A Multi-level Weighted Concept Drift Detection Method

meng Han
North Minzu University
zhiqiang Chen (15720602388@163.com)
North Minzu University https://orcid.org/0000-0002-9517-3478
Hongxin WU
North Minzu University
muhang Li
North Minzu University
xilong Zhang
North Minzu University

Research Article

Keywords: data stream, concept drift, drift detection method, sliding window, multi-level mechanism

Posted Date: September 16th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1306349/v1

A Multi-level Weighted Concept Drift Detection Method

Meng Han, Zhiqiang Chen, Hongxin Wu, Muhang Li, and Xilong Zhang

(*School of Computer Science and Engineering, North Minzu University, Yinchuan 750021*)

Abstract The concept drift detection method is an online learner. Its main task is to determine the position of drifts in the data stream, so as to reset the classifier after detecting the drift to improve the learning performance, which is very important in practical applications such as user interest prediction or financial transaction fraud detection. A new level transition threshold parameter is proposed, and a piecewise weighting mechanism including "Stable Level-Warning Level-Drift Level" is innovatively introduced in concept drift detection. The instances in the window are weighted in levels, and it is applied to the double sliding window. Based on this, a multi-level weighted drift detection method(MWDDM) is proposed. Especially, two variants which are MWDDM_H and MWDDM_M are proposed basd on Hoeffding inequality and Mcdiarmid inequality respectively. Experiments on artificial datasets show that MWDDM can detect abrupt and gradual concept drift faster than any other comparison algorithms, while maintaining a low false positive ratio and false alarm rate. Experiments on real-world datasets show that MWDDM has the highest classification accuracy in most cases.

Keywords data stream; concept drift; drift detection method; sliding window; multi-level mechanism

1 Introduction

In recent years, big data, Internet of Things technology and artificial intelligence have developed rapidly. All walks of life continue to generate a large amount of data, and it has been growing at an alarming rate. These data are called data streams with their own characteristics, such as network data, Weather forecast data, wireless sensor data, financial and power grid data, etc.[1]. Traditionally, machine learning algorithms have assumed a stationary distribution of data. However, the underlying data distribution in an evolving data flow environment may change over time, a phenomenon known as concept drift, which means that the data distribution at time points x and y satisfies $Dx \neq Dy$[2]. In real life, examples of concept drift include changing user interest preferences, monitoring systems, weather forecasting, and financial fraud detection, etc.[3][4][5]. As concept drift occurs, the old learned models in the past will no longer be effective, resulting in a decrease in classification accuracy. Therefore, it becomes crucial to adapt to changing data distributions to ensure high learning performance.

Currently, quite a few adaptive learning algorithms use concept drift detection methods to detect concept drift in evolving data streams. Typically, when the learner detects drift, the classification model is updated or retrained to accommodate concept drift. In the past decades, many concept drift detection methods have been proposed, mainly including statistical-based methods[6][7][8][9][10], window-based methods [11][12][13][14], and sequence analysis-based methods[15]. In the past proposed drift detection methods, many methods either require huge time and memory costs, or cannot detect concept drift as quickly as possible while maintaining low false and false nagative ratios. Based on this, this paper proposes a multi-level weighted concept drift detection method (Multi-level Weighted Drift Detection Method, MWDDM). MWDDM can detect concept drift with low detection delay and keep low false positive ratio and false nagative ratio to detect abrupt and gradual concept drift in data stream.

In this paper, we innovatively introduce a multi-level weighted drift detection mechanism of "stable level-warning level-drift level" in concept drift detection by proposing a threshold parameter for level transition, also, a window mechanism where a long sliding windows overlapps with a short sliding window is used in MWDDM. The algorithm will assign weights to the instances in the two windows during the "stable level", the newest instance in the window will be assigned a larger weight, and the old outdated instances will be assigned a smaller weight, and the difference between the weight value of instances at this time is small. At the same time, the weighted average of correct prediction and the maximum weighted average of correct prediction within the window are calculated at the same time. After

entering the "warning level", the algorithm will increase the difference of weight values between instances within the window to detect drift faster, and update the weighted average of correct prediction and the maximum weighted average of correct prediction. Finally, in the "drift level", MWDDM_H and MWDDM_M use the Hoeffding bound based on Hoeffding's inequality and the Mcdiarmid bound generated by Mcdiarmid's inequality, respectively, to determine if the difference between the weighted average of correct prediction and the maximum weighted average of correct prediction exceeds the value in advance. defined threshold, the occurrence of a concept drift will be reported. At this point, the classifier will be reset for retraining.

2 Related work

2.1 Concept drift

Concept drift is a widespread problem in data stream mining, caused by the change or evolution of streaming data over time. Changes in the underlying distribution cause the feature vectors of arriving instances to no longer reflect class labels. This can negatively impact the reliability and accuracy of classifiers making predictions using the streaming data distribution. Suppose the data stream is in the form of consecutive (x_t, y_t) instances, where t=1, 2, 3⋯., and x_t is a feature vector and *y* belongs to a set with *n* class labels That is, $y \in \{y_1, y_2, \cdots y_n\}$. A prediction obtained by the predictor based on the feature vector x_t at a specific time can be denoted by $\hat{y}_t$. Then the concept drift in the time t_0 to t_1 can be defined as formula (1)[16]. Here p_t represents the joint probability distribution between the feature vector x_t and the target class label y_t at time t. The change of the data flow distribution is the concept drift, which can be reflected in the change of the joint probability distribution.

$$\exists x_t : p_{t_0}(x_t, y_t) \neq p_{t_1}(x_t, y_t) \tag{1}$$

In literatures, concept drift is further described. At a certain moment, $p(x_t, y_t)$ can be obtained from the conditional class concept distribution by formula (2).

$$p(x_t, y_t) = p(y_t)p(x_t|y_t) \tag{2}$$

Then, when the input x_t is predicted, the posterior probability distribution can be obtained according to the Bayesian decision theory as shown in formula (3).

$$p(x_t) = \sum_{t=1}^{n} P(y_t)P(x_t|y_t) \quad \circ$$
$$p(y_t|x_t) = p(y_t)p(x_t|y_t)/p(x_t) \quad where$$
$$y \in \{y_1, y_2, \ldots y_n\} \tag{3}$$

The above is the definition of concept drift in general. In addition, the time step at which a new target concept replaces the old target concept is usually referred to as the duration of concept drift, and the shorter the duration of completing the drift, the faster the drift. Therefore, according to the speed of concept drift, concept drift can be divided into abrupt drift, gradual drift, incremental type and recurring drift[17]. Figure 1 shows the difference between the four types of concept drift.

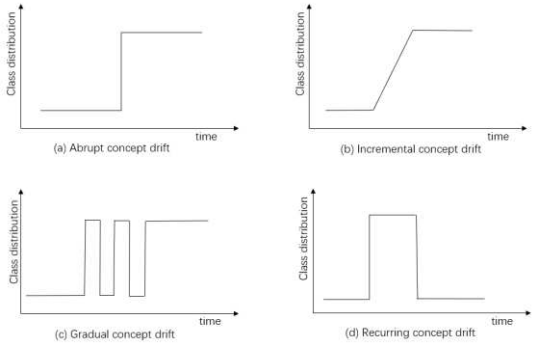


Figure 1 concept drift

2.2 Drift detection methods of different window mechanism

The window mechanism has been widely used to deal with the concept drift problem. They argue that the most recent observed instances are the most useful information, and incrementally estimate changes over time or data windows. The window mechanism defines a window as a short in-memory data structure that can store informative data

or summarize some statistics about model behavior or data distribution in order to describe the current concept. The sliding window mechanism has become one of the most commonly used window mechanisms for drift detection methods. The sliding window is generally composed of a first-in-first-out (FIFO) data structure. After a sliding window defines a window of size n, as a new instance arrives, the oldest instance is discarded[19]. Its mechanism is shown in Figure 2. The FHDDM drift detection method[13] uses a sliding window and Hoeffding inequality to calculate and compare the difference between the maximum prediction accuracy and the current prediction accuracy to detect drift.

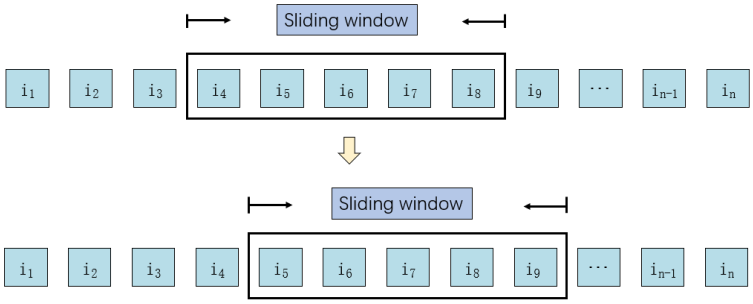


Figure 2 sliding window

The two-window mechanism is mainly divided into separate type, adjacent type and overlapping type. The mechanisms are shown in Figure 3, Figure 4, and Figure 5, respectively.

STEPD[8] uses a statistical test of equal proportions with continuity correction on the data in two separate windows, signaling a warning and drift when a significant difference in accuracy between the recent and old windows is detected. Adaptive Sliding Window (ADWIN)[11] is one of the classic drift detection methods using adjoining double windows. The main idea of ADWIN is: when the average values in the two sub-windows w_1 and w_2 of the latest window W show a sufficiently large difference, and it is inferred that the corresponding predicted values are different, the old window is deleted. The mean of the two windows is defined according to the Hoeffding boundary is greater than the threshold. Based on ADWIN, SEED[12] compares two sub-windows, and when the average of the sub-windows is higher than the selected threshold, the old sub-windows are discarded. It calculates its test statistic using Hoeffding's inequality with Bonferroni correction. The FHDDMS[21] drift detection method uses two superimposed sliding windows to obtain prediction results to detect concept drift.

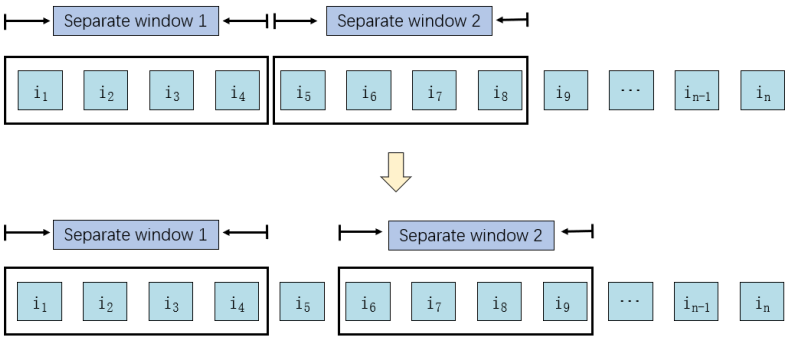


Figure 3 Separate window

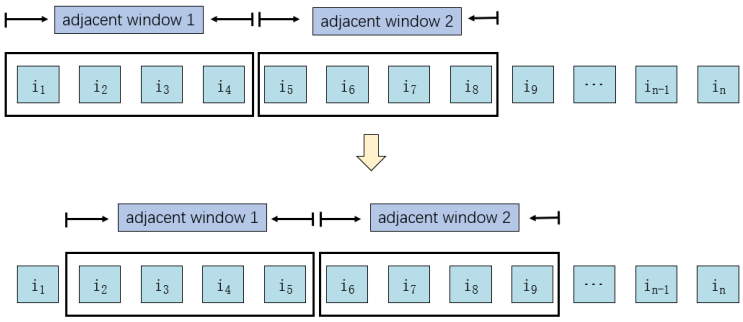


Figure 4 adjacent window

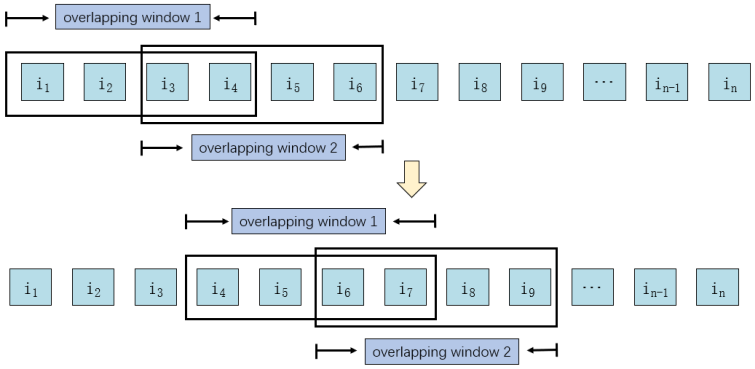


Figure 5 overlapping window

3 Proposed algorithm

In this paper, the abrupt and gradual concept drift in the data stream is taken as the research object, and a level transition threshold parameter is proposed. In the concept drift detection, a multi-level weighted mechanism including "stable level-warning level-drift level" is introduced. The weighting mechanism changes the difference between the weight value of instance. Finally, combined with the double-layer sliding window mechanism, a multi-level weighted Drift Detection Method (MWDDM) is proposed. In addition, two variants of MWDDM are proposed based on Hoeffding's inequality and Mcdiarmid's inequality, respectively: MWDDM_H and MWDDM_M.

3.1 Multi-level weighted mechanism

Many drift detection algorithms based on sliding window have been proposed. ADWIN[11], DDM[6], STEPD[8], FHDDM[13] are all classical drift detection methods using sliding windows. Most of the above algorithms compare the differences in two sub-windows within a window to detect drift.

It is comprehensively found that a shorter sliding window can detect the change of data distribution in the data stream more quickly when abrupt concept drift is occurring, and timely warn a drift signal and make the learner make corresponding changes to adapt to the concept drift. In addition, for gradual concept drift with long drift length, the short sliding window may not be able to adapt to the slowly changing data flow, so a sliding window with a larger length may be more suitable for dealing with gradual concept drift. Based on the above conclusions, this paper uses a combination of sliding windows with overlapping long sliding window and short sliding window to simultaneously adapt to the abrupt and gradual concept drifts in the data stream. Its double sliding window mechanism is shown in Figure 6 below.

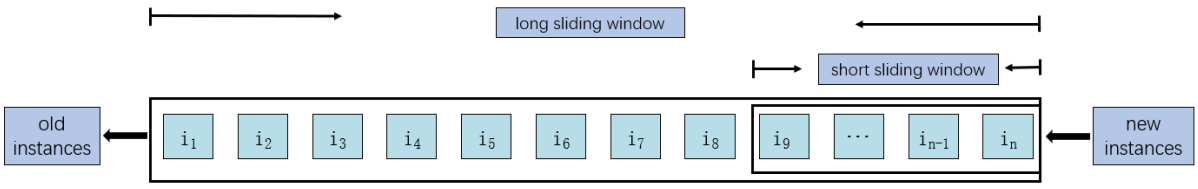


Figure 6 two sliding window mechanism

Also, in a data stream environment, the old instance is considered obsolete or no longer valid. Therefore, incremental learners should be trained using the most recent instances, as the latter are more reflective of the current situation in the context of data stream. Online learning algorithms typically use fading factors or weighting methods to increase the weight of recent instances. This is very important from an adaptive learning perspective, especially when a transition between two concepts in a data stream occurs, i.e. concept drift. Therefore, according to this observation, giving more weight to the newest instances in the window helps to detect concept drift faster.

Based on this, this paper proposes a piecewise weighted concept drift detection mechanism. The drift detection level is divided into three levels, namely "stable level", "warning level" and "drift level".

First, in this algorithm, the data stream is composed of paired instance groups $(\overrightarrow{x_i}, y_i)$, where $\overrightarrow{x_i}$ is the attribute vector and y_i is its corresponding class. For each instance, Naive Bayes Or a classifier such as Hoeffding tree will make a prediction $\hat{y}_i$, and then compare $\hat{y}_i$ with the actual result y_i to decide whether the prediction is correct or no($\hat{y}_i = y_i$), if the current prediction is correct, then at the same time insert 1 into the long sliding window and short sliding window, and insert 0 if the prediction is wrong.

During the "stable level", we weight the instances within two windows. The weighting method used in this paper is a linear weighting method, and its weighting mechanism is shown in Figure 7. As the number of instances increases, the weight value of the newest instance increases linearly compared to the weight value of the old instance. The article defines ω_i as the weight value assigned to an instance. In the linear weighting method, $\omega_{i+1} - \omega_i = diff$, that is, the calculation formula (8) of the weight value of an instance in the window is as follows:

$$\omega_i = 1 + (i-1) * diff \tag{8}$$

During the "stable level", the *diff* is assigned a value of 0.01. Then, this paper defines the weighted average classification prediction accuracy $u_{s\cdot\omega}$ and $u_{l\cdot\omega}$ within the short sliding window and the long sliding window. Its calculation formula is shown in the following formula (4) and formula (5). Among them, $|W_s|$ and $|W_l|$ represent the length of the long sliding window and the short sliding window, respectively.

$$u_{s\cdot\omega} = \frac{\sum_{i=1}^{W_s}(\omega_i * W_i)}{\sum_{i=1}^{W_s}\omega_i} \tag{4}$$

$$u_{l\cdot\omega} = \frac{\sum_{i=1}^{W_l}(\omega_i * W_i)}{\sum_{i=1}^{W_l}\omega_i} \tag{5}$$

Here，$\omega_i = 1 + (i-1) * 0.01$。

At the same time, before the next concept drift is reported, the paper defines the maximum weighted average classification prediction accuracy observed so far in the long sliding window and the short sliding window, respectively, $u_{s\cdot\omega}^{max}$ and $u_{l\cdot\omega}^{max}$, which is calculated as follows,

$$\text{if } u_{s\cdot\omega}^{max} < u_{s\cdot\omega}, \quad \text{then } u_{s\cdot\omega}^{max} = u_{s\cdot\omega}$$
$$\text{if } u_{l\cdot\omega}^{max} < u_{l\cdot\omega}, \quad \text{then } u_{l\cdot\omega}^{max} = u_{l\cdot\omega}$$

Then, in order to judge when the drift detection method enters the "warning level", this paper defines a level transition threshold parameter λ_s and λ_l for the long sliding window and the short sliding window, respectively. The calculation methods are shown in formula (6) and formula (7) respectively.

$$\lambda_s = \frac{u_{s\cdot\omega}}{u_{s\cdot\omega}^{max}} \tag{6}$$

$$\lambda_l = \frac{u_{l\cdot\omega}}{u_{l\cdot\omega}^{max}} \tag{7}$$

When either of the two conditions $\lambda_s > \theta_s$ or $\lambda_l > \theta_l$ is satisfied, the algorithm will enter the "warning level", where θ_s=0.78, θ_l=0.85. Among them, the determination of the pre-defined thresholds θ_s and θ_l will be discussed in detail in the experimental section.

During the "warning level", the algorithm will increase the difference in weight values between the instances in the long and short sliding windows to emphasize the importance of the latest instances so that the detection method can detect concept drift faster. Therefore, after the algorithm enters the "warning level", the weighted average of correct prediction $u_{s\cdot\omega}$ and $u_{l\cdot\omega}$ within the long and short sliding windows are updated to $u_{s\cdot\omega}'$ and $u_{l\cdot\omega}'$, similarly, the maximum weighted average of correct prediction $u_{s\cdot\omega}^{max}$ and $u_{l\cdot\omega}^{max}$ will also be updated to $u_{s\cdot\omega}^{max\prime}$and $u_{l\cdot\omega}^{max\prime}$. Its calculation formula is as follows.

$$u_{s\cdot\omega}{}' = \frac{\sum_{i=1}^{W_s}(\omega_i * W_i)}{\sum_{i=1}^{w_s}\omega_i} \tag{9}$$

$$u_{l\cdot\omega}{}' = \frac{\sum_{i=1}^{W_l}(\omega_i * W_i)}{\sum_{i=1}^{w_l}\omega_i} \tag{10}$$

Here， $\omega_i = 1 + (i-1) * 5$。

The maximum weighted average of correct prediction within the long and short sliding Windows $u_{s\cdot\omega}^{max\prime}$ and$u_{l\cdot\omega}^{max\prime}$are updated as follows

$$\text{if } u_{s\cdot\omega}^{max\prime} < u_{s\cdot\omega}{}', \quad \text{then } u_{s\cdot\omega}^{max\prime} = u_{s\cdot\omega}{}'$$
$$\text{if } u_{l\cdot\omega}^{max\prime} < u_{l\cdot\omega}{}', \quad \text{then } u_{l\cdot\omega}^{max\prime} = u_{l\cdot\omega}{}'$$

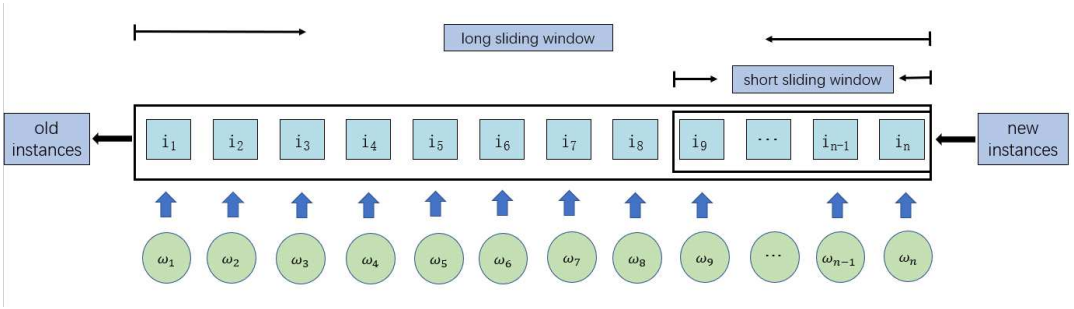


Figure 7 Weighted mechanism

Finally, in the "drift level", MWDDM_H and MWDDM_M are determined by the Hoeffding and Mcdiarmid bounds generated by the Hoeffding inequality and Mcdiarmid inequality respectively. If the difference between the maximum weighted average of correct prediction and the weighted average of correct prediction in the long and short sliding windows is greater than the pre-defined threshold, the occurrence of a concept drift will be reported. At this time, the classifier will be reset to retrain to adapt to the new data distribution

The Hoeffding inequality used by MWDDM_H is shown in Theorem 1 below[15].

Theorem1 Hoeffding inequality

Let $X_1, X_2, ..., X_n$ be n independent random variables， $X_i \in [a_i, b_i], i \in \{1,2,3 \ldots\ldots n\}$。The difference between the empirical means $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, for any ε, there is the following formula (11):

$$p_r\{E[f] - f \geq \varepsilon_M\} \leq \exp\left(-\frac{2 * n^2 * \varepsilon_M^2}{\Sigma_{i=1}^{n}(b_i - a_i)^2}\right) \tag{11}$$

According to this theorem, considering the average $\bar{X}$ which can be at most δ at a given significance level, the estimated error ε_δ, that is, the Hoeffding bound is shown in Equation (12):

$$\varepsilon_\delta = \sqrt{\frac{1}{2n} In \frac{1}{\delta}} \tag{12}$$

Therefore, MWDDM_H defines two thresholds $\varepsilon_{s\cdot H}$ and $\varepsilon_{l\cdot H}$ for the long and short sliding windows respectively, and the calculation formulas are shown in formula (13) and formula (14) respectively;

$$\varepsilon_{s\cdot H} = \sqrt{\frac{1}{2 * W_s} In \frac{1}{\delta_H}} \tag{13}$$

$$\varepsilon_{l\cdot H} = \sqrt{\frac{1}{2 * W_l} In \frac{1}{\delta_H}} \tag{14}$$

MWDDM_H defines the difference between the maximum weighted average of correct prediction and the

current weighted average of correct prediction in the long and short sliding windows as Δ_s and Δ_l, where $\Delta_s = u_{s\cdot\omega}^{max} - u_{s\cdot\omega}$ and $\Delta_l = u_{l\cdot\omega}^{max} - u_{l\cdot\omega}$. Then, when Δ_s is greater than the pre-defined threshold $\varepsilon_{s\cdot H}$ or ε_l is greater than the pre-defined threshold $\Delta_{l\cdot H}$, when either condition is satisfied, the occurrence of concept drift will be reported.

The Mcdiarmid inequality used by MWDDM_M is shown in Theorem 2.

Theorem 2 McDiarmid's inequality

Let $X_1, X_2, ..., X_n$ be n independent random variables that all take values in the set X. Furthermore, let $f: X^n \rightarrow R$ $X_1, ..., X_n$, we have $\forall i, \forall x_1, ..., x_n, x_i' \in X$, which is shown as formula (15)。

$$|f(x_1, \dots x_i, \dots x_n) - f(x_1, \dots x_i' \dots x_n)| \leq c_i \tag{15}$$

This means that replacing x_i with some arbitrary value changes the function f at most c_i . For all ε_M>0, we have formula (16).

$$p_r\{E[f] - f \geq \varepsilon_M\} \leq \exp\left(-\frac{2\varepsilon_M^2}{\Sigma_{\mathrm{i}=1}^{n} c_i^2}\right) \tag{16}$$

Finally, given a confidence δ_M, the obtained ε_M, the Mcdiarmid bound, is shown in formula (17).

$$\varepsilon_M = \sqrt{\frac{In\left(\frac{1}{\delta_M}\right)\Sigma_{\mathrm{i}=1}^{n} v_i^2}{2}} \tag{17}$$

Therefore, MWDDM_M defines two thresholds $\varepsilon_{s\cdot M}$ and $\varepsilon_{l\cdot M}$ for the long and short sliding windows respectively, and the calculation formulas are shown in formula (18) and formula (19). Here, $v_i = \frac{w(i)}{\sum_{i=1}^{n} w(i)}$。

$$\varepsilon_{s\cdot M} = \sqrt{\frac{In\left(\frac{1}{\delta_M}\right)\Sigma_{\mathrm{i}=1}^{\mathrm{W_s}} v_i^2}{2}} \tag{18}$$

$$\varepsilon_{l\cdot M} = \sqrt{\frac{In\left(\frac{1}{\delta_M}\right)\Sigma_{\mathrm{i}=1}^{\mathrm{W_l}} v_i^2}{2}} \tag{19}$$

MWDDM_M still uses the difference between the maximum weighted average of correct prediction and the current weighted average of correct prediction in the same long and short sliding window as in MWDDM_H, namely Δ_s and Δ_l, where $\Delta_s = u_{s\cdot\omega}^{max} - u_{s\cdot\omega}$ and $\Delta_l = u_{l\cdot\omega}^{max} - u_{l\cdot\omega}$. Then, when Δ_s is greater than the pre-defined threshold $\varepsilon_{s\cdot M}$ or Δ_l is greater than the pre-defined threshold $\varepsilon_{l\cdot M}$, when either condition is satisfied, the occurrence of concept drift will be reported.

3.2 Multi-level weighted drift detection method(MWDDM)

Based on the piecewise weighting mechanism proposed above, this paper will analyze the predictions produced by the learner and store them in a two-layer sliding window, and then apply a decision model to try to detect changes in the data distribution and indicate the occurrence of concept drift.

Specifically, given a set of pairs of instances $(\overrightarrow{\mathrm{x_i}}, \mathrm{y_i})$, where $\overrightarrow{\mathrm{x_i}}$ is an attribute vector and $\mathrm{y_i}$ is its corresponding class, for each instance, the base learner will make a prediction $\hat{y}_i$, and then compare with the actual result $\mathrm{y_i}$ to decide whether the prediction is correct or not ($\hat{y}_i = y_i$). The information of the prediction results is stored in the sliding window for the detection model to use. In general, most existing drift detectors analyze the classification accuracy (error rate) and its corresponding standard deviation by predicting the results, and find differences within different windows. Different drift detection methods use different strategies or statistics to monitor the performance of the base classifier and decide when concept drift occurs.

Based on the PAC learning model, MWDDM assumes that as long as the sample distribution is stationary, the error rate will decrease when the number of samples increases, that is, the distribution accuracy tends to increase. Therefore, an increase in error rate or a decrease in classification accuracy indicates a change in the data distribution. Immediately thereafter, the learning performance of existing learners is likely to be degraded. With this idea, the classification

accuracy rate (or error rate) of the classifier can be used to reflect the data distribution changes in the current data stream. Specifically, this paper uses the superimposed long and short sliding windows to obtain the classification prediction results. Based on Hoeffding's inequality and Mcdiarmid's inequality, two variants of the algorithm are proposed, namely MWDDM_H and MWDDM_M.

The specific flow of MWDDM is shown in Algorithm 1 below. Lines 1-3 of the algorithm indicate that the window size of the two sliding windows will be initialized, and the parameter values in the algorithm will be assigned, and then $\varepsilon_{s\cdot H}$、 $\varepsilon_{l\cdot H}$ and $\varepsilon_{s\cdot M}$、 $\varepsilon_{l\cdot M}$ are calculated for MWDDM_H and MWDDM_M respectively. Lines 4-7 of the algorithm indicate whether the instances in the window are full, and if so, discard the oldest instance and insert the newest instance. Lines 8-13 of the algorithm indicate that in the "stable level", the weighted average of correct prediction $u_{s\cdot\omega}$、$u_{l\cdot\omega}$ within the window are calculated, and $u_{s\cdot\omega}^{max}$ and $u_{l\cdot\omega}^{max}$ are updated. Lines 14-22 of the algorithm indicate that it will judge whether the algorithm has entered the "warning level", and if so, update weighted average of correct prediction $u_{s\cdot\omega}$、$u_{l\cdot\omega}$ to $u_{s\cdot\omega}{}'$、$u_{l\cdot\omega}{}'$, and update to obtain the maximum weighted average of correct prediction $u_{s\cdot\omega}^{max\prime}$、$u_{l\cdot\omega}^{max\prime}$. Calculate the difference Δ_s *and* Δ_l between the maximum weighted average of correct prediction and the current weighted average of correct prediction. Lines 24-29 of the algorithm indicate that during the "drift level", MWDDM_H and MWDDM_M will determine whether Δ_s and Δ_l are greater than a pre-defined threshold, and if so, will report the occurrence of a drift and reset the classifier for retraining.

Algorithm 1 MWDDM

1:	$n_s = \|W_s\|, n_l = \|W_l\|$ $MWDDM\_H:\ \delta_H = delta$	/ initialize the window size
2:	$MWDDM\_M: \delta_M = delta$ $d = diff,\ \theta_s = 0.78,\ \theta_l = 0.85$	/ initialize parameter values
3:	$MWDDM\_H: \varepsilon_{s\cdot H} = \sqrt{\frac{1}{2 * \mathrm{W_s}} In \frac{1}{\delta_H}}$, $\varepsilon_{l\cdot H} = \sqrt{\frac{1}{2 * \mathrm{W_l}} In \frac{1}{\delta_H}}$ $MWDDM\_M: \varepsilon_{s\cdot M} = \sqrt{\frac{In\left(\frac{1}{\delta_M}\right) \Sigma_{\mathrm{i}=1}^{\mathrm{W_s}} v_i^2}{2}}$, $\varepsilon_{l\cdot M} = \sqrt{\frac{In\left(\frac{1}{\delta_M}\right) \Sigma_{\mathrm{i}=1}^{\mathrm{W_l}} v_i^2}{2}}$	/calculate $\varepsilon_{s\cdot H}$、 $\varepsilon_{l\cdot H}$ /calculate $\varepsilon_{s\cdot M}$、 $\varepsilon_{l\cdot M}$
4:	**if** Win.size() = n **then**	/if window is full
5:	Win.drop()	/drop oldest instances
6:	**end if**	
7:	Win.push()	/insert newest instances
8:	**/*stable level*/**	
9:	$u_{s\cdot\omega} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{W_s}} (\omega_i * \mathrm{W_i})}{\sum_{\mathrm{i}=1}^{\mathrm{w_s}} \omega_i}$ $u_{l\cdot\omega} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{W_l}} (\omega_i * \mathrm{W_i})}{\sum_{\mathrm{i}=1}^{\mathrm{w_l}} \omega_i}$	/calculate $u_{s\cdot\omega}$、$u_{l\cdot\omega}$
10:	**if** $u_{s\cdot\omega}^{max} < u_{s\cdot\omega}$, **then**	/update $u_{s\cdot\omega}^{max}$
11:	$u_{s\cdot\omega}^{max} = u_{s\cdot\omega}$	
12:	**if** $u_{l\cdot\omega}^{max} < u_{l\cdot\omega}$, **then**	/update $u_{l\cdot\omega}^{max}$
13:	$u_{l\cdot\omega}^{max} = u_{l\cdot\omega}$	
14:	**if** ($\lambda_s = \frac{u_{s\cdot\omega}}{u_{s\cdot\omega}^{max}} > \theta_s$ or $\lambda_l = \frac{u_{l\cdot\omega}}{u_{l\cdot\omega}^{max}} > \theta_l$) **then**	/judge to enter "warning level"
15:	**/*warning level*/**	
16:	$u_{s\cdot\omega}{}' = \frac{\sum_{\mathrm{i}=1}^{\mathrm{W_s}} ((1 + (i-1) * 0.01) * \mathrm{W_i})}{\sum_{\mathrm{i}=1}^{\mathrm{w_s}} 1 + (i-1) * 0.01}$	/update $u_{s\cdot\omega}$ to $u_{s\cdot\omega}{}'$

17:	$u_{l\cdot\omega}{}' = \frac{\sum_{i=1}^{W_l}((1+(i-1)*5)*W_i)}{\sum_{i=1}^{W_l}1+(i-1)*5}$	/update $u_{l\cdot\omega}$to $u_{l\cdot\omega}{}'$
18:	**if** $u_{s\cdot\omega}^{max\prime} < u_{s\cdot\omega}{}'$, **then**	/update $u_{s\cdot\omega}^{max\prime}$
19:	$u_{s\cdot\omega}^{max\prime} = u_{s\cdot\omega}{}'$	
20:	**if** $u_{l\cdot\omega}^{max\prime} < u_{l\cdot\omega}{}'$, **then**	/update $u_{l\cdot\omega}^{max\prime}$
21:	$u_{l\cdot\omega}^{max\prime} = u_{l\cdot\omega}{}'$	
22:	$\Delta_s = u_{s\cdot\omega}^{max'} - u_{s\cdot\omega}{}'$、$\Delta_l = u_{l\cdot\omega}^{max'} - u_{l\cdot\omega}{}'$	/calculate Δ_s、Δ_l
23:	**/*drift level*/**	
24:	**MWDDM_H:**	
25:	**if** $(\Delta_s > \varepsilon_{s\cdot H}$ *or* $\Delta_l > \varepsilon_{l\cdot H})$	/judge if a drift has occurred
26:	**MWDDM_M:**	
27:	**if** $(\Delta_s > \varepsilon_{s\cdot M}$ *or* $\Delta_l > \varepsilon_{l\cdot M})$	
28:	`return true`	/if yes, report a drift happening
29:	Win=[ ], $u_{s\cdot\omega}^{max}$、$u_{l\cdot\omega}^{max}$=0, $u_{s\cdot\omega}^{max\prime}$、$u_{l\cdot\omega}^{max\prime} = 0$	/reset the window

4 Experiments

In this section, in order to verify the effectiveness of the MWDDM proposed in this paper, this paper conducts experimental evaluations on both artificial datasets and real-world datasets. The experimental platform is Massive Online Analysis (MOA) framework[23]. This paper compares MWDDM with the latest drift detection algorithms, including DDM[6], EDDM[7], RDDM[22], FHDDM[13], FHDDMS[21], MDDM[14], and HDDM[20]. Our experiments are performed on the processor Intel(R) Core(TM) i5-4200H CPU @ 2.80GHz and 8gb RAM. In Section 4.1, this paper introduces the evaluation metrics used in the experiment, introduces the dataset used in the experiment in Section 4.2, and presents and analyzes the experimental results in Section 4.3.

4.1 Evaluation metrics

Currently, the evaluation metric for detecting concept drift in a data stream is detection delay. When drift occurs at a certain moment, the drift detection algorithm cannot detect it immediately, that is, there is usually a delay in drift detection. Therefore, in order to effectively evaluate the timeliness of drift detection, Detection Delay (DD) is introduced to describe the number of instances between the actual position of the drift and the detected position to evaluate the timeliness of the algorithm.

In addition, the True Positive Ratio (TPR), False Positive Ratio (FPR) and False Negative Ratio (FNR) are defined according to the maximum detection delay Δd introduced in [13]. The maximum detection delay Δd is a threshold that determines how far a detected drift is allowed to be from the true position of the drift to be considered a true drift. In this paper, the maximum detection delay Δd is set to 250 in the dataset containing abrupt concept drift, and 1000 in the dataset containing gradual concept drift. The definitions of false positive ratio and false nagative ratio are as follows:

True Positive Ratio (TPR): Suppose the moment when the drift occurs is T, then the number of drifts detected in the interval [T, T+Δd] is regarded as the number of correct detections (TP) is the correct detection The number of drifts. In this paper, the true positive ratio TPR is defined as the number of correct detections /the total number of drifts in the interval [T, T+Δd].

False Positive Ratio (FPR): Let the moment when the drift occurs be T, if the drift detector detects a drift that exceeds the acceptable detection interval, it will falsely issue a drift alarm. The number of drifts detected outside the interval [T, T+Δd] is regarded as the number of false positives (FP), and the false positive rate FPR is defined as FP/(FP+TP).

False Negative Ratio (FNR): False negative means that the drift detector mistakenly ignores the drift that occurs in the interval [T, T+Δd], and the number of missed drifts is the number of false negatives (FN), define the false nagative

ratio FNR as FN/(FN+TP).

Finally, classification accuracy (Accuracy), memory usage (model cost) and running time (evaluation time) in real-world datasets are also important evaluation metrics.

4.2 Datasets

There have been many studies evaluating the proposed algorithms on artificial datasets of specific types of concept drift. One of the advantages of artificial datasets is to know details such as where the drift is. The real-world dataset used in this paper is as follows. They are frequently used in the fields of concept drift detection and adaptive learning in data streams. All artificial and real-world datasets used in this paper are summarized in Table 1.

SINE: This dataset contains abrupt drift. It takes two properties (*x* and *y*) that are uniformly distributed in [0, 1]. Additionally, the dataset is classified using the following function y = sin(x). Therefore, any instances below the curve are classified as positive, while others are negative until the first drift occurs. The dataset contains a total of 100,000 instances, and every 20,000 instances, a drift occurs, and then reverse classification occurs. The dataset contains a total of four drifts at 20,000, 40,000, 60,000, and 80,000 instances with 10% noise.

MIXED: This dataset contains abrupt drift. The dataset has two numerical properties x and y uniformly distributed in [0,1], and two boolean properties v and w. An instance is classified as positive requires at least two or three of the following conditions: v, w, $y < 0.5 + 0.3 * \sin(2\pi x)$. The dataset flips the classification after one drift, and drifts every 20,000 instances with 10% noise.

CIRCLES: This dataset contains gradual drift, which has two continuous attributes x and y. The four circle equations represent 4 different concepts, the instances inside the circle are classified as positive, and the instances outside the circle are negative, a total of two categories. Drift is created by gradually changing the equation of the circle at the drift point. The dataset contains a total of 100,000 instances, and a gradual drift occurs every 25,000 instances, with 10% noise.

LED: This dataset contains gradual drift. The goal of this dataset is to predict numbers on a seven-segment display, where each number has a 10% chance of being displayed. This dataset has 7 class-related attributes and 17 unrelated attributes. Simulate concept drift by exchanging related properties. The dataset contains a total of 100,000 instances, and a gradual drift occurs every 25,000 instances, with 10% noise.

ELECTRICITY: It contains 45,312 instances with 8 input attributes, recorded every half hour for two years by the NSW Electricity Company in Australia. The classifier must predict the rise (Up) or the fall (Down) of the electricity price. Concept drift may stem from changes in consumption habits or emergencies.

FOREST COVERTYPE: It consists of 54 attributes and 581,012 instances describing 7 forest cover types at 30 × 30 m cells obtained from the United States Forest Service (USFS) information system for the Roosevelt National Forest in Northern Colorado 4 wilderness areas.

POKERHAND: It consists of 1,000,000 instances, where each instance is an example of five cards drawn from a standard 52-card deck. Each card is described by two attributes (suit and rank), for a total of ten predicted attributes.

Table 1 Summary of datasets

	datasets	instances	features	class labels	num of drifts	concepts	drift type
artificial datasets	SINE	100000	2	2	4	20000	abrupt
	MIXED	100000	4	2	4	20000	abrupt
	LED	100000	24	10	3	25000	gradual
	CIRCLES	100000	2	2	3	25000	gradual
real-world datasets	ELECTRICITY	45312	8	2	\	\	unknown
	POKER HAND	1000000	2	10	\	\	unknown
	FOREST COVERTYPE	581012	54	7	\	\	unknown

4.3 Experimental results and analysis

(1) Experiments of Parameter analysis

First, the determination of parameters θ_s and θ_l in MWDDM is experimentally analyzed. If the artificial data and λ_s and λ_l of all instances in the real-world dataset are collected, the number is too large to clearly show the trend of λ_s and λ_l changing with the instances. Therefore, this paper collects the parameter values of 1000 instances before the first drift point and 1000 instances after the first drift point in all artificial datasets, and uses Naive Bayes and Hoeffding trees as classifiers for experiments respectively. . Figures 8 and 10 show the changing trends of the parameter value λ_s around the first drift point in the abrupt drift dataset and the gradual drift dataset, respectively. Figure 9 and Figure 11 show the changing trend of the parameter value λ_l in the abrupt drift dataset and the gradual drift dataset near the first drift point, respectively.

It can be seen from Figure 8 and Figure 10 that the parameter value λ_s of the algorithm is 0.78-1.0 in most cases, whether in the abrupt drift dataset (SINE, MIXED) or the gradual drift dataset (CIRCLES, LED). The range of λ_s fluctuates continuously, and when instance=20000 near the drift point, the λ_s value drops sharply from 0.78 to about 0.4. This means that when $7.8 \leqslant \lambda_s \leqslant 1$, the algorithm speculates that the data distribution in the data stream is in the "stable level", and when $\lambda_s < 0.78$, the algorithm may experience conceptual drift and enter the "warning level". Therefore, this paper sets the parameter value θ_s in the algorithm to 0.78. In addition, it can be seen from Figure 9 and Figure 11 that the variation trend of λ_l is constantly changing in the range of 0.85-1.0 in most cases, while in the range near the drift point, the value of λ_s drops sharply from 0.85 to around 0.7, and this change is particularly evident in Figure 12. At the same time, what is shown in the figure is the change trend of λ_l value under the long sliding window in the gradual dataset, which is particularly important for detecting gradual concept drift. Therefore, this paper sets the value of θ_l to 0.85.

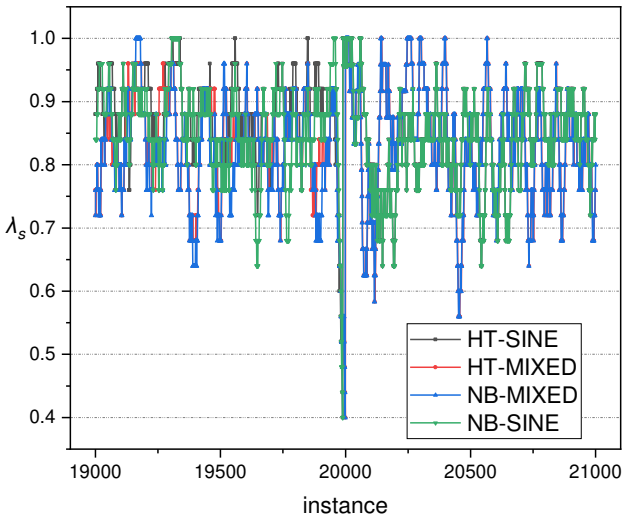


Figure 8 λ_s in datasets with abrupt drift

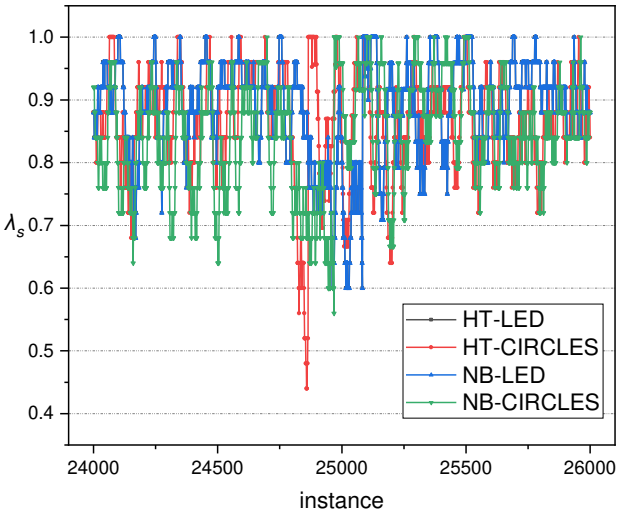


Figure 10 λ_s in datasets with gradual drift

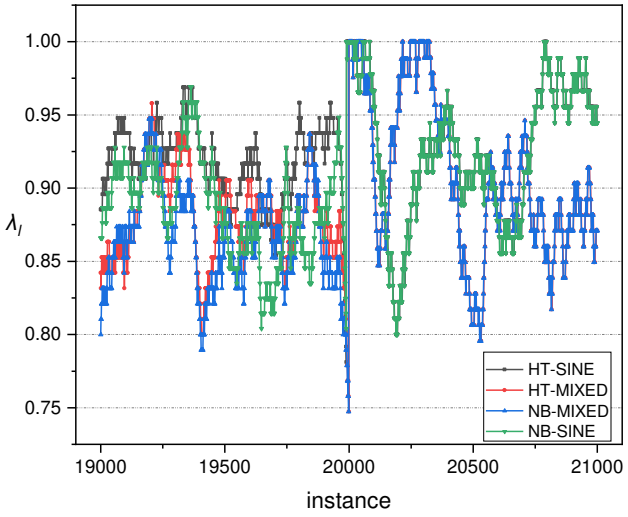


Figure 9 λ_l in datasets with abrupt drift

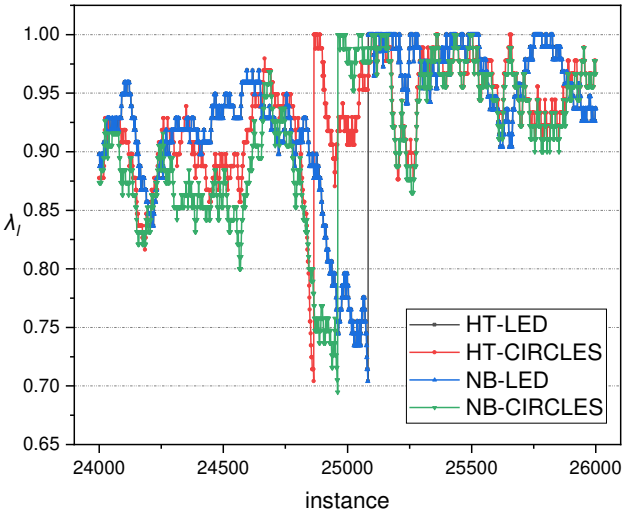


Figure 11 λ_l in datasets with gradual drift

(2) Experiments of drift detection performance

In this paper, the proposed algorithm MWDDM and other comparison algorithms are tested on artificial datasets with catastrophic concept drift, namely SINE and MIXED, and artificial datasets with gradual concept drift, namely CIRCLES and LED. Experiments were carried out on the NB and Hoeffding tree (HT) as learners respectively, and the drift detection performance of the algorithm was summarized and analyzed.

In this paper, the maximum detection delay Δd is set to 250 on datasets with abrupt drift (SINE, MIXED), and 1000 on datasets with gradual drift (CIRCLES, LED), because the drift width of gradual drift is considered in this paper. If Δd is set too small, it will lead to a higher false positive ratio.

Table 2 shows the drift detection performance of MWDDM_H and MWDDM_M and other comparative algorithms using Naive Bayes and Hoeffding trees as learners, respectively, in the LED artificial dataset, which contains gradual-type concept drift. Regardless of whether Naive Bayes or Hoeffding tree is used as the learner, MWDDM_H and MWDDM_M achieve the lowest detection delay among all algorithms, followed by MDDM, FHDDM, FHDDMS and HDDM_W. Specifically, when using Naive Bayes as the learner, the detection delay of MWDDM_H is reduced by 9.07 compared with MDDM_E, and it is 45.73 compared with FHDDMS. Both EDDM and DDM have the highest detection delay. In addition, EDDM has the highest false positive ratio and false nagative ratio.

Table 2 drift detection performance in LED

	detector	DD	TPR	FPR	FNR
NB	MWDDM_H	215.20	1	0	0
	MWDDM_M	215.22	1	0	0
	FHDDMS	260.93	1	0	0
	FHDDM	260.93	1	0	0
	MDDM_A	257.20	1	0	0
	MDDM_E	224.27	1	0	0
	MDDM_G	224.27	1	0	0
	HDDM_W	287.40	1	0	0
	HDDM_A	310.20	1	0	0
	EDDM	741.67	0.33	0.67	0.67
	RDDM	332.93	1	0	0
	DDM	432.73	1	0	0
HT	MWDDM_H	210.27	1	0	0

	MWDDM_M	215.20	1	0	0
	FHDDMS	256.00	1	0	0
	FHDDM	261.00	1	0	0
	MDDM_A	260.93	1	0	0
	MDDM_E	224.27	1	0	0
	MDDM_G	224.27	1	0	0
	HDDM_W	287.40	1	0	0
	HDDM_A	277.73	1	0	0
	EDDM	876.75	0.27	0.87	0.73
	RDDM	338.13	1	0	0
	DDM	444.00	1	0	0

Table 3 shows the drift detection performance on the artificial dataset CIRCLES, which contains gradual-type concept drift. Among all algorithms, MWDDM has the lowest detection delay in the dataset, followed by MDDM and HDDM_W. Compared with the next best performance of MDDM_E, MWDDM_H reduces the detection delay by 6.87, and compared with FHDDMS, it reduces by 60.40. Compared with MDDM_E, the reduction in detection delay reaches 8.26, and compared with FHDDMS, it reaches 61.73. FHDDMS, FHDDM and MDDM_A achieved the lowest false positive ratio, and EDDM achieved the highest false positive ratio. Meanwhile, EDDM and DDM have the highest false nagative ratio. In addition, the MWDDM algorithm in this paper also has false detections to some extent.

Table 3 drift detection performance in CIRCLES

	detector	DD	TPR	FPR	FHR
NB	MWDDM_H	120.20	1	0.23	0
	MWDDM_M	118.87	1	0.23	0
	FHDDMS	180.60	1	0	0
	FHDDM	180.60	1	0	0
	MDDM_A	186.67	1	0	0
	MDDM_E	127.07	1	0.1	0
	MDDM_G	127.07	1	0.1	0
	HDDM_W	157.00	1	0.26	0
	HDDM_A	237.73	1	0.13	0
	EDDM	956.00	0.07	0.99	0.93
	RDDM	403.53	1	0.36	0
	DDM	640.70	0.67	0.37	0.33
HT	MWDDM_H	54.60	1	0.10	0
	MWDDM_M	52.67	1	0.10	0
	FHDDMS	69.80	1	0	0
	FHDDM	77.27	1	0	0
	MDDM_A	77.27	1	0	0
	MDDM_E	68.53	1	0.05	0
	MDDM_G	61.40	1	0.13	0
	HDDM_W	76.27	1	0.13	0
	HDDM_A	73.67	1	0.18	0
	EDDM	316.00	0.13	0.99	0.87
	RDDM	272.13	1	0.05	0
	DDM	429.33	1	0.05	0

Table 4 shows the drift detection performance on the artificial dataset SINE, which contains abrupt concept drift. When using Naive Bayes as the classifier, MWDDM_H achieves the lowest detection delay and false nagative ratio

among all algorithms, but also has a certain false positive ratio, followed by MWDDM_M. In addition, EDDM and DDM have the highest detection delay and the lowest true positive ratio. When using Hoeffding tree as the classifier, HDDM_W achieved the lowest detection delay, followed by MWDDM_H and MWDDM_M, and EDDM and DDM had the highest detection delay.

Table 4 drift detection performance in SINE

	detector	DD	TPR	FPR	FNR
NB	MWDDM_H	33.80	1	0.10	0
	MWDDM_M	34.50	1	0.10	0
	FHDDMS	40.80	1	0	0
	FHDDM	49.85	1	0	0
	MDDM_A	41.65	1	0	0
	MDDM_E	41.70	1	0	0
	MDDM_G	40.70	1	0.04	0
	HDDM_W	34.75	1	0	0
	HDDM_A	94.15	1	0.04	0
	EDDM	209.00	0.05	0.99	0.95
	RDDM	88.63	1	0.31	0
	DDM	160.02	0.70	0.07	0
HT	MWDDM_H	34.05	1	0.01	0
	MWDDM_M	34.90	1	0.01	0
	FHDDMS	41.55	1	0.04	0
	FHDDM	49.20	1	0.04	0
	MDDM_A	51.80	1	0.04	0
	MDDM_E	39.35	1	0.07	0
	MDDM_G	39.35	1	0.07	0
	HDDM_W	33.35	1	0.04	0
	HDDM_A	57.65	1	0.12	0
	EDDM	247.50	0.05	0.99	0.95
	RDDM	97.65	1.00	0.31	0
	DDM	155.00	0.90	0.31	0.10

Finally, the drift detection performance of the algorithm in the MIXED dataset, which also has abrupt drift, is shown in Table 5. Similarly, MWDDM_M achieved the lowest detection delay and highest true positive ratio, followed by MWDDM_H, but both outperformed algorithms such as FHDDMS and HDDM_W. EDDM and DDM have the highest false negative rate.

To sum up, the experiments of MWDDM_H and MWDDM_M and the comparison algorithm on artificial datasets show that both in the dataset with abrupt drift and the dataset with gradual drift, in most cases, they can outperform all other comparison algorithms. All have the lowest detection delay, the highest true positive ratio and the lowest false nagative ratio. In addition, the algorithm proposed in this paper has certain defects, that is, MWDDM_H and MWDDM_M have a certain false positive ratio within an acceptable range in some data sets.

Table 5 drift detection performance in MIXED

	detector	DD	TPR	FPR	FNR
NB	MWDDM_H	33.35	1	0.20	0
	MWDDM_M	32.75	1	0.20	0
	FHDDMS	40.65	1	0.15	0
	FHDDM	48.70	1	0.12	0
	MDDM_A	39.45	1	0.12	0

	MDDM_E	39.65	1	0.17	0
	MDDM_G	39.65	1	0.17	0
	HDDM_W	34.40	1	0.30	0
	HDDM_A	72.80	1	0	0
	EDDM	42.50	0.15	0.98	0.85
	RDDM	99.05	1	0.37	0
	DDM	157.55	0.80	0.28	0.20
HT	MWDDM_H	32.25	1	0.25	0
	MWDDM_M	32.30	1	0.25	0
	FHDDMS	42.45	1	0.23	0
	FHDDM	52.40	1	0	0
	MDDM_A	49.10	1	0.12	0
	MDDM_E	39.40	1	0.29	0
	MDDM_G	40.30	1	0.28	0
	HDDM_W	35.70	1	0.35	0
	HDDM_A	66.60	1	0.25	0
	EDDM	338.25	0.20	0.99	0.80
	RDDM	94.35	1.00	0.50	0.00
	DDM	170.60	0.85	0.18	0.15

(3) Experiments of accuracy

In this paper, the proposed algorithms MWDDM_H and MWDDM_M are tested for accuracy in real-world datasets, namely POKER HAND, ELECTRICITY and FOREST COVERTYPE. The real-world dataset means that the specific location and duration of the concept drift in the dataset will not be known, so the evaluation indicators such as detection delay, true positive ratio, false positive rate and false nagative ratio will not be able to be evaluated on the real-world dataset. Therefore, on the real-world dataset, we consider the classification accuracy in the dataset as well as the running time and memory consumption.

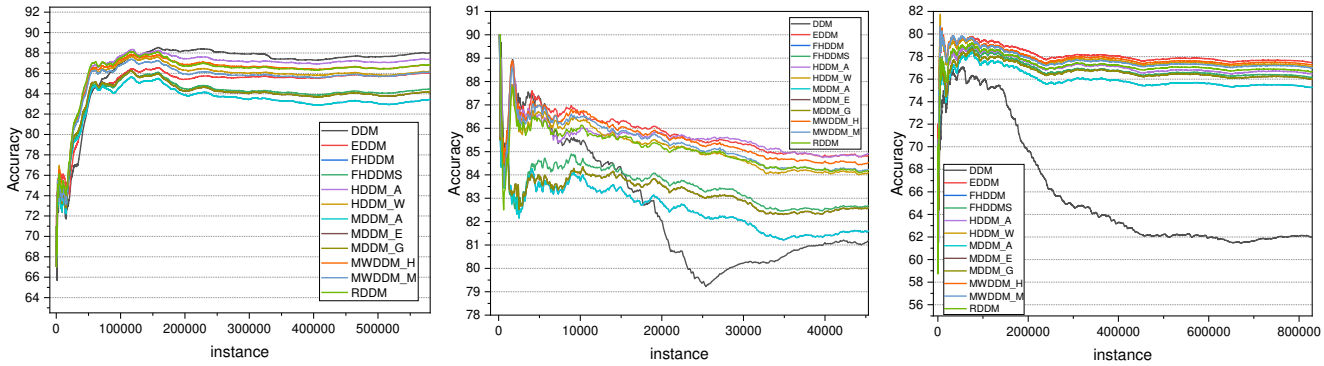
Figure 12 shows the classification accuracy of MWDDM_H, MWDDM_M, and other comparison algorithms on three real-world datasets using Naive Bayes as the classifier. In Figure 12(a), showing the performance on the POKER HAND dataset, DDM, HDDM_A, and MWDDM_H achieve the highest classification accuracy. In Figure 12(b), EDDM, HDDM_A, and MWDDM_H and MWDDM_M achieve the highest classification accuracy. In Figure 12(c), EDDM and MWDDM_H achieved the highest classification accuracy.

Figure 13 shows the classification accuracy of MWDDM_H, MWDDM_M, and other comparison algorithms on three real-world datasets using Hoeffding trees as classifiers. In Fig. 13(a), it can be concluded that DDM, HDDM_A, and MWDDM_H and MWDDM_M have the highest classification accuracy. In Figure 13(b), in the first half of the ELECTRICITY dataset, RDDM has the highest classification accuracy in most cases. In the second half of the dataset, MWDDM_H and MWDDM_M are more accurate in classification in most cases. Finally, in Figure 13(c), MWDDM_H and MWDDM_M have higher classification accuracy in most cases, followed by RDDM. In addition, it can be found in Figure 12 and Figure 13 that the classification accuracy of MWDDM_H and MWDDM_M can rise faster, that is, faster than all other algorithms to recover from the concept drift that may exist in the real-world dataset, which also means that MWDDM_H and MWDDM_M have lower detection latency for concept drift, enabling faster detection of drift and then resetting the classifier.

Finally, this paper summarizes and analyzes the spatiotemporal consumption of MWDDM_H, MWDDM_M and other comparison algorithms on real-world datasets. Tables 6 and 7 show the space-time consumption of MWDDM_H and MWDDM_M and other algorithms on three real-world datasets with Naive Bayes and Hoeffding trees as learners, respectively. For the convenience of display, the three real-world datasets of POKER HAND, ELECTRICITY and

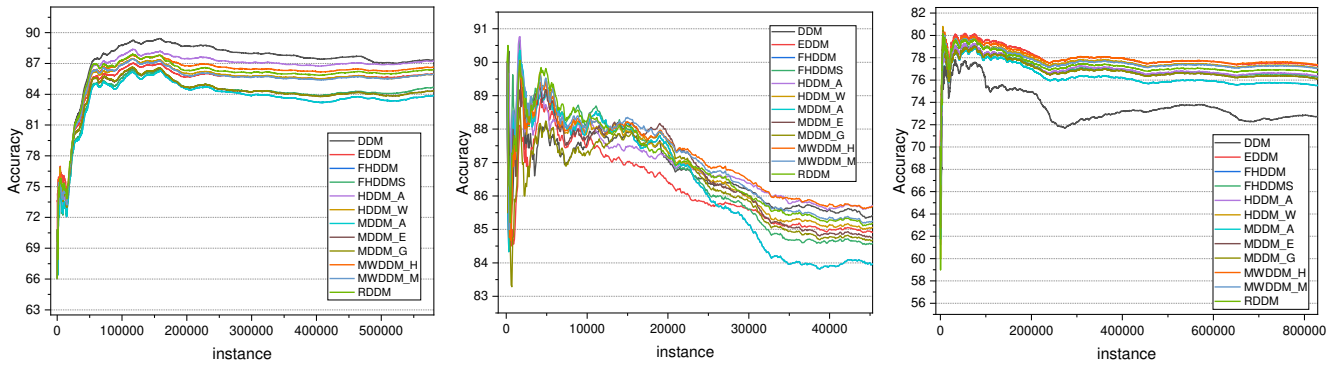
FOREST COVERTYPE are represented by PH, ELE, and FC in the table, respectively. In terms of running time, MWDDM_H and MWDDM_M spend less running time on datasets POKER HAND and ELECTRICITY than most other comparison algorithms. In terms of memory consumption, although MWDDM_H and MWDDM_M use double-layer windows, they can generally achieve less memory consumption due to the way of accessing the prediction results.

In summary, the experiments of MWDDM_H and MWDDM_M and their comparison algorithms on three real-world datasets show that MWDDM_H and MWDDM_M have the highest or higher classification accuracy in most cases, and their time and space consumption also have excellent performance. In particular, it is able to recover from concept drift that may exist in real-world datasets faster than other contrast algorithms, which shows that MWDDM_H and MWDDM_M can detect drift faster to allow the learner to react.



(a) NB+ COVERTYPE (b) NB+ ELECTRICITY (c) NB+ POKER HAND

Figure 12 classification accuracy in real-world dataset using NB



(b) HT+ COVERTYPE (b) HT+ ELECTRICITY (c) HT+ POKER HAND

Figure 13 classification accuracy in real-world dataset using HT

Table 6 time and space performance using NB

	evaluation time			model cost		
	PH	ELE	FC	PH	ELE	FC
MWDDM_H	4.00	0.47	12.31	2.02	1.71	1.42
MWDDM_M	4.91	0.54	11.79	2.44	1.89	1.34
DDM	7.97	0.48	12.10	5.75	1.84	1.74
EDDM	3.92	0.52	12.11	2.23	1.50	1.70
RDDM	4.16	0.45	11.53	2.80	2.62	1.90
FHDDM	4.01	0.50	11.83	1.93	1.64	1.32
FHDDMS	4.25	0.50	11.73	2.07	1.68	1.32
MDDM_A	4.39	0.55	13.27	2.14	1.85	1.50
MDDM_E	4.59	0.50	12.56	2.24	1.69	1.42
MDDM_G	4.23	0.48	12.61	2.07	1.64	1.42
HDDM_A	4.30	0.52	11.42	2.04	1.67	1.54
HDDM_W	4.25	0.48	14.47	2.08	1.64	1.63

Table 7 time and space performance using HT

	evaluation time			model cost		
	PH	ELE	FC	PH	ELE	FC
MWDDM_H	7.39	0.98	21.64	7.32	8.30	3.74
MWDDM_M	7.65	1.01	20.43	7.44	8.38	3.50
DDM	9.42	1.34	25.23	8.90	1.22	4.30
EDDM	9.81	1.09	22.53	1.02	8.44	3.74
RDDM	9.66	1.27	26.47	1.11	1.40	6.26
FHDDM	6.28	0.80	20.03	5.99	6.43	3.40
FHDDMS	6.55	0.86	22.08	6.29	7.00	3.76
MDDM_A	6.52	0.95	19.03	6.27	7.79	3.24
MDDM_E	7.09	0.78	22.78	6.82	6.37	3.88
MDDM_G	6.55	0.91	20.45	6.30	7.41	3.48
HDDM_A	8.55	1.14	33.14	8.36	9.189	6.88
HDDM_W	9.27	1.02	27.72	8.25	8.31	4.73

5 Conclusion

In many real-world application scenarios such as user preferences, monitoring systems, weather forecasting, and financial fraud detection, concept drift has become an urgent problem to be solved. In order to better solve the problem of concept drift in data flow, this paper proposes a piecewise weighted concept drift detection method (MWDDM), which proposes a threshold parameter for level transition, and introduces a "Stable level-Warning level-Drift level" three-level segmentation weighting mechanism in the concept drift detection process, and apply it to the double sliding window mechanism. During the "stable level", MWDDM will assign weights to the instances within the window, the newest instances are assigned a larger weight, and the old outdated instances are assigned a lower weight, and the difference in weight values between instances is small. After entering the "warning level", the algorithm will increase the difference in weights between instances within the windows to detect concept drift faster. Finally, in the "drift level", two variants of the algorithm use Hoeffding's inequality and Mcdiarmid's inequality to determine whether concept drift has occurred. The method in this paper can detect the abrupt drift and gradual concept drift in the data stream at the same time faster with lower false positive ratio and false nagative ratio, and achieve high classification accuracy in real-world datasets, and achieve excellent performance in terms of space-time consumption. In future work, consider using an adaptive windowing mechanism and take measures to enhance the robustness to noise in the data stream to reduce the false positive ratio.

Ethical approval

With the unanimous consent of all our authors. The paper is only about a research on a machine learning algorithm and does not involve ethical disputes.

Funding details

This work is supported by the National Nature Science Foundation of China (62062004), the Ningxia Natural Science Foundation Project (2020AAC03216)

Conflict of interest

No potential conflict of interest was reported by the authors

Informed Consent

All authors are informed and agreed to be one of the authors of this article

Authorship contributions

The first author mainly guides the second author to complete the writing of the paper and complete the experiments The second author developed the model, carried out the parameter estimations and planned as well as performed the mass transfer experiments. The second author also wrote the main part of the manuscript and took part in the planning and execution of the fermentation experiments. The rest of authors took part in the development of the model, planned and carried out the main part of the fermentation experiments, analyzed the results and assisted in the mass transfer experiments.

References

[1] Dongre P B, Malik L G. A review on real time data stream classification and adapting to various concept drift scenarios[C]//Proc of 2014 IEEE International Advance Computing Conference (IACC). Gurgaon: IEEE, 2014: 533-537.

[2] IWASHITA A S, PAPA J P. An Overview on Concept Drift Learning[J]. IEEE Access, 2018, 7:1532-1547.

[3] MASSI M C, IEVA F, LETTIERI E. Data mining application to healthcare fraud detection: a two-step unsupervised clustering method for outlier detection with administrative databases[J]. BMC Medical Informatics and Decision Making, 2020, 20(1):1-11.

[4] DEMERTZIS K, ILIADIS L, ANEZAKIS V. A Dynamic Ensemble Learning Framework for Data Stream Analysis and Real-Time Threat Detection[C]//Proceedings of Artificial Neural Networks and Machine Learning(ICANN). Greece: Springer, 2018:669-681.

[5] DASH R, SAMAL S, DASH R, et al. An integrated TOPSIS crow search based classifier ensemble: In application to stock index price movement prediction[J]. Applied Soft Computing, 2019, 85:105784.

[6] GAMA J, MEDAS P, et al. Learning with Drift Detection[J]. Advances in Artificial Intelligence – SBIA, 2004, 3171:286-295.

[7] BAENA M, et al. Early drift detection method[C]//Proceedings of the International Workshop on Knowledge Discovery from Data Streams. Porto: Citeseer, 2006, 6:77–86.

[8] Nishida K, Yamauchi K. Detecting Concept Drift Using Statistical Testing[M]//Berlin, Heidelberg: Springer Berlin Heidelberg:264-269.

[9] Barros R S M D, Hidalgo J I G, Cabral D R D L. Wilcoxon Rank Sum Test Drift Detector[J]. Neurocomputing, 2018, 275: 1954-1963.

[10] MAHDI O A, PARDEDE E, ALI N, et al. Diversity measure as a new drift detection method in data streaming[J]. Knowledge-Based Systems, 2020, 191:105227.

[11] BIFET A, GAVALDÁ R. Learning from Time-Changing Data with Adaptive Windowing[C]//Proceedings of the Seventh SIAM International Conference on Data Mining. Minneapolis: SIAM, 2007:443-448.

[12] HUANG D T J, KOH Y S, DOBBIE G, et al. Detecting Volatility Shift in Data Streams[C]//Proceedings of IEEE International Conference on Data Mining(ICDM). Shenzhen: IEEE, 2014:863-868.

[13] PESARANGHADER A, VIKTOR H L. Fast Hoeffding Drift Detection Method for Evolving Data Streams[C]//Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2016:96-111.

[14] PESARANGHADER A, VIKTOR H L. McDiarmid Drift Detection Methods for Evolving Data Streams[C]//Proceedings of 2018 International Joint Conference on Neural Networks, 2018:1-9.

[15] João Gama, Raquel Sebastião, Pedro Pereira Rodrigues, On evaluating stream learning algorithms, Mach. Learn. 90 (3) (2013) 317–346.

[16] LU J, LIU A, DONG F, et al. Learning under Concept Drift: A Review[J]. IEEE Transactions on Knowledge and Data Engineering, 2020, 31(12):2346-2363.

[17] WARES S, ISAACS J, ELYAN E. Data stream mining: methods and challenges for handling concept drift[J]. SN Applied Sciences, 2019, 1(11).

[18] Hoeffding, W.: Probability inequalities for sums of bounded random variables. Am. Stat. Assoc. 58(301), 13–30 (1963)

[19] KHAMASSI I, SAYED-MOUCHAWEH M, HAMMAMI M, et al. Discussion and review on evolving data streams and concept drift adapting[J]. Evolving Systems, 2019, 9(1):1-23.

[20] FRIAS-BLANCO I, CAMPO-AVILA J D, RAMOS-JIMENEZ G, et al. Online and Non-Parametric Drift Detection Methods Based on Hoeffding Bounds[J]. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(3):810-823.

[21] Ali P , Herna V , Eric P . Reservoir of Diverse Adaptive Learners and Stacking Fast Hoeffding Drift\n, Detection Methods for Evolving Data Streams[J]. Machine Learning, 2017(3):1-33.

[22] Barros, R. S., Cabral, D. R., Gonçalves, P. M, Jr., & Santos, S. G. (2017). Rddm: Reactive drift detection method. Expert Systems with Applications, 90, 344–355.

[23] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, “Moa: Massive online analysis,” Journal of Machine Learning Research, vol. 11, no. May, pp. 1601–1604, 2010.

Meng Han, born in 1982. Ph.D. Her main research interests include Data Mining.

Zhiqiang Chen, born in 1998. M.S. His main interest is Data Stream Classification.

Hongxin Wu, born in 1998. M.S. His main interest is Data Stream Classification.

Muhang Li, born in 1997. M.S. His main interest is Pattern Mining.

Xilong Zhang, born in 1996. M.S. His main interest is Data Stream Classification.