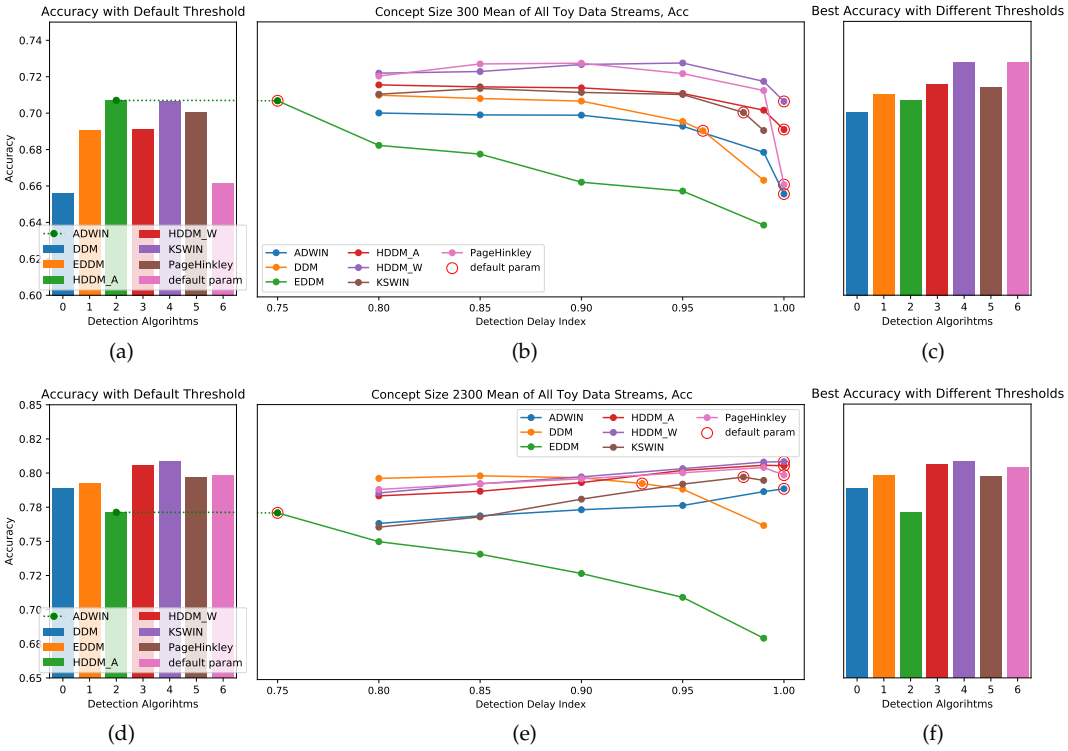



Fig. 5. Learning accuracy with different drift scenarios. Concept size: subfigures (a)-(c) 300 (frequent drift); (d)-(f) 2300 (occasional drift). Drift threshold: (a) & (d) default; (b) & (e) DD Index-aligned (circled observation points indicate the accuracy and robustness levels when using the method's default drift threshold); (c) & (f) highest accuracy among all robustness levels. These results demonstrate that drift threshold selection significantly impacts the accuracy of data stream learning under concept drift.

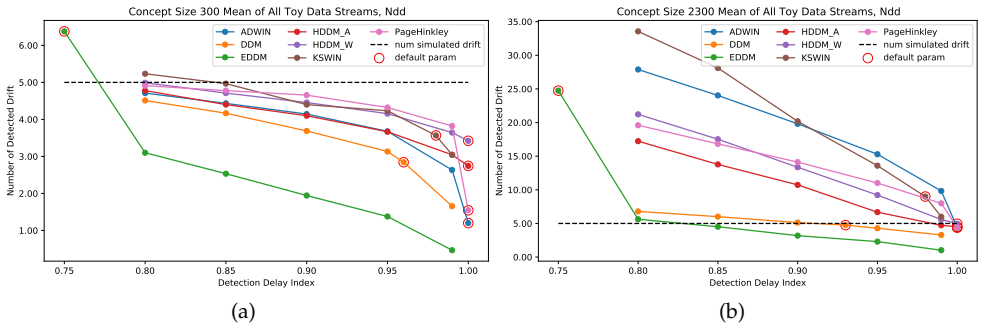



Fig. 6. Number of drifts detected. Subfigure (a) shows the number of drifts detected with the concept size of 300; (b) is 2300. Circles observation points indicate the number of drifts detected using the default drift threshold. The black dashed lines mark the ground truth number of drifts. In these data streams, there were five drifts (six concepts). The results show that, at the same robustness level, the false alarm rate increases as the concept size increases, i.e., as drifts become less frequent. Thus, performance would improve if the drift threshold could adapt to the prevailing drift frequency.

[4] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–37, 2014.

[5] Y. Xu, R. Xu, W. Yan, and P. Ardis, "Concept drift learning with alternating learners," in *Proceedings of the 2017 International Joint Conference on Neural Networks*, 2017, pp. 2104–2111.

[6] L. L. Minku and X. Yao, "Ddd a new ensemble approach for dealing with concept drift," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 4, pp. 619–633, 2012.

[7] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 5, pp. 730–742, 2010.

[8] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proceedings of the Seventeenth Brazilian Symposium on Artificial Intelligence*, vol. 3171. Sao Luis, Maranhao, Brazil: Springer, 2004, pp. 286–295.

[9] M. Pratama, J. Lu, E. Lughofer, G. Zhang, and M. J. Er, "An incremental learning of concept drifts using evolving type-2 recurrent fuzzy neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 5, pp. 1175–1192, 2016.

[10] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, "Scikit-multiflow: A multi-output streaming framework," *J. Mach. Learn. Res.*, vol. 19, no. 1, pp. 2915–2914, 2018.

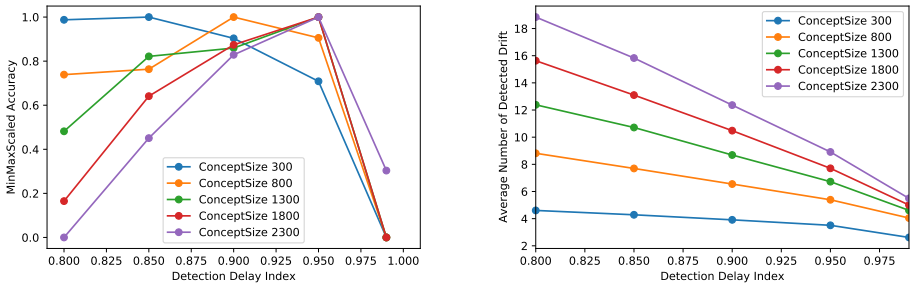
[11] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno, "Early drift detection method," in *Proceedings of the Fourth International Workshop on Knowledge Discovery from Data Streams*, vol. 6, 2006, pp. 77–86.

[12] I. Frias-Blanco, J. d. Campo-Avila, G. Ramos-Jimenes, R. Morales-Bueno, A. Ortiz-Diaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 3, pp. 810–823, 2015.

[13] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1-2, pp. 100–115, 1954.

[14] C. Raab, M. Heusinger, and F.-M. Schleif, "Reactive soft prototype computing for concept drift streams," *Neurocomputing*, 2020.

[15] A. Bifet and R. Gavaldà, "Learning from time-changing data




(a) (b)

Fig. 7. The impacts of concept size and robustness levels on stream learning accuracy. Subfigure (a) shows MinMaxScaled accuracy with different concept sizes versus different robustness levels, calculated as $\mathbf{Acc_{scaled}} = \frac{\mathbf{Acc}-\mathbf{Acc_{min}}}{\mathbf{Acc_{max}}-\mathbf{Acc_{min}}}$. We use MinMaxScaled accuracy to zoom the changes in the accuracy with different robustness level. (b) shows the number of drifts detected at different concept sizes versus different robustness levels. These results show that there are strong correlations between learning accuracy, robustness level and concept sizes (drift frequency).

TABLE 8
Classification accuracy with the ADWIN, DDM, and EDDM detection methods. The Default column shows the classification accuracy using the default drift threshold. The Best and the DD Index (DDI) columns show the best classification accuracy and corresponding the DD Index within the range $\omega \in \{0.99, \ldots, 0.8\}$. The shaded DDI values signal where the Default threshold outperformed the Best threshold. In every case, we see a DDI of either $0.99$ or $0.8$, which suggests that the actual robustness required to align the data steam falls outside the predefined range of $\omega \in \{0.99, \ldots, 0.8\}$

| | ADWIN | | | DDM | | | EDDM | | |
|---|---|---|---|---|---|---|---|---|---|
| | Default | Best | DDI | Default | Best | DDI | Default | Best | DDI |
| AGR | 0.713 | **0.726** | 0.990 | 0.713 | **0.718** | 0.800 | 0.651 | **0.664** | 0.950 |
| HYP | **0.869** | 0.831 | 0.990 | 0.861 | **0.869** | 0.950 | 0.861 | **0.869** | 0.800 |
| RBF | **0.715** | 0.706 | 0.990 | 0.715 | **0.717** | 0.900 | 0.678 | **0.715** | 0.800 |
| RTG | **0.668** | 0.667 | 0.800 | **0.668** | **0.668** | 0.950 | 0.653 | **0.668** | 0.800 |
| SEA | 0.847 | **0.851** | 0.950 | 0.833 | **0.862** | 0.850 | 0.844 | **0.847** | 0.900 |
| Airline | **0.620** | **0.620** | 0.990 | 0.620 | **0.630** | 0.900 | **0.620** | 0.610 | 0.990 |
| Elec | 0.790 | **0.830** | 0.850 | 0.820 | **0.830** | 0.800 | **0.830** | 0.770 | 0.800 |
| Spam | 0.920 | **0.940** | 0.850 | 0.920 | **0.930** | 0.900 | 0.930 | **0.940** | 0.850 |
| Weather | 0.700 | **0.720** | 0.950 | 0.700 | **0.730** | 0.850 | **0.740** | 0.700 | 0.800 |

TABLE 9
Data stream classification accuracy with the HDDM-A, HDDM-W, KSWIN and PageHinkley detection methods.

| | HDDM-A | | | HDDM-W | | | KSWIN | | | PageHinkley | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Default | Best | DDI | Default | Best | DDI | Default | Best | DDI | Default | Best | DDI |
| AGR | 0.724 | **0.731** | 0.950 | **0.725** | 0.724 | 0.990 | **0.720** | **0.720** | 0.990 | 0.714 | **0.725** | 0.950 |
| HYP | **0.869** | **0.869** | 0.990 | **0.865** | **0.865** | 0.990 | 0.861 | **0.869** | 0.990 | **0.869** | 0.861 | 0.990 |
| RBF | 0.715 | **0.716** | 0.950 | **0.699** | 0.698 | 0.990 | 0.694 | **0.715** | 0.990 | 0.715 | **0.716** | 0.990 |
| RTG | **0.668** | **0.668** | 0.990 | **0.663** | **0.663** | 0.990 | 0.657 | **0.660** | 0.990 | **0.668** | 0.665 | 0.990 |
| SEA | **0.861** | 0.859 | 0.990 | 0.859 | **0.865** | 0.850 | 0.848 | **0.851** | 0.900 | 0.856 | **0.867** | 0.990 |
| Airline | **0.630** | **0.630** | 0.990 | **0.620** | **0.620** | 0.990 | 0.610 | **0.620** | 0.990 | **0.630** | 0.620 | 0.990 |
| Elec | 0.830 | **0.840** | 0.800 | 0.830 | **0.840** | 0.800 | 0.810 | **0.820** | 0.950 | 0.780 | **0.840** | 0.800 |
| Spam | 0.920 | **0.930** | 0.950 | 0.920 | **0.930** | 0.800 | 0.920 | **0.930** | 0.850 | 0.920 | **0.940** | 0.850 |
| Weather | 0.720 | **0.730** | 0.850 | 0.730 | **0.740** | 0.800 | 0.670 | **0.710** | 0.850 | 0.700 | **0.730** | 0.850 |

with adaptive windowing," in *Proceedings of the Seventh SIAM International Conference on Data Mining*. Minneapolis, MN, USA: SIAM, 2007, pp. 443–448.

[16] J. Lu, A. Liu, Y. Song, and G. Zhang, "Data-driven decision support under concept drift in streamed big data," *Complex & Intelligent Systems*, vol. 6, no. 1, pp. 157–163, 2020.

[17] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognit.*, vol. 45, no. 1, pp. 521–530, 2012.

[18] Y. Song, J. Lu, A. Liu, H. Lu, and G. Zhang, "A segment-based drift adaptation method for data streams," *IEEE Trans. Neural Netw. Learn. Syst.*, 2021.

[19] Y. Song, J. Lu, H. Lu, and G. Zhang, "Learning data streams with changing distributions and temporal dependency," *IEEE Trans. Neural Netw. Learn. Syst.*, 2021.

[20] N. Lu, J. Lu, G. Zhang, and R. L. De Mantaras, "A concept drift-tolerant case-base editing technique," *Artificial Intelligence*, vol. 230, pp. 108–133, 2016.

[21] A. Liu, Y. Song, G. Zhang, and J. Lu, "Regional concept drift detection and density synchronized drift adaptation," in *Proceedings of the Twenty-sixth International Joint Conference on Artificial Intelligence*, Melbourne, Australia, 2017, pp. 2280–2286.

[22] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39–57, 2017.

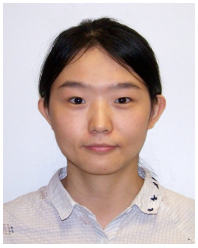
[23] J. Shao, Z. Ahmadi, and S. Kramer, "Prototype-based learning on concept-drifting data streams," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 412–421.

[24] K. Wang, J. Lu, A. Liu, G. Zhang, and L. Xiong, "Evolving gradient boost: A pruning scheme based on loss improvement ratio for

learning under concept drift," *IEEE Trans. Cybern.*, 2021.
[25] S. Yu and Z. Abraham, "Concept drift detection with hierarchical hypothesis testing," in *Proceedings of the Seventeenth SIAM International Conference on Data Mining*. SIAM, 2017, pp. 768–776.
[26] J. Gama and G. Castillo, "Learning with local drift detection," in *Proceedings of the Second International Conference on Advanced Data Mining and Applications*. Springer, 2006, pp. 42–55.
[27] A. Liu, G. Zhang, and J. Lu, "Fuzzy time windowing for gradual concept drift adaptation," in *Proceedings of the Twenty-sixth IEEE International Conference on Fuzzy Systems*. Naples Italy: IEEE, 2017.
[28] S. Xu and J. Wang, "Dynamic extreme learning machine for data stream classification," *Neurocomputing*, vol. 238, pp. 433–449, 2017.
[29] K. Nishida and K. Yamauchi, "Detecting concept drift using statistical testing," in *Proceedings of the Tenth International Conference on Discovery Science*, V. Corruble, M. Takeda, and E. Suzuki, Eds. Sendai, Japan, October 1-4, 2007: Springer Berlin Heidelberg, 2007, pp. 264–269.
[30] A. Liu, J. Lu, F. Liu, and G. Zhang, "Accumulating regional density dissimilarity for concept drift detection in data streams," *Pattern Recognit.*, vol. 76, pp. 256–272, 2018.
[31] G. Boracchi, D. Carrera, C. Cervellera, and D. Maccio, "Quanttree: Histograms for change detection in multivariate data streams," in *Proceedings of the 2018 International Conference on Machine Learning*, 2018, pp. 638–647.
[32] A. Liu, J. Lu, and G. Zhang, "Concept drift detection: Dealing with missing values via fuzzy distance estimations," *IEEE Trans. Fuzzy Syst.*, 2020.
[33] J. Lu, J. Xuan, G. Zhang, and X. Luo, "Structural property-aware multilayer network embedding for latent factor analysis," *Pattern Recognit.*, vol. 76, pp. 228–241, 2018.
[34] A. Liu, G. Zhang, and J. Lu, "Diverse instance-weighting ensemble based on region drift disagreement for concept drift adaptation," *IEEE Trans. Neural Netw. Learn. Syst.*, 2020.
[35] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining*. San Francisco, California: ACM, 2001, pp. 377–382.
[36] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining*. ACM, 2001, pp. 97–106.
[37] R. Agrawal, T. Imielinski, and A. Swami, "Database mining: A performance perspective," *IEEE Trans. Knowl. Data Eng.*, vol. 5, no. 6, pp. 914–925, 1993.

**Anjin Liu** is a Postdoctoral Research Fellow in the Australian Artificial Intelligence Institute, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia. He received the BIT degree (Honour) at the University of Sydney in 2012. His research interests include concept drift detection, adaptive data stream learning, multi-stream learning, machine learning and big data analytics.

**Jie Lu** (F'18) is a Distinguished Professor and the Director of the Australian Artificial Intelligence Institute at the University of Technology Sydney, Australia. She is an IEEE Fellow, an IFSA Fellow and Australian Laureate Fellow. She received her PhD degree from the Curtin University, Australia, in 2000. Her main research expertise is in concept drift, fuzzy transfer learning, decision support systems and recommender systems. She has published six research books and 450 papers in IEEE Transactions and other journals and conferences. She has completed over 20 Australian Research Council discovery grants and other research projects. She serves as Editor-In-Chief for Knowledge-Based Systems (Elsevier) and Editor-In-Chief for the International Journal on Computational Intelligence Systems (Atlantis), has delivered 30 keynote speeches at international conferences, and has chaired 10 international conferences.



**Yiliao Song** (S'17) received a M.S. in probability and statistics in mathematics from the School of Mathematics and Statistics, Lanzhou University, China, in 2015. She is working toward a Ph.D. with the Faculty of Engineering and Information Technology, University of Technology Sydney, Australia. Her research interests include regression, prediction, concept drift and data stream mining. She has published 17 papers related to concept drift, and data stream prediction.

**Junyu Xuan** Dr Junyu Xuan is an ARC Discovery Early Career Researcher Award (DECRA) Fellow and Lecturer of Australia Artificial Intelligence Institute in Faculty of Engineering and IT at the University of Technology Sydney (UTS). His research interests include Machine Learning, Bayesian Nonparametric Learning, Text Mining, Web Mining, etc. He has published over 40 papers in high-quality journals and conferences, including Artificial Intelligence Journal, Machine Learning Journal, IEEE TNNLS, IEEE TFS, IEEE TKDE, IEEE TCYB, ACM TOIS, ACM TIST, ACM Computing Surveys, ICDM, NeurIPS, AAAI, IJCNN, etc. He severed as the PC member for conferences, e.g. NIPS, ICLR, IJCAI and IJCNN.

**Guangquan Zhang** is an Associate Professor and Director of the Decision Systems and e-Service Intelligent (DeSI) Research Laboratory with the Australian Artificial Intelligence Institute, University of Technology Sydney, Australia. He received a Ph.D. degree in applied mathematics from Curtin University of Technology, Australia, in 2001. His research interests include fuzzy modeling in machine learning and data analytics. He has authored five monographs, five textbooks, and 470 papers, including 225 refereed international journal papers. He has been awarded 9 Australian Research Council (ARC) Discovery Project and many other research grants. He was awarded an ARC Queen Elizabeth II Fellowship in 2005. He has served in editorial boards of several international journals, and as a guest editor of eight special issues for IEEE Transactions and other international journals.