

RESEARCH

Open Access



Multi-type concept drift detection under a dual-layer variable sliding window in frequent pattern mining with cloud computing

Jing Chen^{1,3}, Shengyi Yang⁴, Ting Gao³, Yue Ying², Tian Li⁵ and Peng Li^{2*}

Abstract

The detection of different types of concept drift has wide applications in the fields of cloud computing and security information detection. Concept drift detection can indeed assist in promptly identifying instances where model performance deteriorates or when there are changes in data distribution. This paper focuses on the problem of concept drift detection in order to conduct frequent pattern mining. To address the limitation of fixed sliding windows in adapting to evolving data streams, we propose a variable sliding window frequent pattern mining algorithm, which dynamically adjusts the window size to adapt to new concept drifts and detect them in a timely manner. Furthermore, considering the challenge of existing concept drift detection algorithms that struggle to adapt to different types of drifting data simultaneously, we introduce an additional dual-layer embedded variable sliding window. This approach helps differentiate types of concept drift and incorporates a decay model for drift adaptation. The proposed algorithm can effectively detect different types of concept drift in data streams, perform targeted drift adaptation, and exhibit efficiency in terms of time complexity and memory consumption. Additionally, the algorithm maintains stable performance, avoiding abrupt changes due to window size variations and ensuring overall robustness.

Keywords Data Stream, Frequent Pattern Mining (FPM), Dual-layer Variable Sliding Window, Concept Drift Detection (CDD), Cloud Computing

Introduction

Multi-type concept drift (CD) detection is widely applied in cloud and security applications [1]. The mining of CD in frequent patterns (FPs) with latent risks, allows the monitoring of changes in potential-risk patterns and related data models in cloud environments [2–4]. It is used in cloud computing [5], security information detection [6–9], blockchain-based [10], healthcare data prediction [11], intelligent data processing in IoT [12–15], recommendation systems [16, 17], and fault container instance seq finding [18]. CD mining reduces data leakage risk due to sudden data mutations, and improves privacy, and can be used for the timely identification of model performance deterioration or data distribution changes so that adjustments and optimizations can be

*Correspondence:

Peng Li
lipeng@njupt.edu.cn

¹ School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing 210023, China

² School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China

³ Baotou Teachers' College of Inner Mongolia University of Science and Technology, Baotou 014030, Inner Mongolia, China

⁴ School of Physics and Mechatronic Engineering, Guizhou Minzu University, Guiyang 550025, China

⁵ School of Computer and Software, Nanjing Vocational University of Industry Technology, Nanjing 210003, China

applied. The aim of the present study is to address CD in hidden risk patterns and provide decision support for future research. Challenges in this field include data streams' high velocity and variability, which cause issues with computational resource availability. For example, data streams may be stored in a temporal DB in a highly trusted and low-latency industrial network, but there is a risk of data tampering during transmission [52, 53]. Concept drift [19] is also a prominent issue in decision-making within the e-commerce data stream domain [20]. It refers to the phenomenon where the underlying concepts or relationships in the data change over time. In e-commerce data streams [21], CD can occur due to evolving user preferences, market trends, or external factors. To mitigate the impact of CD on decision-making, it is crucial to develop mining models that can effectively adapt to these changes in e-commerce data streams [22]. These models should be able to detect and adjust to shifts in data patterns, ensuring that the decision-making process remains accurate and relevant over time.

Riverola et al. demonstrated an improvement to a successful e-mail filtering model to track spam domain CD [23], while Gulla et al. discussed a new approach to detect semantic drift using concept signatures [24]. A new Bayesian framework [25] in data stream pattern recognition was presented for feature selection. Ruano-Ordás et al. presented a detailed study of CD in the e-mail domain considering types of CD and message classes (spam and ham) [26]. Ding et al. studied entropy-based time domain feature extraction for online CD detection [27]. Rabiou et al. aimed to provide a literature review of models to guide researchers and practitioners [20]. Two variants of the model [28] have been proposed to minimize negative transfers in high-volume model transfer frameworks. CD-tolerant transfer learning, which adapts the target model and source domain knowledge to changing environments, has not been well explored. A hybrid ensemble approach dealt with this problem when target domain data were generated chunk by chunk from non-stationary environments [21]. In 2023, Liu et al. introduced two new CD handling methods, namely error contribution weighting and gradient descent weighting [22], which are based on the principle of continuous adaptive weighting and aim to improve detection and handling of CD, adapting to changes in data streams in constantly evolving environments. There are also other classical active detection algorithms, such as the drift detection method [29] and the early drift detection method [30].

The above mentioned studies were predominantly focused on the detection of CD in data streams. However, there is indeed a research gap when it comes to detecting and identifying multiple types of CD during FP

mining processes in complex data streams. To address the challenges encountered in mining FPs on evolving data streams [31–33], we propose the dual-layer variable sliding window concept drift detection (DLVSW-CDTD) algorithm, which utilizes a dual-layer variable sliding window whose size is determined dynamically based on the occurrence of multi-type CD. The core idea of DLVSW-CDTD is to adjust the window size based on the stability of the data stream and the utilization of a dual-layer sliding window. When the data stream is in a stable state without CD, the window size continuously increases to capture FPs over longer periods. This is done to fully utilize longer periods of data for mining FPs and avoid prematurely discarding useful information [34].

CD indicates significant changes in data distribution or feature relationships, which may render the previously mined FPs inaccurate or less useful. To adapt to such changes, our algorithm automatically adjusts the window size to capture the new FPs that emerge due to CD [35]. The VSW-CDD algorithm achieves dynamic window adjustment by considering the evolution of the data stream and the changes in FPs [36]. This approach effectively adapts to changes in the data stream and maintains high accuracy and efficiency in the process of FP mining [37].

Dynamic changes in data and CD is an ongoing problem in cloud computing environments. CD is a change in the statistical characteristics of data over time, which can be caused by various factors such as user behavior, system configuration, application updates, etc. [38]. Such changes may have a significant impact on data analysis and decision making, therefore, detecting and responding to CD is an important task in data analysis and processing [39].

FP mining is a method of finding frequently occurring patterns or correlations in big data, and it has a wide range of applications in many fields, such as market analysis, social network analysis, and anomaly detection [40]. However, in the process of FP mining, CD may negatively affect the mining results, so it is necessary to detect and deal with CD [41].

The double-layer variable sliding window strategy is an effective method for detecting CD by considering both local and global statistical properties [42]. However, existing dual-layer variable sliding window strategies have some problems in dealing with multiple types of CDs, such as not being able to effectively deal with CDs at different levels of granularity, or to accurately recognize multiple types of CDs that occur simultaneously [43].

Aiming at the above problems, in this paper a multi-type CD detection method is investigated, which can effectively deal with CD in FP mining under a dual-layer

variable sliding window strategy [44]. The main contribution of this paper is to propose a new multi-type CD detection method, which can consider both local and global statistical properties, can detect CD at different granularity levels, and can accurately identify multiple types of simultaneous CDs [45]. The research results in this paper will help to improve the efficiency and accuracy of FP mining, which is of great theoretical significance and important for solving problems in practical applications [46]. For example, in recommendation systems in cloud environments, changes in user behavior can be monitored in real time using the proposed multi-type CD detection method, so as to adjust the recommendation strategy and improve the accuracy and user satisfaction with the recommendation system [47]. In addition, the method can also be applied in the fields of anomaly detection, system monitoring and decision support in cloud environments [48].

The structure of the remaining paper is as follows: The second section provides an introduction to the relevant technical background knowledge. The third section presents the proposed algorithm, where the limitations and issues of fixed sliding window mining for data streams are first discussed, and then a new window dynamically adjusted based on the concept of data stream size is proposed. Subsequently, we propose the DLVSW-CDTD algorithms to effectively detect different types of CD during the data stream mining process. In the fourth section, extensive experiments are conducted using real and synthetic datasets obtained using the open-source data mining library SPMF [25]. The results confirm the efficiency and feasibility of the algorithm. In the fifth part, the main work of this paper is summarized, and a brief overview and prospect of future research directions is given.

Related work and definitions

Overview of the data stream frequent pattern

In cloud computing environments, FP mining faces a series of challenges. First, the volume of the data streams is huge, and traditional FP mining methods cannot handle such a large amount of data effectively. Second, the data in the stream are dynamically changing, i.e., CD occurs, which requires real-time monitoring and updating of FPs. In addition, to ensure the cost-efficient utilization of cloud resources, efficient computation and storage management is necessary to meet the requirements of high concurrency and high throughput.

To address the above challenges, researchers have proposed many effective FP mining methods for data streams. These methods mainly include: sliding window-based methods, tree-based methods, statistics-based methods, etc. These methods achieve efficient FP mining through the utilization of distributed computing and

storage resources. For example, sliding window-based approaches detect FPs by sliding a window on the data stream and statistically learning the data inside the window; tree-based approaches discover FPs by constructing and traversing a tree structure; and statistics-based approaches discover FPs by building a statistical model that describes the distribution and changes in the data stream [40, 41]. Table 1 shows the symbol overview of FP.

Generally, a set of continuously-arriving data is defined as a data stream, expressed as $DS = \{T_1, T_2, \dots, T_i, \dots\}$, as shown in Table 2. The data in this example contain 5 transactions. T_i represents the sample arriving at time i . Each datum has a unique identity, denoted as a TID.

In the data stream, the FP P is defined as the number of samples containing P , denoted as $freq(P)$. The support of the FPs P is expressed as $support(P) = freq(P)/n$, where n is the number of samples included in the data stream. The concept of the FP is defined as follows. Given a data stream DS which contains n samples, we define a minimum support threshold θ , whose the value range is $(0, 1]$. If the mode P satisfies Eq. 1, it is called the FP.

$$support(P) = freq(P)/n \geq \theta \quad (1)$$

Introduction of the window model

Window model

In FP mining, a dual-layer variable sliding window strategy for multi-type CD detection is an effective way to deal with rapid changes in data streams. Window modeling is a data processing technique that captures changes

Table 1 Symbol overview of frequent pattern

Symbol	Meaning
DS	Data Stream
T_i	Transaction that arrives at time i
TID	Data Identification
$freq(P)$	frequent pattern P
$support(P)$	Support degree of the frequent pattern P
θ	Minimum support threshold
n	Number of data in the data stream

Table 2 Data Stream DS

TID	Data
1	{a,b}
2	{a,c,d,e}
3	{b,c,d,f}
4	{a,b,c,d}
5	{a,c,f}

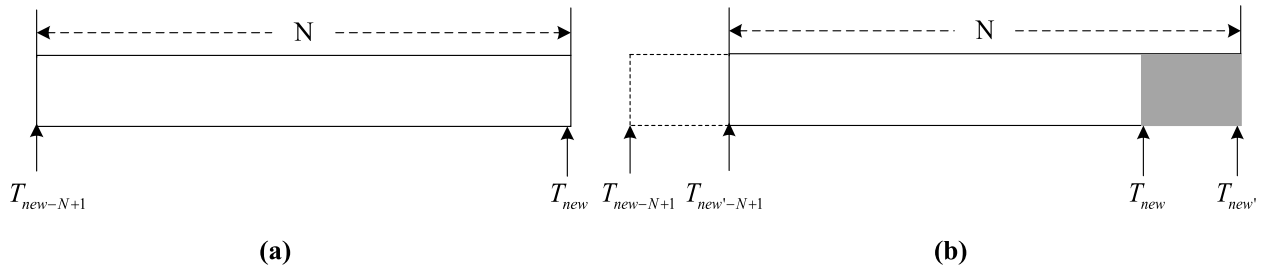


Fig. 1 **a** Handling the transaction T_{new} . **b** Handling the transaction $T_{new'}$

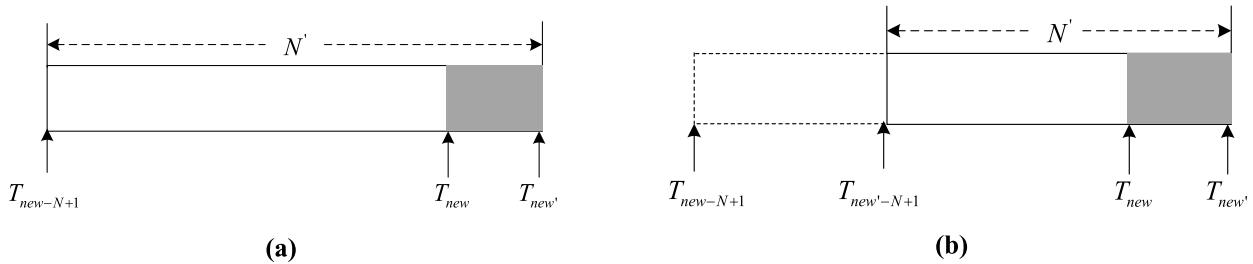


Fig. 2 **a** Handling transactions without concept drift $T_{new'}$. **b** Handle transactions when there is a concept drift $T_{new'}$

in data by sliding a window over the data stream. In CD detection, window modeling can facilitate the tracking of changes in data distribution. In multi-type CD detection, the window model needs to handle different types of data, which increases the complexity of processing. However, this complexity can be handled efficiently through the use of a two-layer variable sliding window strategy.

Cloud computing is a computing model that allows users to access shared computing resources over the Internet. Cloud computing provides powerful computing power and storage space to handle large-scale data. In multi-type CD detection, cloud computing can provide the required computing resources and storage space to handle large-scale data streams. Cloud processing can be facilitated through data stream slicing. In multi-type CD detection, the combination of window modeling and cloud computing can provide an effective solution. First, the window model can capture changes in the data, while cloud computing can provide the required computational resources and storage space to handle these changes. Second, by using a dual-layer variable sliding window strategy, we can handle different types of data more efficiently. Finally, the distributed computing resources in the cloud can accelerate data processing and improve the efficiency of CD detection. The combination of window modeling and cloud computing plays an important role in multi-type CD detection. By integrating these two techniques, we can effectively cope with rapid changes in the data stream and enhance the efficiency of CD detection.

There are three commonly used window models, namely the landmark, the sliding and the damped window models, with the sliding model being the most commonly used. The sliding window model is also divided into two types, namely the fixed-width sliding window, where the number of samples in the window is fixed, and the other is the variable sliding window, where the number of data in the window is variable. Data processing takes place in different ways, as shown in Figs. 1 and 2, respectively; the relevant symbols are shown in Table 3.

As shown in Fig. 1(a) and Fig. 1(b), a fixed-width sliding window can handle the latest data by directly removing the expired data. Figure 2(a) shows the state of the fixed sliding window when there is no new data input. When the latest data $T_{new'}$ enters the window, the $|new' - new|$ bars between $T_{new-N+1}$ and $T_{new'-N+1}$ will be removed from the window. The details are shown in Fig. 1(b). In general, the length of the window N is not very large to avoid the CD of the data within the window.

Assuming that the size of a given window is N , as shown in Fig. 2(a) and Fig. 2(b), variable sliding window-based methods handle conceptual changes caused by the most recent entry window by expanding and contracting the window's width.

As shown in Fig. 2(a), when the recently entered data do not cause a conceptual change, the sliding window scales from size N to $N + |new' - new|$ to handle the latest data. As shown in Fig. 2(b), when the data in the

Table 3 Symbol description of the window model

Symbol	Meaning
N	Window length
N'	Length after the window size change
T_{new}	The latest data at the current moment of the window
$T_{new'}$	The latest data for the new entry window
$T_{new-N+1}$	End data of the current moment of the window
$T_{new'-N+1}$	End data of the new entry window
$ new' - new $	Number of data between $T_{new-N+1}$ and $T_{new'-N+1}$

window change conceptually, the size of the sliding window shrinks from N to N' .

The fixed sliding window approach involves determining the window size based on prior or experience. This can be done by referring to previous studies or considering the characteristics of the dataset. Once the window size has been set, it remains constant throughout the mining process. On the other hand, variable sliding window algorithms can adjust the window size in different ways. For instance, Hui Chen et al. [42] proposed a time decay model to differentiate patterns in recent transactions from historical transactions, ensuring that the most recent information in the data stream is given prominence. They utilized the concept of FP change to dynamically determine the appropriate window size [43].

Attenuation model

Usually, data stream changes over time are unpredictable. In such cases, it is undesirable to treat all samples as equally important. In general, the latest data generated is more valuable than historical data. Therefore, the attenuation model is an effective method to deal with this kind of time-sensitive data streams. In essence, attenuation models involve the association of weights of historical data or modes with time, and with the passage of time, these weight coefficients change accordingly to emphasize the importance of recent data.

When setting up the attenuation factors, there are usually three types of settings. Random decay factors are stochastic, and may cause instability of the FPs obtained by the mining algorithm. Fixed values are usually based on previous related studies, and the quality of the effect depends on the knowledge of experts. Dynamically calculated are obtained by combining other parameter values in the algorithm design. After the experimental verification of some studies, this approach was selected for the proposed algorithm. In current mining studies of data streams, decay models are commonly used in combination with window models.

Conceptual drift processing method for cloud computing

In data stream mining, the arriving data may change over time due to the inherent temporal nature of the data stream [45]. This phenomenon is generally known as CD. In cloud computing environments, the CD encountered in data preprocessing is mainly due to the diversity of data sources. To address these challenges, the following measures can be applied. In the data cleaning stage, by removing duplicate, invalid, or erroneous data, the data quality can be improved, which mitigates the impact of CD. In the feature selection stage, to counteract the impact of CD on features, representative and stable features can be selected to reduce CD's impact on the model. In the data labeling stage, diverse labeling methods and labelers should be used to enhance the accuracy and reliability of the data.

Currently, CD is generally classified based on the speed of concept change [48]. As shown in Fig. 3(a), a solid circle marked with numbers is used to represent each paragraph of data, and the numbers represent the chronological order. It can be seen that the transition between Concept 1 and Concept 2 is fast, and the old Concept 1 is soon replaced by Concept 2 with a completely different data distribution. This type of drift is referred to as mutant CD. On the contrary, as shown in Fig. 3(b), the transition between Concept 1 and Concept 2 is slow; the former is replaced gradually, and the concepts are more or less similar before and after the drift, so this drift is referred to as gradual CD.

Among the many methods that deal with CD, the window-based CD monitoring method is one of the common methods. Larger windows are associated with higher performance accuracy, but they may also contain unnoticed CD, while smaller windows facilitate better detection of CD [19]. For example, Husheng et al. [49] proposed a CD-type identification method based on multi-sliding windows. The method consisted of three stages; first, the drift position was detected during the first detection stage by sliding the base window forward. Then, during the growth stage, the growth of the accompanying window was used to detect the drift length and identify the drift categories based on the drift length. Finally, during the tracking stage, the drift subcategories are identified based on the different tracking flow ratio curves generated during the window tracking process. Therefore, this method is able to effectively identify the type of CD, accurately analyze the key information in the online learning process, and improve the efficiency and generalization performance of streaming data analysis and mining.

Therefore, most existing studies adopt a "circuitous" strategy to detect CD. It involves determining whether a data stream has experienced CD by considering the "possible cause of CD" and the "possible consequences

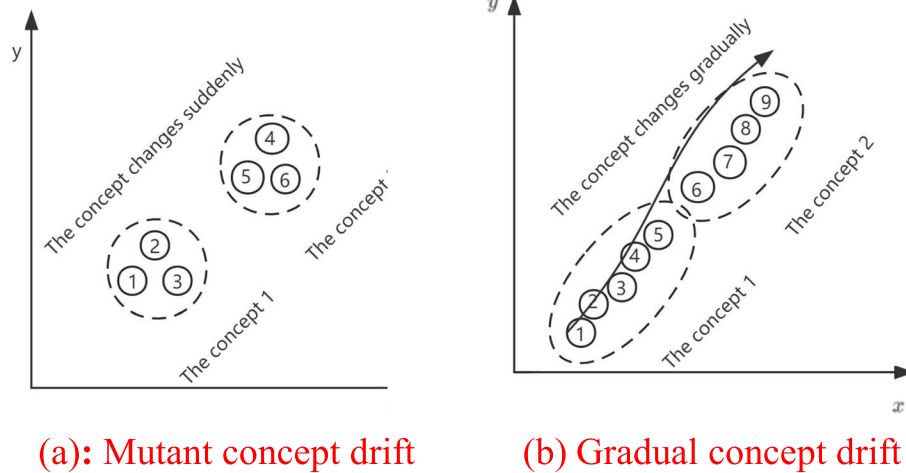


Fig. 3 Common type of concept drift

after CD" [50]. For instance, Lu et al. [51] tackle real-time data and pre-existing CD, focusing on detecting causes through a data-driven approach and comprehensively handling CD in terms of time, content and manner.

Processing model and algorithm

The proposed DLVSW-CTD algorithm

The DLVSW-CTD algorithm is an algorithm for handling CD in cloud computing environments. In the following, we will focus on the algorithm's relation to cloud computing.

In cloud computing environments, updating must account for the aging and overfitting of the model. To enhance the accuracy and robustness of the model, the following measures can be adopted. By utilizing the real-time data processing capabilities of cloud computing, the performance of the model is monitored and evaluated in real time. This ensures timely detection of aging or overfitting phenomena. In practical applications, occurrences of CD require corresponding adaptation of the data or modification of mining models depending on the CD type. Therefore, the new algorithm proposed in this paper is suitable for detecting multiple types of CD.

To address the issue of CD in the process of FP mining, the DLVSW-CTD algorithm (Dual-layer Variable Sliding Window-CD Type Detection), which utilizes a dual-layer variable sliding window model to handle CD phenomena in data streams and applies it to FP mining. The model utilizes two algorithms based on CD detection and type detection of the variable sliding window, aiming to handle various types of CD problems and ensure that FPs are based on the latest trends in the data.

The DLVSW-CTD algorithm framework

In this section, the proposed FP mining algorithm is introduced. The algorithm model framework is primarily divided into five parts, as shown in Fig. 4. After the data pre-processed, they first pass through the first module, which applies the Variable-Size Window Drift Detection algorithm (VSW-DD) to detect CD. Upon detecting CD, the second module is used to adapt or modify the mining model accordingly for Multi-type Concept Detection. This algorithm is suitable for data mining scenarios with multiple CDs.

The DLVSW-CTD is an algorithmic framework for handling CD in cloud computing environments. In the following, we will focus on the aspects of the framework that are specifically related to cloud computing.

In this algorithm, based on the concept of CD detection, the window size is initially set manually, and is then adjusted to adapt to the data stream according to the FP changes, the potential distribution changes of the data and the type of CD. Then, different types of CD are detected using the length of the embedded dual-layer window, and different attenuation coefficients are applied to reduce the impact of different types of CD, to dig out the latest concepts of the FPs, and to reduce the impact of CD on mining. When the data are constantly updated, at each instance a new data pane is inserted and the presence of mutation or gradient is assessed.

In the first stage of the DLVSW-CTD framework, involving the algorithm initialization. The second stage is the application of the VSW-CDD algorithm for concept detection with the variable window size, and includes two aspects. The first aspect is the FP set mining using the initialized window, and saving the results in a monitoring prefix tree through the insertion of a

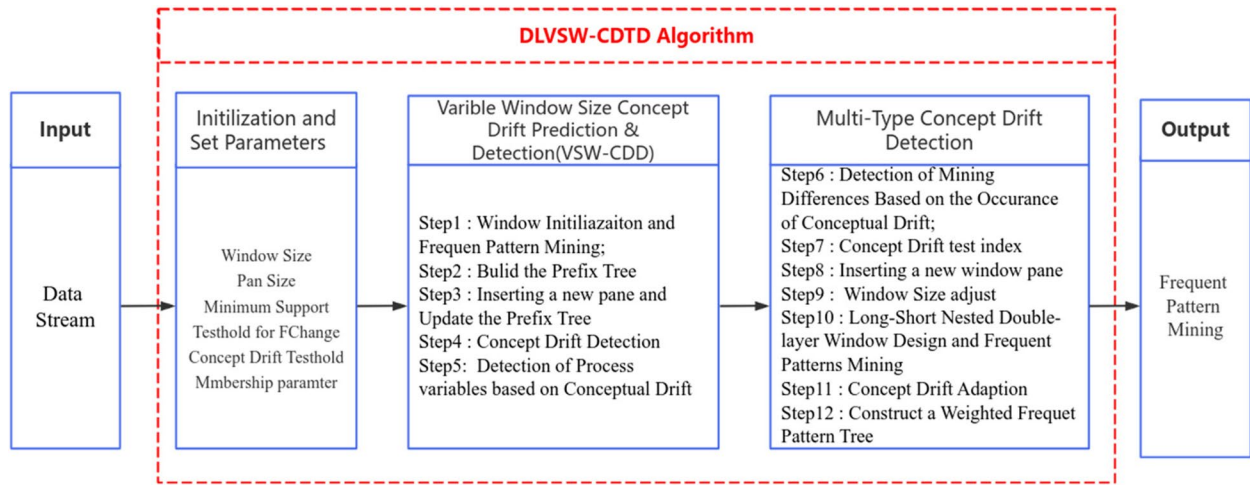


Fig. 4 DLVSW-CTD Algorithm Framework

new data pane, and update the prefix tree (Step1—Step4). If there is no change, the window size continues to increase, and new data panes are inserted continuously (Step 5).

In the fourth stage (Step6- Step12), according to the data stream characteristics of the different CD types, the embedded dual-layer window is adapted based on the detected effect to analyze the data within the window. Finally, in the fifth stage, the latest set of FPs is output.

Design of the DLVSW-CTD algorithm

In the algorithm design of this section, the associated symbolic definitions are shown in Table 4.

Step 1: window size initialization and frequent pattern mining

The window is defined through its initial size and other parameters such as pane size, minimum change threshold and minimum support threshold. The relevant parameters are set according to literature values, and can be adjusted according to the experimental results. After window initialization, the FP-growth algorithm is used to mine the FP set and save the results in a simple compact prefix tree (SCP-Tree), as described in the following.

Step 2: build the prefix tree

The SCP-Tree structure adopted in this paper is a simple and compact tree-like data structure similar to FP-Tree. The FPs are inserted in the SCP-Tree incrementally when each data pane arrives, and the tree is dynamically adjusted through branch sorting.

Table 4 Symbol description

Symbol	Meaning
$\theta(i)$	The weight of the data in bar i in the window
CP_S	Concept drift checkpoint within a short window
ms	Minimum support threshold
MTh	Frequent pattern change rate
Rt	Minimum threshold for the conceptual drift test index R
LW	Long window
SW	Short window
λ	Log-type membership function parameter value
SF_{t_i}	Frequent set of items within a short window at the time t_i
$SF_{t_i}^+(t_j)$	The set of new frequent term sets at t_j within a short window relative to t_i
$SF_{t_i}^-(t_j)$	Within a short window, the set of items infrequent at t_j , but the set of items frequent at t_i
$SFChange_{t_i}(t_j)$	The change rate of the frequent term set at the t_j within the short window relative to the t_i
$LFChange_{t_i}(t_j)$	The change rate of the frequent term set at the t_j within the long window relative to the t_i

Let p be a non-root node in the SCP Tree. If p is a regular node, then it contains 5 fields. The structure is shown in Fig. 5, where $p.item$ is the project name; $p.count$ records the support count of the item; $p.next$ points to

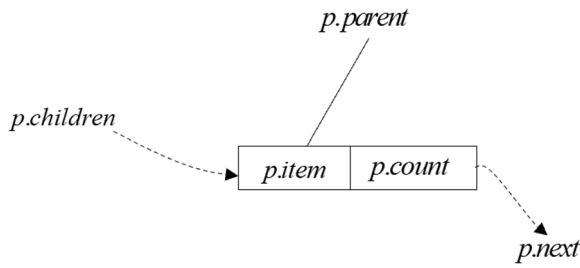


Fig. 5 Schematic diagram of the conventional node structure in the SCP Tree

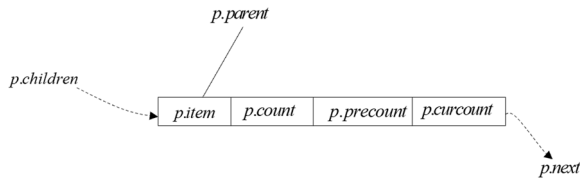


Fig. 6 Schematic representation of the tail node structure in the SCP Tree

the next node of the same project name (represented by a link in the tree; if no relevant branch exists, the link to the node from the header table is inserted); *p.parent* records the parent node, and *p.children* is the header pointer of the child node.

If *p* is a tail node, the structure diagram is shown in Fig. 6. The tail node contains two additional fields: *p.precount* records the support count before the checkpoint and *p.curcount* represents the support counts after recording the checkpoint. The initial values of both these fields are 0. The SCP-Tree also contains a *Head_Table* structure, which records the total support count per item in the tree and ranks the counts in descending order. Starting with an empty tree, each incoming data pane needs to be inserted into the SCP-Tree.

Step 3: insert a new pane and update the prefix tree

After completing the window initialization process, new data are added to the window by updating the support for all the relevant FPs. For new items included in the newly-arriving data, a new node is created and the support count of the items is added to the header table. To truly identify all new sets of frequent items, all individual items in the existing window need to be monitored through a support count update. After inserting the full pane, the prefix tree is scanned and updated.

Step 4: concept drift detection

CD detection in the DLVSW-CDTD Algorithm is divided into two parts, namely the detection of the process variables that cause CD, and the other is the detection of the change of the mining results caused by the CD.

Step 5: detection of variables based on causing conceptual drift

Hypothesis testing is a method to infer the distribution characteristics of the population data based on the characteristics of the distribution of the sample data [23]. Its purpose is to judge whether there is a sampling error or essential difference between samples, or between a sample and the population. The common types of test assumptions include the F-test, the T-test and the U-test. In CD detection based on the data distribution, the distance function is commonly adopted to quantify the distribution relationship of the old and new data samples [24].

Step 6: detection of mining differences based on the occurrence of conceptual drift

In data mining, the underlying distribution of the data stream changes due to CD, so that the FP set changes accordingly. To better reflect recent changes, old concepts must be replaced immediately. In the problem of FP mining, the concept of an FP refers to the set of FPs, which is used as the target variable of the model description. Then, the change in FPs determines the difference between the two concepts. The concept of FP change is defined as follows:

Let F_{t_1} and F_{t_2} represent the set of frequent terms at time points t_1 and t_2 respectively. Thus, $F_{t_1}^+(t_2) = F_{t_2} - F_{t_1}$ is the set of new FPs at t_2 relative to t_1 , while $F_{t_1}^-(t_2) = F_{t_1} - F_{t_2}$ is the set of infrequent terms at t_2 but frequent items at t_1 . The rate of change of the set of frequent terms at t_2 relative to t_1 is defined as shown in Eq. 2.

$$FChange_{t_1}(t_2) = \frac{|F_{t_1}^+(t_2)| + |F_{t_1}^-(t_2)|}{|F_{t_1}| + |F_{t_1}^+(t_2)|} \quad (2)$$

where $|F_{t_1}|$ is the number of frequent item sets in the set F_{t_1} , and the rate of change is a value between 0 and 1.

The test defining this rate of change is $FChange$, and the return value $\lambda_{test}^{FChange}$ is calculated according to Eq. 3.

$$\lambda_{test}^{FChange} = \begin{cases} 0, & \text{The change rate is smaller than the given threshold} \\ 1, & \text{The change rate is bigger than the given threshold} \end{cases} \quad (3)$$

According to the defined threshold of this change value, if the $FChange$ calculated during the mining exceeds the threshold given by the user, the concept is considered to have changed.

Step 7: concept drift test index

Definition 3.1: The index of test CD R is defined and used to determine whether a data sample in the window has CD or not. It consists of two parts of the detection results, namely the detection results of data distribution

ϕ and the detection results of FP mining *FChange*. The corresponding calculations are shown in Eqs. 4 and 5.

$$R = \lambda_{test}^{FChange} + \lambda_{test}^F + 2\lambda_{test}^t + 2\lambda_{test}^U \quad (4)$$

$$\phi = \lambda_{test}^F + 2\lambda_{test}^t + 2\lambda_{test}^U \quad (5)$$

Specifically, ϕ is obtained according to the statistical test results of the Euclidean distance between windows, and the coefficient values of three tests are λ_{test}^F , λ_{test}^t and λ_{test}^U , and are used to distinguish the difference form of the distance distribution. When the Euclidean distance distribution between two windows passes the *F*-test and the *t*-test, the value of ϕ is 0; when the *F*-test but the *U*-test is passed, the value of ϕ is 1; when the *F*-test is passed, the value of ϕ is 2; when the *F*-test and the *U*-test fail, the value of ϕ is set to 3.

Step 8: window size adjustment

According to the sliding window algorithm design, a new window represents new information in the input data stream. Since our ultimate goal is to mine the set of frequent items on the data stream, after each new insertion pane, the amount of change *FChange* in the FP set is first determined. To improve the efficiency of the algorithm, we use the two sets F^+ and F^- to represent new FPs and new infrequent patterns and track the changes of the associated FP sets at checkpoint *CP*. The two sets are updated after each insertion pane.

As shown in Fig. 7, the process of window size change is as follows. Concept changes are detected based on the associated patterns of the inspection node (*CP*) and are used to determine whether CD has occurred within the window when new data are inserted. The position of the inspection node is not fixed and moves forward accordingly as concept changes are detected. The initial position of the inspection node is marked using the TID identifier of the last data in the initialized window. The CD test index *R* is calculated after each pane insertion.

Step 9: long and short embedded dual-layer window design and FPM

In this section, the window design for two common CD types in the data stream, mutant drift and gradient drift, is introduced. To address the characteristics of these two drift types, a dual-layer window structure is designed. The window structure is shown in Fig. 8

A long window is divided near its head to create space for a short window. Therefore, each time a data input is detected, the short window is given priority for detection. If no CD is detected in the short window, the data in the long window are examined. As shown in Fig. 8, the dual window has its head on the right and the tail on the left. The head of the long window corresponds to a short window, which is responsible for detecting abrupt CD in the data stream, while the long window is used to detect gradual CD.

The long window size is represented by $|LW|$, the short window size by $|SW|$, and the relative relationship between them is determined through λ , calculated as shown in Eq. 6. λ is a preset value that determines the shape of the log membership function used, and remains unchanged across the split window regardless of the window size.

$$|SW| = (1 - \lambda) \times |LW| \quad (6)$$

When the data stream begins to enter the window, after initialization in the dual-layer window, the parameter settings include the initial window size, the relative relationship of the long embedded window λ , the minimum support threshold δ , etc., as shown in Table 4. The FP-Growth algorithm is still used to mine the embedded long and short windows. The initial default window data represents the latest data concepts. Furthermore, in the dual-layer window model presented in this section, an attenuation module is designed for drift adaptation, which calculates the relative weighted support of an item by calculating the ratio of the weighted support count to the sum of all item counts.

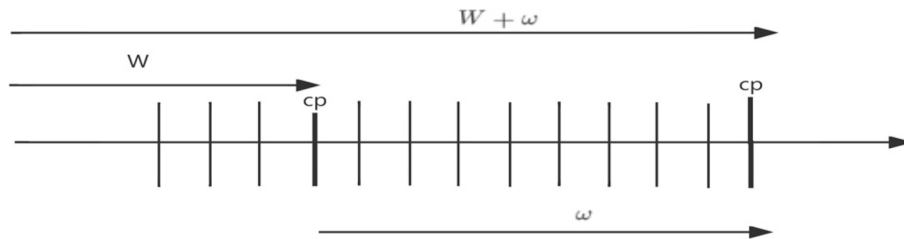


Fig. 7 Schematic diagram of window size change and check node change

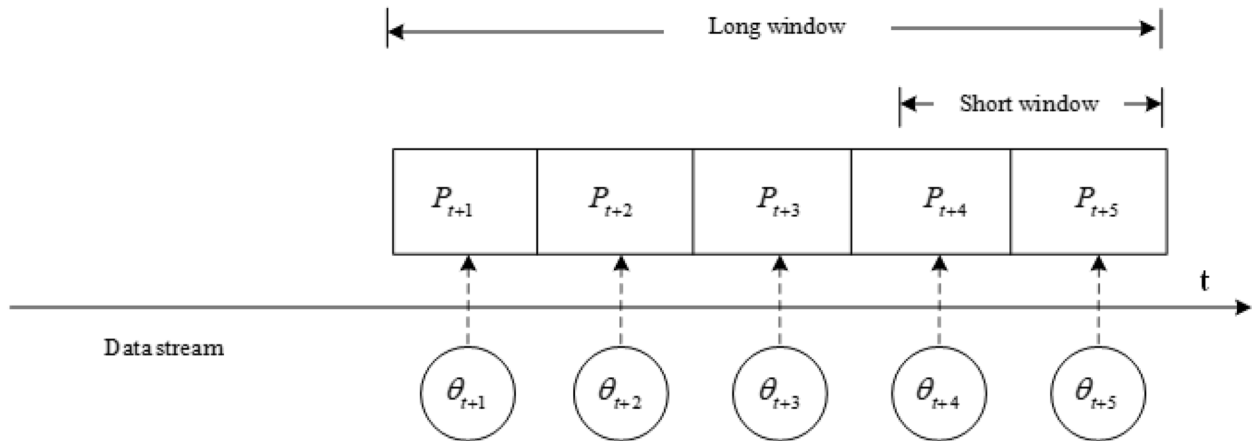


Fig. 8 Schematic diagram of the embedded dual-layer window

Step 10: concept drift type detection mechanism

According to the designed dual-layer embedded window, the input data streams are subjected to inspection using long and short windows. Equations 7 and 8 are used to calculate the concept change of FPs in the long and short windows, in order to determine if CD has occurred in the window.

If the FP set of two time nodes t_1 and t_2 are considered and SF_{t_1} and SF_{t_2} are used to represent short windows at these time points, then $SF_{t_1}^+(t_2) = SF_{t_2} - SF_{t_1}$ is the set of new sets of FPs with a short window at t_2 relative to t_1 and $SF_{t_1}^-(t_2) = SF_{t_1} - SF_{t_2}$ is the set of terms where short windows are infrequent at t_2 , but frequent at t_1 . Then, the rate of change in the FP set at the short window t_2 relative to t_1 is $SFChange_{t_1}(t_2)$. The calculation is shown in Eq. 7.

$$SFChange_{t_1}(t_2) = \frac{|SF_{t_1}^+(t_2)| + |SF_{t_1}^-(t_2)|}{|SF_{t_1}| + |SF_{t_1}^+(t_2)|} \quad (7)$$

where $|SF_{t_1}|$ is the number of items in the set SF_{t_1} , and the rate of change is a value between 0 and 1. Similarly, the formula for the rate of change $LFChange_{t_1}(t_2)$ at the long window t_2 versus t_1 is shown in formula 8:

$$LFChange_{t_1}(t_2) = \frac{|LF_{t_1}^+(t_2)| + |LF_{t_1}^-(t_2)|}{|LF_{t_1}| + |LF_{t_1}^+(t_2)|} \quad (8)$$

Table 5 Concept drift type determination of data in the window

Data drift type in the window	Drift was detected in the short windows	Drift was detected in the long window
mutant	yes	deny
Gradient type	deny	yes
No drift occurred	deny	deny

When data enter the window, the changes in FPs in the embedded short window are first calculated. If the rate of change of the FP set is greater than or equal to the given threshold, it is considered that the data in the short window have undergone a mutation, thus the presence of CD is considered to have occurred in the entire window. Similarly, if the rate of change of FPs in the long window exceeds the threshold, the data in the window are gradually drifting. The determination of CD types can be seen in Table 5.

Step 11: concept drift adaptation

Based on the design of the CD detection mechanism, when CD is detected in the short window, mutant CD occurs. For such drift, the drift data in the window are eliminated, the detection node CP_S of the short window is moved forward, and a new window ω is formed to continue mining.

Figure 9 shows the process of eliminating the effects of old data and forming a new window when mutant CD has occurred within the window. Consistent with the previous design, the data pane is inserted from left to right into the current window. Concept changes within a short window are measured against the detection node CP_S , and the amount of changes to the FP set $SFChange$ is calculated after each insertion pane. As shown in Fig. 9, after one or more data panes are inserted into the window, if a concept change is detected in the inspection node of a short window, the data information between the current checkpoint and the previous check node in the window is given a weight value of 0, which will eliminate the impact of mutant data. Then, a new window ω is formed and the checkpoint is moved to a new node CP_S where concept change is detected. If no concept changes are detected within the short window, the FP changes within the long window.

Table 6 Introduction to the weighted frequent pattern tree node name

Name	Meaning
Item-name	The name of the project
Count	Record the sum of the weights of the transaction data set to which all items belong from the root node to the path of that node
Pnode	Parent node pointer
Snode	Pointer to the next project node with the same project name
hlink	Pointing to the first node with the same value as the Hitem-name domain
Hitem-name	Project name of the header pointer table

process of detecting CD and adaptation drift is iterated; in row 5, new data are inserted in the window; in line 6, the corresponding FP set is updated. Lines 7–14 process the item set that becomes frequent or infrequent after the new pane insertion. CD is detected on line 15, the window is determined and the decay weight is assigned, and then the FP mining results are updated.

Experimental results and analysis

Based on the design of several modules, in this section a dual-layer variable window FP mining algorithm is proposed for CD type detection, the DLVSW-CDTD algorithm. The pseudo-code for the algorithm is shown below. All the steps described above are represented in this pseudo-code as follows, and some of the steps use function calls instead of detailed procedures. The pseudo-code for DLVSW-CDTD algorithm.

Introduction of the experimental environment and data set

Experimental environment

A random transactional data generator provided in the open source data mining library SPMF [25] was used to generate a comprehensive transactional data set, as follows.

The meaning of letters in the name of the data set is: D: number of sequences in the data set; T: average number of item sets in each transaction datum; I: the average size of the item set of a potential frequent sequence; K: abbreviation of the number 1000; for example, Article 10,000 is denoted as Article 10 K in this paper. The dataset generated by this section is T20I6D100K. The main software environment is shown in Table 7.

Open source data mining library SPMF

We use a random transactional data generator provided in an open source data mining library SPMF [25]

Table 7 Main software environment and the version

Name	Edition
OS	Window10
PyCharm	6.8.0
matplotlib	2.2.3
pandas	0.25.3

to generate a comprehensive transactional data set. The relevant information of the data set is introduced as follows. I: the average size of the item set in the potential frequent sequence, K: the abbreviation of the number 1000, such as the data in Article 10,000, are represented by the data in Article 10 K in this paper. The generated data set in this section is T20I6D100K.

Performance verification of the DLVSW-CDTD algorithm

In the first set of experiments, the DLVSW-CDTD Algorithm was evaluated for its ability to achieve two desired functions: detecting CD in the data stream and adjusting the window size after detecting the CD. To demonstrate how the DLVSW-CDTD Algorithm adjusted the window size following a CD occurrence, a dataset with a predefined CD at a specific location was used for the experiment. It was chosen to highlight the algorithm's effectiveness in adapting the window size in response to detected CD.

Dataset setting

For the frequent and infrequent items in the T20I6D200K-X dataset generated using the open source data mining library SPMF, the minimum support was set to 2%. Then, 50% of the frequent and infrequent items in the dataset were selected and a new set of items was formulated, which was inserted at the end of the first dataset to synthesize a new data set with a clear CD point.

Related parameter setting and description

As per the DLVSW-CDTD algorithm's design, the first check node (i.e. the data of article 20 K) was located after the window initialization. When the algorithm detects a change in the concept of the data, the checkpoint will move to a new location. According to the dataset T20I6D200K-X's design, in the section 70 K data, there is a CD, so the R-value here must be greater than or equal to 3, and it is obvious from Fig. 12 that the experimental results were as expected. At the 70 K data, the R-value was 3, the algorithm considered the first 20 K data from the initial node to the checkpoint as drift (i.e. expired) data, removed

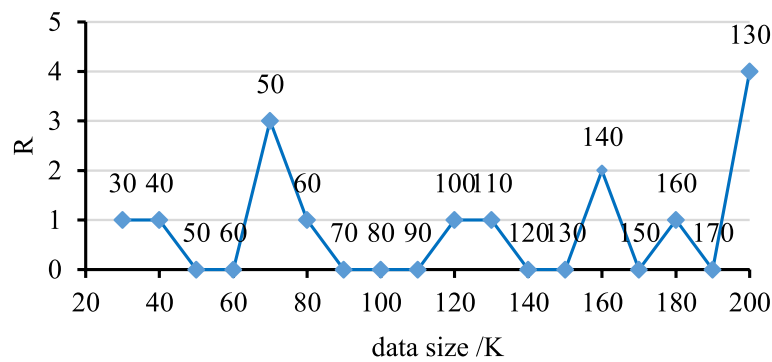


Fig. 12 Performance verification of the Algorithm (initial pane 20 K, pane size 10 K)

them from the window, and adjusted the window size to 50 K. In addition, the checkpoint was moved to the point where the conceptual change was detected (i.e. Article 70 K data). Similarly, after Article 200 K data, a conceptual change was detected again, and the window was re-sized to 130 K, marked in bold in Fig. 12.

Window size change

To verify whether the value of the initial window and pane sizes will affect the CD detection effect and the process of window size adjustment, different values for the initial window and pane sizes were used to perform the above experiments without changing the minimum support and correlation thresholds.

The initial window and pane size were changed simultaneously to 10 K and 5 K, respectively, and the experimental effect is shown in Fig. 13. Then, only the initial window size was changed to 40 K while the pane size remained unchanged to 10 K. The experimental results are shown in Fig. 14. Finally, the initial window size was set to 20 K, and the pane size was changed to 5 K. The experimental effect is shown in Fig. 15.

As shown in Figs. 13, 14 and 15, for the same dataset, the size of the initial window and the data pane did not affect the CD detection effect. Both conceptual changes were detected in the data in Articles 70 K and 200 K, and the final window size used by the algorithm was the same.

In conclusion, the size of the initial window and pane will not affect the CD detection and window size adjustment of the DLVSW-CDTD algorithm; the algorithm will adaptively adjust the size of the window with the CD. Compared to fixed-size sliding window approaches, the set of FP obtains reflects the latest concept in the data.

Threshold and correlation analysis

For the experiments presented in this section, the BMS-Webview-1 dataset was used. For all experiments in this

data set, taking into account relevant literature studies, the pane size, initial window size, minimum support and threshold $FChange$ were set to 5 K, 20 K, 2% and 0.1, respectively, and then different change thresholds were used to show the effect of this parameter on concept change detection and window size.

From Figs. 16, 17, 18 and 20, it can be seen that for four different CD change thresholds, the window size changes as the input data increase. In Figs. 16 and 17, the concept change threshold was set to 2, and it can be seen that the data stream was detected at 45 K, 65 K, 65 K and 85 K, and the concept test values exceeded 2. The size of the window was adjusted to new values, i.e. to 25 K, 20 K and 20 K, respectively. In Fig. 18, the CD test threshold was 3, and the CD was detected in the seventh pane and the last pane, and the window size decreased to 35 K and 30 K, respectively. Finally, in Fig. 19, no conceptual change was detected during the data mining process with a test-of-concept change threshold of 4. This is due to the high setting of the concept change threshold relative to the data stream. Thus, as shown in Fig. 20, the window size continuously increases as the pane inserts.

Since the DLVSW-CDTD algorithm performs two functions during CD detection section, namely data distribution analysis and mining result detection, R consists of two parts, namely the FP change index $FChange$ and the test index ϕ of the data distribution. The value of R determines whether CD has occurred. Therefore, the correlation between the two parameters and R was analyzed.

As can be seen from Fig. 20, compared with the FP change index $FChange$, the value of ϕ , i.e. the test index of the data distribution, affects the R -value more, which is also in line with the experimental expectation. When there are more input data and FP tends to saturate, the change of $FChange$ will become smaller.

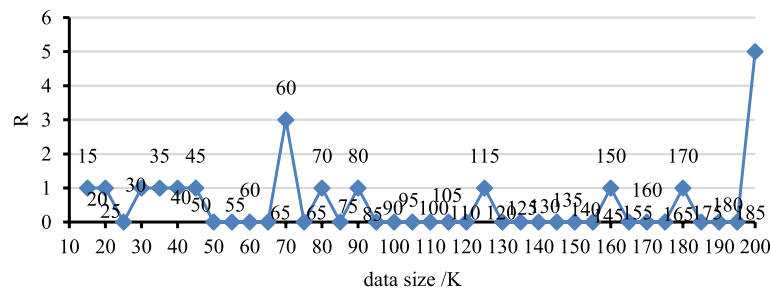


Fig. 13 Performance verification of the DLVSW-CTD Algorithm (initial pane 10 K, pane size 5 K)

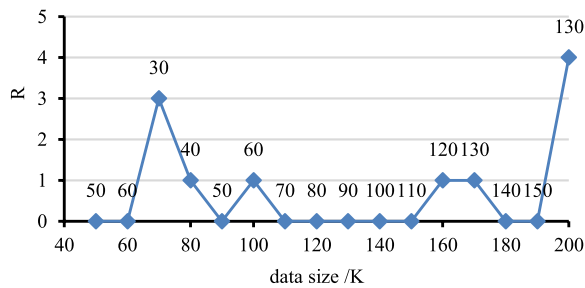


Fig. 14 Performance verification of the DLVSW-CTD Algorithm (initial pane 40 K, pane size 10 K)

Dual-layer window concept drift detection

In this experiment, the proposed dual-layer window detection algorithm DLVSW-CTD is compared to other algorithms which can timely detect mutant and gradient CD. The data in the BMS-Webview-1 were used for the comparison. Because part of the detection steps of the DLVSW-CTD algorithm directly call the CD detection part of the variable sliding window, the size of the window must change with the detected CD, thus directly verifying whether the proposed DLVSW-CTD algorithm can detect two types of CD.

As shown in Fig. 21, as the data input size increases, the length of the embedded window of the FP change rate is also changing. The point beyond the threshold is marked in red, and corresponds to the mutant and

gradient CD. According to the design of the algorithm, the corresponding window size must change; the effect is shown in Fig. 22.

Execution time

To further analyze the performance of the proposed algorithm, this execution time experiment was designed to test the running time of each function of the proposed DLVSW-CTD Algorithm. The data volume was fixed to 60 K, or 60,000 data samples. The test content included the proportion of time spent on different types of CD detection and processing of dual-layer windows, the time ratio of frequent update modes, and the proportion of other operation times. The experimental results are shown in Table 8.

In conclusion, in the overall operation of the DLVSW-CTD algorithm, the time consumption is mainly concentrated in the update of FP modes. The main purpose of the algorithm is to construct the weighted FP tree, while the time consumed in data reading, CD detection and drift adaptation to the modules is small. These processing steps are characterized by an overall fast calculation speed and a high processing efficiency, so the time proportion spent on them was less than 10%. In addition, the absolute operation time of each module was also relatively stable, and it did not produce relatively large fluctuations as the size of the initial window changed.

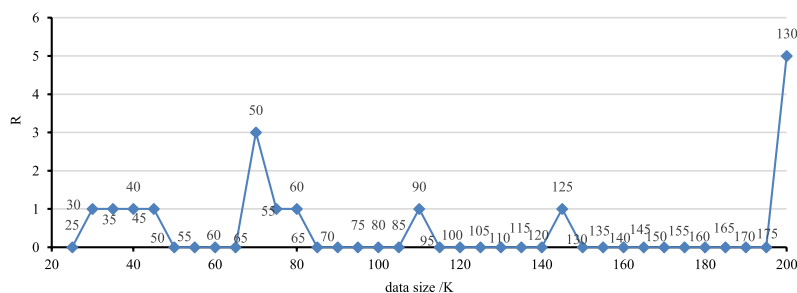


Fig. 15 Performance verification of the DLVSW-CTD Algorithm (initial pane 20 K, pane size 5 K)

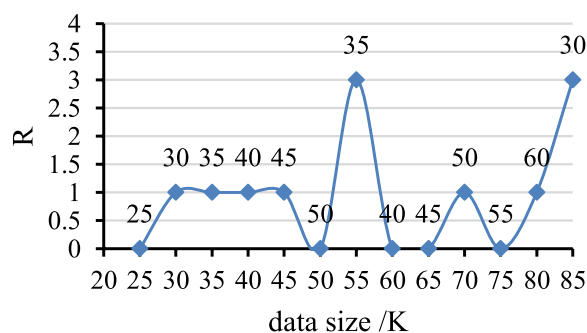


Fig. 16 The test of concept index R threshold is 1

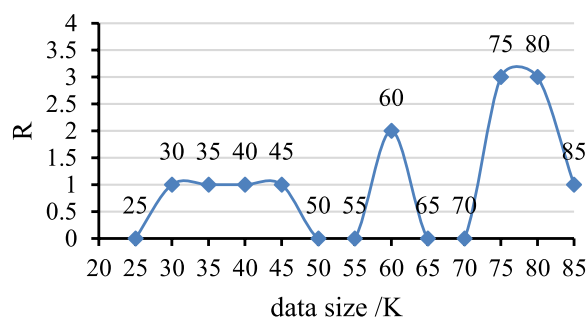


Fig. 17 The test of concept index R threshold is 2

Comparative analysis

The DLVSW-CDTD algorithm deals with the CD problem by first identifying the CD in the data stream through a multi-type CD detection technique. After detecting the CD, the algorithm will use a dual-layer

variable sliding window strategy to smooth the data stream. This strategy can effectively handle the real-time and volatile nature of the data stream while still maintaining the accuracy of CD detection.

In the comparison analysis, the same amount of transactional data was used and the same minimum support was set to test the different algorithms. The DLVSW-CDTD algorithm was compared with the variable-moment algorithm with a variable window, and the FP-Growth algorithm with a fixed sliding window size. The initial window size was set to 10 K, while the unified step size and the pane size were set to 5 K. The experimental results are shown in Figs. 23 and 24. With the continuous data input, the running time was increased.

Conclusion and future work

Summary of research work

The experimental results demonstrate that compared to existing data stream mining algorithms based on the window model, the proposed DLVSW-CDTD algorithm detects different types of CD in the data stream effectively. It achieves this by embedding a dual-layer window that allows for window size adjustment and applying varying attenuation weights to the drifting data, thereby achieving drift adaptation. Additionally, the time proportions of each module in the DLVSW-CDTD algorithm fluctuate by no more than 0.8%. Overall, the DLVSW-CDTD algorithm exhibits certain advantages in terms of time complexity and memory consumption. Furthermore, it maintains stable performance, and is unaffected by the initial window size or the need for processing large amounts of data.

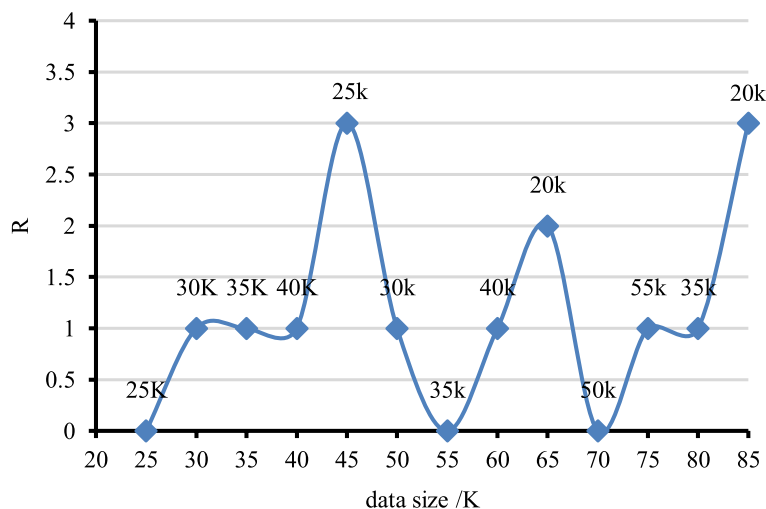


Fig. 18 The concept index R threshold is 3

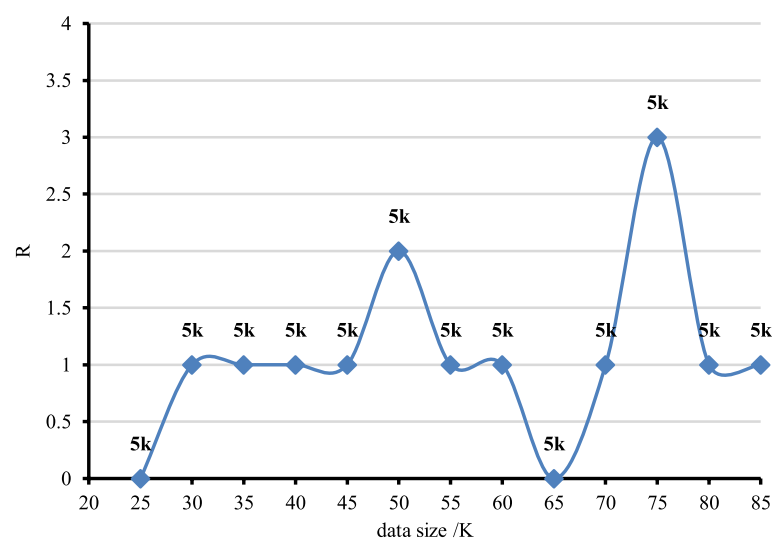


Fig. 19 The concept test index R threshold is 4

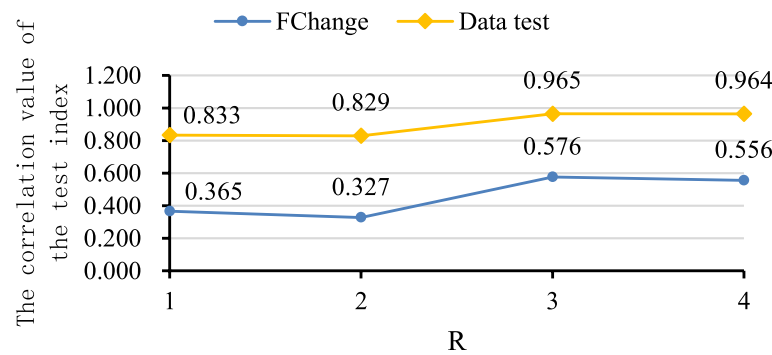


Fig. 20 Correlation analysis of $FChange$ and ϕ with R

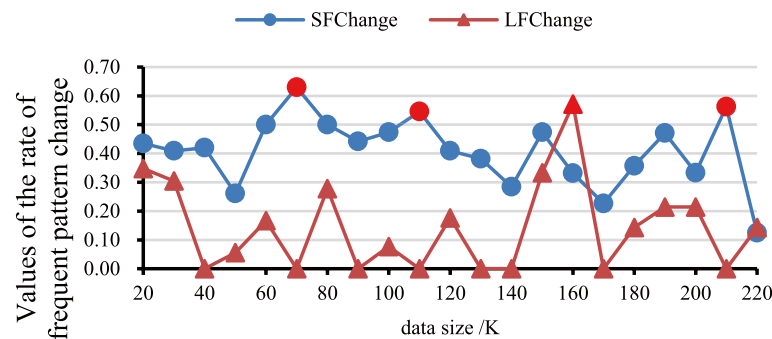


Fig. 21 Frequent pattern changes of nested dual-layer windows

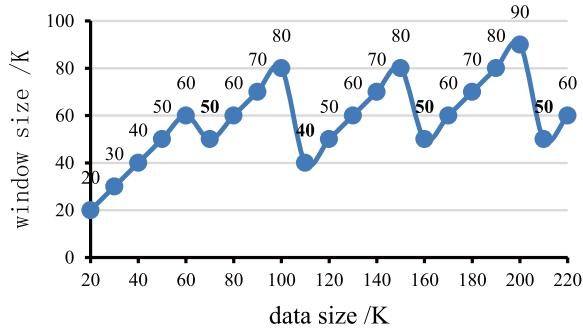


Fig. 22 Window size variation

Table 8 Running time proportion of each function of the algorithm under different initial window sizes

Time ratio	10K	20 K	30 K	40 K	50 K
Reading data	5.7%	6.3%	5.2%	5.7%	5.4%
Update the window	0.3%	0.3%	0.3%	0.1%	0.3%
Short-window concept drift detection	0.2%	0.1%	0.1%	0.0%	0.1%
Handling of the mutant concept drift	8.9%	8.2%	8.9%	9.3%	8.9%
Update frequent pattern	83.9%	84.3%	84.7%	84.1%	84.5%
Long-window concept drift detection	0.1%	0.1%	0.1%	0.0%	0.0%
Handle the gradient concept drift	0.9%	0.8%	0.7%	0.8%	0.8%

Future research work

There are several potential research directions to further explore in this field. Future studies could focus on enhancing the DLVSW-CDTD algorithm through the incorporation of more advanced techniques for cloud computing and security information detection. Some possible directions to consider include:

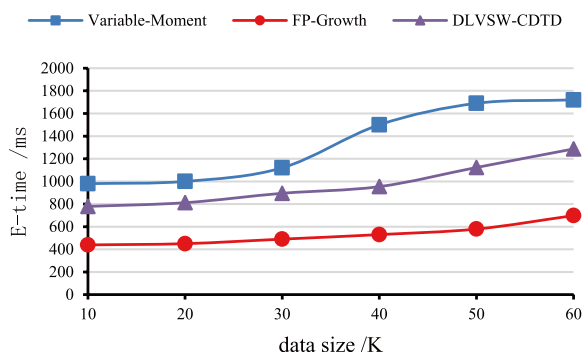


Fig. 23 Comparison of the execution times for the different algorithms

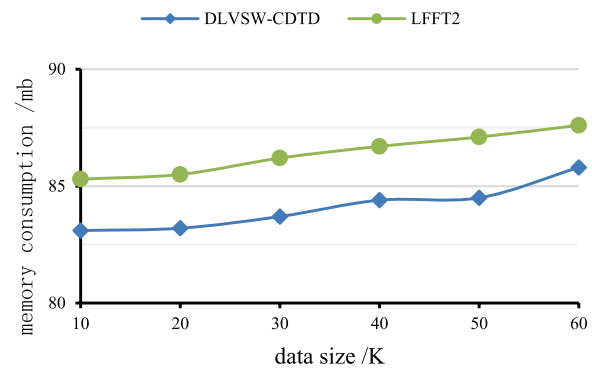


Fig. 24 Comparison of the memory consumption of the algorithm

- (1) **Cloud Computing:** Investigate how the DLVSW-CDTD algorithm can be optimized for cloud computing environments. Techniques such as distributed computing, parallelization, and resource allocation strategies can be explored to improve the algorithm's scalability and performance in handling large-scale data streams.
- (2) **Machine Learning Integration:** Machine learning algorithms can be integrated within the DLVSW-CDTD algorithm to enhance its capabilities. Advanced machine learning techniques, such as deep or reinforcement learning, fuzzy set theory [52, 53] can be adopted to improve the accuracy and effectiveness of CD detection and adaptation.

Acknowledgements

We would like to thank MogoEdit (<https://www.mogoedit.com>) for its English editing during the preparation of this manuscript.

Authors' contributions

Jing Chen: Writing original draft, Review response, Commentary, Revision.
Shengyi Yang and Ting Gao: Writing original draft, Commentary. Yue Ying:
Writing original draft. Tian Li: Commentary, Revision, Validations. Peng Li: Con-
ceptualization, Funding acquisition, Methodology, Supervision, Review.

Funding

The research is sponsored by the National Natural Science Foundation of P. R. China (No. 62102194, No. 62102196); Natural Science Foundation of Inner Mongolia Autonomous Region of China (No.2022MS06010), Natural Science Research Project of Department of Education of Guizhou Province (No.QJJ2022015); Scientific Research Project of Baotou Teachers' College (BSYKY2021-ZZ01, BSYHY202212, BSYHY202211, BSGJ2307).

Availability of data and materials

The data presented in this study are available from the open source data mining library SPMF [25].

Declarations

Competing interests

The authors declare no competing interests.

Received: 10 September 2023 Accepted: 7 December 2023
Published online: 12 February 2024

References

- Bao G, Guo P (2022) Federated learning in cloud-edge collaborative architecture: key technologies, applications and challenges[J]. *Journal of Cloud Computing* 11(1):94
- Ismaeel S, Karim R, Miri A (2018) Proactive dynamic virtual-machine consolidation for energy conservation in cloud data centres[J]. *Journal of Cloud Computing* 7(1):1–28
- Wang F, Wang L, Li G, Wang Y, Lv C, Qi L (2022) Edge-Cloud-enabled Matrix Factorization for Diversified APIs Recommendation in Mashup Creation. *World Wide Web Journal* 25(5):1809–1829
- Yang Y, Ding S, Liu Y, Meng S, Chi X, Ma R, Yan C (2022) Fast wireless sensor for anomaly detection based on data stream in an edge-computing-enabled smart greenhouse. *Digital Commun Netw* 8(4):498–507
- Zhangyang Xu, Zhu D, Chen J, Baohua Yu (2022) Splitting and placement of data-intensive applications with machine learning for power system in cloud computing. *Digital Commun Netw* 8(4):476–484
- Al-Ghuwairi AR, Sharrab Y, Al-Fraihat D et al (2023) Intrusion detection in cloud computing based on time series anomalies utilizing machine learning[J]. *Journal of Cloud Computing* 12(1):127
- Xin Su, Jiang Su, Choi D (2022) Location privacy protection of maritime mobile terminals. *Digital Commun Netw* 8(6):932–941
- Peng LI, Xiaotian YU, He XU et al (2021) Secure Localization Technology Based on Dynamic Trust Management in Wireless Sensor Networks. *Chin J Electron* 30(4):759–768
- Miao Y, Bai X, Cao Y, Liu Y, Dai F, Wang F, Qi L, Dou W (2023) A Novel Short-Term Traffic Prediction Model based on SVD and ARIMA with Blockchain in Industrial Internet of Things. *IEEE Internet Things J*. <https://doi.org/10.1109/JIOT.2023.3283611>
- Yang N, Yang L, Du X et al (2023) Blockchain based trusted execution environment architecture analysis for multi-source data fusion scenario[J]. *Journal of Cloud Computing* 12(1):1–16
- Kong L, Li G, Rafique W, Shen S, He Q, Khosravi MR, Wang R, Qi L (2022) Time-aware Missing Healthcare Data Prediction based on ARIMA Model. *IEEE/ACM Trans Comput Biol Bioinf*. <https://doi.org/10.1109/TCBB.2022.3205064>
- Mousavi SN, Chen F, Abbasi M, Khosravi MR, Rafiee M (2022) Efficient pipelined flow classification for intelligent data processing in IoT. *Digital Commun Netw* 8(4):561–575
- Kong L, Wang L, Gong W, Yan C, Duan Y, Qi L (2022) LSH-aware Multitype Health Data Prediction with Privacy Preservation in Edge Environment. *World Wide Web Journal* 25(5):1793–1808
- Yang Y, Yang X, Heidari M, Srivastava G, Khosravi MR, Qi L (2022) ASTREAM: Data-Stream-Driven Scalable Anomaly Detection with Accuracy Guarantee in IIoT Environment. *IEEE Transactions on Network Science and Engineering*. <https://doi.org/10.1109/TNSE.2022.3157730>
- Wang F, Li G, Wang Y, Rafique W, Khosravi MR, Liu G, Liu Y, Qi L (2022) Privacy-aware Traffic Flow Prediction based on Multi-party Sensor Data with Zero Trust in Smart City. *ACM Trans Internet Technol*. <https://doi.org/10.1145/3511904>
- Wang F, Zhu H, Srivastava G, Li S, Khosravi MR, Qi L (2022) Robust Collaborative Filtering Recommendation with User-Item-Trust Records. *IEEE Transactions on Computational Social Systems* 9(4):986–996
- Qi L, Lin W, Zhang X, Dou W, Xiaolong Xu, Chen J (2023) A Correlation Graph based Approach for Personalized and Compatible Web APIs Recommendation in Mobile APP Development. *IEEE Trans Knowl Data Eng* 35(6):5444–5457
- Chen N, Zhong Q, Liu Y et al (2023) Container cascade fault detection based on spatial-temporal correlation in cloud environment[J]. *Journal of Cloud Computing* 12(1):59
- Wares S, Isaacs J, Elyan E (2019) Data stream mining: methods and challenges for handling concept drift[J]. *SN Applied Sciences* 1:1–19
- Rabiu I, Salim N, Da'u A et al (2020) Recommender system based on temporal models: a systematic review[J]. *Applied Sciences* 10(7):2204
- Yang C, Cheung Y, Ding J et al (2021) Concept drift-tolerant transfer learning in dynamic environments[J]. *IEEE Transactions on Neural Networks and Learning Systems* 33(8):3857–3871
- Liu Z, Godahewa R, Bandara K et al (2023) Handling Concept Drift in Global Time Series Forecasting[J] (arXiv preprint arXiv:2304.01512)
- Fdez-Riverola F, Iglesias EL, Díaz F et al (2007) Applying lazy learning algorithms to tackle concept drift in spam filtering[J]. *Expert Syst Appl* 33(1):36–48
- Gulla JA, Solskinnsbakk G, Myrseth P et al (2011) Concept signatures and semantic drift[C]. *Web Information Systems and Technologies: 6th International Conference, WEBIST 2010, Valencia, Spain, April 7–10, 2010, Revised Selected Papers* 6. Springer, Berlin Heidelberg, pp 101–113
- Turkov P, Krasotkina O, Mottl V, et al (2016) Feature selection for handling concept drift in the data stream classification[C]. *Machine Learning and Data Mining in Pattern Recognition: 12th International Conference, MLDM 2016, New York, NY, USA, July 16–21, 2016, Proceedings*. Springer International Publishing, pp 614–629
- Ruano-Ordas D, Fdez-Riverola F, Mendez JR (2018) Concept drift in e-mail datasets: An empirical study with practical implications[J]. *Inf Sci* 428:120–135
- Ding F, Luo C (2019) The entropy-based time domain feature extraction for online concept drift detection[J]. *Entropy* 21(12):1187
- McKay H, Griffiths N, Taylor P et al (2020) Bi-directional online transfer learning: a framework[J]. *Ann Telecommun* 75:523–547
- Gama J, Medas P, Castillo G et al (2004) Learning with drift detection[C]. *Advances in Artificial Intelligence—SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29–October 1, 2004. Proceedings* 17. Springer, Berlin Heidelberg, pp 286–295
- Baena-Garcia M, del Campo-Ávila J, Fidalgo R, Bifet A, Gavalda R, Morales-Bueno R (2006) Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams* Vol. 6, pp. 77–86
- Hulten G, Spencer L, Domingos P (2001) Mining Time-Changing Data Streams[C]. *The Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp 97–106
- Liang NY, Huang GB, Saratchandran P et al (2006) A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks[J]. *IEEE Trans Neural Networks* 17(6):1411–1423
- Jie L, Anjin L, Fan D et al (2019) Learning under Concept Drift: A Review[J]. *IEEE Trans Knowl Data Eng* 31(12):2346–2363
- Ruihua C, Xiaolong Qi, Yanfang L (2023) Online integrated adaptive algorithm for concept drift data flow [J]. *Journal of Nanjing University (Natural Science)* 59(1):134–144
- Cheng H, Huaiping J, Bin W (2023) An integrated adaptive soft measurement method based on spatiotemporal local learning [J]. *J Instrum* 44(1):231–241
- Xiulin Z, Peipei L, Xuegang H et al (2021) Semi-supervised Classification on Data Streams with Recurring Concept Drift and Concept Evolution[J]. *Knowl-Based Syst* 215:1–16
- Anjin L, Guangquan Z, Jie L (2017) Fuzzy Time Windowing for Gradual Concept Drift Adaptation[C]. *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. 1–6
- Abdualrhman M, Padma MC (2019) Deterministic Concept Drift Detection in Ensemble Classifier Based Data Stream Classification Process[J]. *International Journal of Grid and High Performance Computing* 11(1):29–48
- Shuliang X, Lin F, Shenglan L et al (2020) Self-adaption Neighborhood Density Clustering Method for Mixed Data Stream with Concept Drift[J]. *Eng Appl Artif Intell* 89:1–14
- Meng H, Zhihai W, Jian D (2016) A frequent pattern decision tree deals with variable data streams [J]. *J Comput Sci* 39(8):1541–1554
- Aumann Y, Lindell Y (2003) A Statistical Theory for Quantitative Association Rules[J]. *Journal of Intelligent Information Systems* 20(3):255–283
- Chen H, Shu LC, Xia J et al (2012) Mining frequent patterns in a varying-size sliding window of online transactional data streams[J]. *Inf Sci* 215:15–36
- Deypir M, Sadreddini MH, Hashemi S (2012) Towards a Variable Size Sliding Window Model for Frequent Itemset Mining over Data Streams[J]. *Comput Ind Eng* 63(1):161–172
- Pesaranghader A, Viktor HL, Paquet E (2018) McDiarmid Drift Detection Methods for Evolving Data Streams[C]. *International Joint Conference on Neural Networks (IJCNN) 2018*:1–9

45. Iwashita AS, Papa JP (2019) An Overview on Concept Drift Learning[J]. *IEEE Access* 7:1532–1547
46. Bin L, Guanghui Li (2021) A notional drift data flow classification algorithm based on the McDiarmid bound [J]. *Computer Science and Exploration* 15(10):1990–2001
47. Zhiqiang C, Han Meng Wu, Hongxin, et al (2023) A conceptual drift detection method for segment-weighting [J]. *Computer Applications* 43(3):776–784
48. Barros R, Santos S (2019) An Overview and Comprehensive Comparison of Ensembles for Concept Drift[J]. *Information Fusion* 52:213–244
49. Husheng G, Hai L, Qiaoyan R et al (2021) Concept Drift Type Identification Based on Multi-Sliding Windows[J]. *Inf Sci* 585:1–23
50. Mao L, Dongbo Z, Yuanyuan Z (2014) A new method for drift detection based on the concept of overlapping data window distance measure [J]. *Computer Applications* 34(2):542–545
51. Lu J, Liu A, Song Y et al (2020) Data-driven Decision Support under Concept Drift in Streamed Big Data[J]. *Complex & Intelligent Systems* 6(1):157–163
52. Chen J, Li P, Fang W, et al (2021) Fuzzy Frequent Pattern Mining Algorithm Based on Weighted Sliding Window and Type-2 Fuzzy Sets over Medical Data Stream[J]. *Wireless Commun Mobile Comput* 1–17
53. Y Yin, P Li, J Chen (2023) A Variable Sliding Window Algorithm Based on Concept Drift for Frequent Pattern Mining Over Data Streams[C]. 2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 818–825

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.