Hindawi
Mobile Information Systems
Volume 2021, Article ID 3103053, 11 pages
https://doi.org/10.1155/2021/3103053

Research Article

Stream Classification Algorithm Based on Decision Tree

Jinlin Guo,[1] **Haoran Wang**,[1] **Xinwei Li**,[1] **and Li Zhang**[2]

[1]*College of Systems Engineering, National University of Defense Technology, Changsha 410000, China*
[2]*Software College, Northeastern University, Shenyang 110000, China*

Correspondence should be addressed to Jinlin Guo; gjlin99@nudt.edu.cn

Received 24 September 2021; Revised 27 October 2021; Accepted 1 November 2021; Published 21 December 2021

Academic Editor: Fazlullah Khan

Due to the rise of many fields such as e-commerce platforms, a large number of stream data has emerged. The incomplete labeling problem and concept drift problem of these data pose a huge challenge to the existing stream data classification methods. In this respect, a dynamic stream data classification algorithm is proposed for the stream data. For the incomplete labeling problem, this method introduces randomization and iterative strategy based on the very fast decision tree VFDT algorithm to design an iterative integration algorithm, and the algorithm uses the previous model classification result as the next model input and implements the voting mechanism for new data classification. At the same time, the window mechanism is used to store data and calculate the data distribution characteristics in the window, then, combined with the calculated result and the predicted amount of data to adjust the size of the sliding window. Experiments show the superiority of the algorithm in classification accuracy. The aim of the study is to compare different algorithms to evaluate whether classification model adapts to the current data environment.

1. Introduction

With the development of the Internet, sensors, and the Internet of Things, massive streaming data has emerged. Streaming data is data that is continuously generated by different sources. Such data should be processed incrementally using stream processing techniques without having access to all of the data. It includes a variety of data formats, such as log files generated by web applications, online shopping data, traffic monitoring, social networking site information, and geospatial and meteorological satellite data. These stream data imply a large amount of information that is instructive for real-world decision-making. Therefore, many scholars analyze the data to obtain useful information which guides people to make scientific decisions, such as e-commerce platform personalized real-time recommendations, stock market monitoring, network intrusion, abnormal fraud monitoring, most smart device applications, traffic monitoring, and real-time motion analysis, etc. Streaming data processing is advantageous for most scenarios where dynamic data is generated continuously. As an important branch of data mining, the classification problem has important practical significance in the fields of financial credit rating, prevention of telecom fraud, and detection of network intrusion [1]. According to the characteristics of the stream data, the analysis of classification technology has high commercial value and research significance.

Stream data has the characteristics of real-time generation, fast arrival, large amount, and difficulty in repeated acquisition [2]. If traditional classification mining models and algorithms are still used for processing, a large amount of useful information will be lost. Some existing data mining schemes and algorithms fail to fully consider the characteristics of stream data and practical application scenarios, such as concept drift, incomplete labeling, and uneven data flow rate. Some studies assume that all flow data arriving is fully labeled and these labels can be used directly [1]. This assumption is difficult to establish in many practical applications, especially in the areas of telecom fraud and intrusion detection. In addition, as time changes, the underlying concepts in the data stream will change. Many detection algorithms are complex and consume a lot of space and resources. In the stream data processing algorithm based on the sliding window, the sliding window size is fixed or only changes with the concept drift. However, the stream characteristics are not considered, so that when the data flow

rate is very fast or very slow, the instant processing is not obtained, which affects the efficiency of the stream data processing. Therefore, it is of great research value and practical significance to design classification models and methods that can be applied to actual scenes more efficiently.

Aiming at the above problems, this paper proposes a real-time streaming data dynamic classification algorithm that adopts decision tree as the base classification model and combines the idea of an integrated classification algorithm to change the mode of a single classification model and updates the classification model periodically; in the detection of concept drift, the degree of conceptual drift is detected by calculating the difference between the data of front and end part in the sliding window. When the concept drift occurs, the size of the sliding window is adjusted according to the degree of the concept drift and, at the same time, combined with the real-time data traffic situation to adjust the sliding window again, to make data processing more timely and effective.

Section 2 introduces the classification algorithm and the related research of concept drift and related definitions. Section 3 introduces the detailed framework of the algorithm and the specific implementation process of the internal algorithm. Section 4 analyzes the relevant part of the experimental part. Section 5 summarizes full text and the shortcomings of the algorithms and the prospects for future related work.

2. Related Work and Theoretical Basis

2.1. Related Research. At present, the classification algorithms of streaming data environments are mostly researched and designed on some basic machine learning algorithms, including support vector models, Bayesian models (a Bayesian model is a statistical model where you use probability to represent all uncertainty within the model, both the uncertainty regarding the output but also the uncertainty regarding the input), neural network models (it is a simplified model life human brain having input layer, hidden layer, and output layer), and decision tree models (a decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility). Decision tree models are faster and more efficient than other batch algorithms when dealing with stream data.

One of the most influential algorithms is the decision tree-based fast data stream classification algorithm VFDT [3], which introduces Hoeffding Bound [4] to select the best splitting property. Each instance is processed in a constant time. In the classification process, the data only needs to be scanned once. This algorithm has a higher processing speed. However, the VFDT does not perform well when processing continuous attributes. The VFDTc [5] algorithm is optimized based on the VFDT. By introducing a discretization algorithm, the capability of VFDT is extended to process the continuous attributes and improves the classification accuracy. After the combination of the decision tree and other algorithms, the classification performance is further improved. For example, the combination of decision tree and frequent pattern mining proposes a frequent pattern decision tree algorithm that processes stream data and my constraint-based closure frequent patterns [6]. Then, the classification decision tree is established. According to the observed data set in the order domain and the time domain, based on the C4.5, the inductive learning time data classification is proposed [7].

As machine learning has received much attention, integrated learning algorithms have been welcomed as a technology to improve classification accuracy from machine learning. The integrated classification integrates multiple weak classifiers or multiple classification algorithms to produce an effectively combined classifier which greatly improves the classification accuracy of classical algorithms on static data. WE is an integrated classification framework [8]. When the data block arrives at a certain time t, choose a classification learning algorithm to learn the data block independently. N classification models are obtained. Each classification model is given weight according to different methods and obtains the integrated classification model. The SEA algorithm (Schoof–Elkies–Atkin algorithm) uses the idea of integrated classification [8]. It divides the training data into ordered data blocks of equal size according to the data arrival time and constructs a base classifier for each piece of data. When the classifier reaches the maximum capacity, it updates the base classifier with a heuristic replacement mechanism. Wang et al. designed a generalized framework for data stream classification named weighted bagging [9]. It builds a basic classifier based on C4.5 algorithm, used in data mining as a Decision Tree Classifier which can be employed to generate a decision, based on a certain sample of data, and K-based classifiers build an integrated classifier by weighted voting. In the implementation process, the memory consumption of these methods is too large, which reduces the processing rate.

The concept drift problem makes stream data classification face a lot of challenges. Therefore, many researchers have studied the concept drift problem and created many solutions.

The algorithms [10, 11] detect the concept drift by calculating the information entropy of the old and new data blocks. Observing the difference between the entropy values, [12, 13] judge the conceptual drift by building clusters. For example, the clustering data blocks are obtained by the K-Means algorithm. Calculating the distance between cluster centers and determining whether concept drift occurs, the label propagation method makes full use of the class label information and the internal structure information from the cluster to infer the class label of each cluster.

In addition to the above algorithms, sliding windows are one of the important mechanisms for dealing with conceptual drift problems. Based on VFDT, CVFDT [14] algorithm adds a sliding window to update the generated decision tree according to the sample data flowing into the window. As more and more algorithms evolve, researchers use dual-window and multiwindow mechanisms. The (Adaptive Windowing) ADWIN method arbitrarily divides the window W into two subwindow [15]. When the mean difference between the two subwindows exceeds a certain

limit, it is considered that the expected values of the windows are inconsistent. The concept drift occurs and the window shrinks. Based on the ADWIN algorithm, HAT [16] learns the model from the changing data stream, avoiding the problem of sliding window size selection. The (Double-Window-Based Classification) DWCDS [17] algorithm adopts a two-layer window mechanism. The double-layer window is a sliding window composed of multiple basic windows. The concept drift is judged by periodically detecting the change in the distribution of stream data in the window. But the periodic detection of the algorithm consumes a large amount of time. Yim et al. [18] proposed a variable sliding window. The algorithm initializes the window by storm parallelization window scheme. According to the Poisson distribution and holding inequality, the information distribution implicit in the window is detected. If the expected value $\mu_1 - \mu_2$ is less than the threshold, it means that there may be a conceptual drift. Then the window is appropriately reduced. Otherwise, the window is appropriately enlarged. The algorithm improves the throughput and processing speed by referring to the parallelization mechanism. However, the algorithm does not take into account different types of conceptual drift. At the same time, when the window is reduced too small, it may result in insufficient sample data, greater noise impact, and more errors.

Aiming at the shortcomings of the above concept drift detection algorithm, this paper proposes a decision tree classification algorithm based on data attribute features. According to the attribute characteristics of the data, it observes the feature difference between the front and end of the data to detect concept drift. It uses the window mechanism and expands or shrinks the window size according to the degree of conceptual drift. The window size is adjusted twice according to the increase and decrease of the data volume, so that the data can be processed effectively and timely at different flow rates. The classifier is periodically updated to meet the changes of the data concept. According to the different types of concept drift, corresponding measures are taken to deal with the conceptual drift effectively.

2.2. Related Concepts. The data stream $D = \{d_1, d_2, d_3, \ldots, d_n\}$ is a sequence of data that is generated over time. For each piece of data $d_i(in)$, there is a corresponding attribute set and a corresponding potential class label. By training and learning the data of known class labels, an effective classification model is created, and the real class labels are derived.

2.2.1. Hoeffding Inequality. If the sampler has a value range of R, after obtaining n samples, the average value of the sample is $\overline{r}$; then the true average value of r is at least $\overline{r} - \varepsilon$ with a probability of $1 - \delta$, where

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}. \tag{1}$$

The minimum value of n in the formula is the minimum real number required for the current node to split.

2.2.2. Degree of Difference. Assume a matrix M of $m \times n$ dimensions; here $a_{ij} \in M\,(0 \le i < m, 0 \le j < n)$. Two submatrices M_k, M_l are shown as

$$M_k = \begin{bmatrix} a_{00} & \cdots & a_{0,n-1} \\ \vdots & \ddots & \vdots \\ a_{k0} & \cdots & a_{k,n-1} \end{bmatrix},$$
$$M_l = \begin{bmatrix} a_{l0} & \cdots & a_{l,n-1} \\ \vdots & \ddots & \vdots \\ a_{m0} & \cdots & a_{m,n-1} \end{bmatrix}, \tag{2}$$
$$0 \le k \le l < m - 1,$$

where $\sigma_k = (\sigma_{k0}, \sigma_{k1}, \ldots, \sigma_{k,n-1})$ is the statistical vector of M_k and $\sigma_l = (\sigma_{l0}, \sigma_{l1}, \ldots, \sigma_{l,n-1})$ is the statistical vector of M_l. Then define the degree of difference δ of between matrices M_k and M_l as

$$\delta = \frac{1}{n} \sum_{i}^{n-1} (\sigma_{ki} - \sigma_{li}). \tag{3}$$

The degree of difference δ is used to measure whether the data has a distribution change or not. If δ is greater than a predetermined threshold, it can be considered that the elements of matrix M have undergone a data distribution change.

3. Classification Algorithm Based on Decision Tree for Stream Data

The difficulty of the stream data classification problem lies in how to solve the problem of concept drift and incomplete labeling. If there are no such problems, the traditional classification algorithm can also be valid for stream data classification. To some extent, the difficulties caused by the incomplete labeling problem are also caused by concept drift. When concept drift does not occur, that is, the data distribution is very stable, in this case, even if the ratio of the training data amount to the test data amount is very low, the conventional classification algorithm can perform well.

A decision tree is a basic classification method. Because the decision tree has a relatively small amount of calculation, it is easy to be converted into classification rules with high accuracy, easy to understand, and suitable for high-dimensional data. Therefore, decision trees are often used to do streaming data and classification models.

In this paper, the algorithm uses a decision tree as the classification model to classify the data in the sliding window. Through the degree of concept drift and the prediction of the data volume, the sliding window size is dynamically adjusted. When the classification accuracy is reduced, the classification model is updated. The algorithm framework is shown in Figure 1. It mainly consists of three modules, which are the classifier building module, sliding window

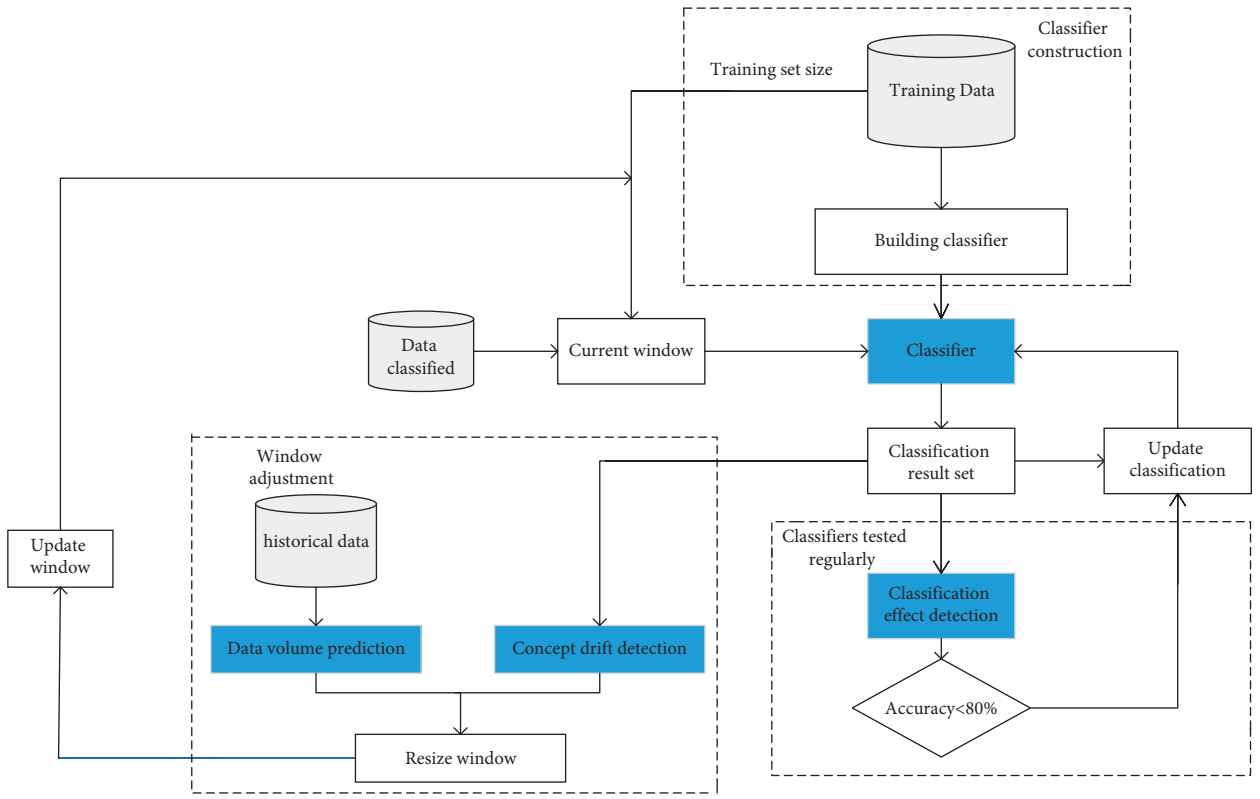

Figure 1: Algorithm architecture.

adjustment module, and classifier periodic detection module. The whole process is described as follows:

(1) In the classifier building module, an integrated classifier composed of multiple decision trees is established through training data with class labels.

(2) Using the current classifier to classify the incoming stream data, the data to be classified is formed into an initial window according to the size of the sliding window. The current classification model is used to classify the data in the window to obtain a classification result set, and the old classification model is updated by using the classification result set. Regularly use the data with the class label to monitor the classification performance, and observe whether the accuracy rate is higher than the preset threshold or not. If the proportion of the classification error is too large, the current classifier needs to be updated.

(3) In the window adjustment module, use the historical data to predict the data volume of the stream data at first. Then perform the concept drift detection on the classification result set obtained in the second step and adjust the size of the window according to the result of the prediction and the detection. At last, update the current window size.

3.1. Dynamic Data Stream Classification Algorithm Based on VFDT Model. In this paper, an integrated classification algorithm DVFDTc, which is based on a decision tree, is designed. The algorithm mainly completes the construction of the classifier. In the process of classifying test data, randomization, voting, and other mechanisms are adopted to avoid the error caused by the contingency, and then the accurate classification of the stream data is realized. When the classified data is enough for sliding window size, construct a new classifier. Then, the base classification model is updated.

The algorithm maintains two data windows Win_1 and Win_c. In the beginning, Win_1 stores the original training data while Win_c stores the data classified in the most recent period. In the subsequent running process, they store the classified data of the nearest neighboring period respectively. After putting the training data to Win_1, the integrated classification algorithm trains the data in Win_1 to obtain the initial classification model M_1. Then, reclassify the data in Win_1 by M_1 *and put the* result set into Win_c. Reconstruct the current classification model M_c with Win_c. So M_1 and M_c form the initial integrated classification model.

For a new incoming data d_i, data d_i is classified by M_1 and M_c, respectively. If these two classification results are consistent, label d_i with the result and put it to temporary data container Win_{amp}. If they are inconsistent, the data of Win_1 and Win_c are mixed and sampled according to a certain ratio. K mixed data sets $H_0, H_1, \ldots, H_{k-2}, H_{k-1}$ are used to construct K new classification models $M_0, M_1, \ldots, M_{k-2}, M_{k-1}$. Finally, the K classification models are used to classify the data d_i. The final result is generated by voting, while the most supported class label is the final result. The marked data is added to the temporary data container

Win_{amp}. When the size of the data container Win_{tmp} reaches the sliding window size, update Win_1 by Win_c and update Win_c by Win_{amp}. At the same time, update the corresponding classification model.

The core idea of Algorithm 1 is as follows:

Concept drift has a great impact on classification performance. This paper also designs the evaluation operation of the current classification model. It uses the classification accuracy rate as an evaluation indicator to determine whether the classification model adapts to the current data environment or not. In this paper, the three times iterative processes are used as the cycle, and the labeled data is used as the verification set. Through classifying the verification set to evaluate the accuracy, if the accuracy is less than the preset value, the current classification model will be removed and the correctly classified data is reconstructed according to the initial base classifier construction process, so that the classification model can adapt to the changing data environment.

4. Concept Drift Detection Algorithm

Due to changes in data distribution, the current classifier cannot adapt to the change of data concept. This leads to a drop in inaccuracy. This paper proposes a concept drift detection algorithm based on data attribute characteristics. The characteristics of each attribute of the data are counted, and observe the difference of data distribution before and after and then determine whether the concept drift occurs or not.

4.1. Concept Drift Based on Attribute Features. In this paper, the attribute characteristics of the data are taken as the research object, and the distribution state of the current data is judged by the attribute characteristics. Take the first $1/n (n \in N_+)$ at the beginning and the last $1/n$ at the end of the sliding window as two subdatasets. Assume they are Win_0 and Win_1. Each data set is looked at as a set of high-dimensional vectors. As shown in (5) and (6), calculate the square mean of different values of the attributes in each data set as the distribution state of the data.

For an $m \times n$ dimensional matrix M, the element $a_{ij} \in M\,(0 \le i < m, 0 \le j < n)$, and the eigenvalue σ of the statistical matrix M is defined as

$$\sigma = \left(\sigma_0, \sigma_1, \ldots, \sigma_j, \ldots, \sigma_{n-1}\right), \tag{4}$$

$$\sigma_j = \frac{1}{m} \sum_{i=0}^{m-1} a_{ij} \cdot a_{ij}. \tag{5}$$

Then the difference between Win_0 and Win_1 is determined according to

$$\frac{\left|\sigma_{win1} - \sigma_{win0}\right|^2}{n} > \varepsilon_{cut}. \tag{6}$$

If their difference is greater than the preset threshold ε_{cut}, then the distribution state of the data is considered changes. That is when the concept drift occurs.

4.2. Selection of Threshold. The size of the threshold plays a key role in the judgment of the concept drift. If the threshold value is too small, the concept drift may occur frequently, while the classification model and sliding window are updated frequently. This will result in a decrease in classification efficiency and accuracy. If the value is too large, then there may be little or no conceptual drift. It also reduces the accuracy of the classification. So the selection of the threshold size is very important.

To avoid the above situations and to ensure that the number of occurrences of concept drift is within a reasonable range, this paper proposes a dynamic method to adjust the threshold value. Determine whether to adjust the size of the threshold according to the frequency of concept drift.

The threshold is set as the difference between the head and tail data of the initial sliding window at first. During the classification process, observe the rate of occurrences of the concept drift. If the rate is more than $1/f$, the concept drift is considered to be too frequent, and then the threshold will be increased by $1/p$. If the concept drift occurs less than $1/q$ or hardly occurs, the threshold will be reduced by $1/p$. Repeat the above process until the concept drift ratio is within a reasonable range.

4.3. Rationality Analysis of the Concept Drift Detection Method. The detection process of concept drift is mainly judged according to the change of data distribution characteristics. The changes of data distribution characteristics usually have two types which are rise and fall from the perspective of the changing trend of eigenvalues; but from the perspective of the magnitude of change, there are mutations and gradients.

For the case of sudden changes of data distribution, analyze the head and the tail subwindows Win_0 and Win_1 of the current sliding window. The result is shown as Figure 2.

It can be seen from Figure 2 that the mutation of the data distribution can be effectively detected by calculating the eigenvalues σ_0 and σ_1 of the two windows.

For the case that the data distribution is gradually changed, if the gradation trend is monotonous, it can be well captured according to (6). Conversely, if the trend is not monotonous and erratic, then the error caused by the gradient of the data distribution can be corrected by the randomness of the sample in the integrated classification algorithm. Assume that the data distribution changes irregularly as shown in Figure 3.

If the data distribution difference between the head and tail does not reach the threshold, the data distribution change in the current window can be ignored. Observe whether the class label for the test data d is the same for the two classification models M_0 and M_c. If they are the same, the data feature changes can be ignored. Otherwise, the data of these two subwindows are proved to have a large feature change. Then, random sampling is performed in two data windows in proportion. By using the reservoir sampling method, the data extraction probability is equal for both sliding windows. The data extracted from the two data

(1) $Win_1 = S_0$/* Training set to build initial classification model */
(2) M_1 = BuildVFDT(Win_1)
(3) ClassValue = M_1.Classify(Win_1)
(4) $Win_c = S_0$, ClassValue;
(5) M_c = BuildVFDT(Win_c)
(6) For $d_i = 1, \ldots, n$ do
(7) $ClassValue_{oi} = M_1$.Classify(d_i)
(8) $ClassValue_{ci} = M_c$.Classify(d_i)
(9) If($ClassValue_{oi} == ClassValue_{ci}$)
(10) $WriteToLocal(ClassValue_{oi})$
(11) Win_{tmp}.add($ClassValue_{oi}$)
(12) Else
(13) For $j = 1, \ldots, k$ do
(14) $H_j = Mix\_Sample(Win_1, Win_c)$
(15) $M_j = buildVFDT(H_j)$
(16) For $j = 1, \ldots, k$ do
(17) $R_{hj} = M_j$.classify(d_i)
(18) $tmpR = Most\_Appears(R_h)$
(19) $WriteToLocal(tmpR)$
(20) If ($i\%\ Win_c$.size == 0)
(21) $Win_1 = Win_c$
(22) $Win_2 = Win_{\text{tmp}}$
(23) $M_1 = M_c$
(24) $M_2 = BuildVFDT(Win_c)$

Algorithm 1: Classification algorithm DVFDTc.

windows is combined into a new data window. The data of the new data window is trained to obtain k classification models $M_0, M_1, \ldots, M_{k-1}$. The voting mechanism is used to determine the class label of the test data d. By reasonably setting the sampling ratio of the two data windows, the classification accuracy of the hybrid type can be effectively improved.

Through the above analysis, the proposed concept drift detection algorithm can effectively deal with the concept drift of mutation and gradual change.

5. Sliding Window Dynamic Adjustment Algorithm

In the sliding window-based classification algorithm, the size of the sliding window has a great impact on the speed and accuracy of data processing. Therefore, the size of the sliding window should be constantly changing as the data changes, so that the current data can be processed accurately in real time. Two main factors are affecting the size of the sliding window. First, the flow speed of the data stream. The sliding window size should be adjusted according to the change of the flow speed; secondly, the concept drift. When the concept drift occurs, the window should be properly reduced, and the data distribution in the window should be guaranteed to be consistent as much as possible.

Therefore, this paper detects the concept drift and predicts the speed of data and expands or shrinks the size of the data window based on the results of the detection and prediction.

5.1. Data Traffic Prediction. According to the flow characteristics of the data stream, the number of data changes in a day may be regular. For example, a web page may be heavily accessed during a certain period while it may be rarely visited during another period. When a large amount of data is gathered at a certain moment, it may cause the data to be lost without processing if the sliding window still maintains the original size. Conversely, if the flow speed is slow, the window will spend a lot of time waiting for the aggregation of data. It may cause a waste of resources and time. This paper predicts the possible data flow speed before the next window arrives to adjust the sliding window size.

This paper implements time serialization and studies the law of increase and decrease of data volume per day from historical data. Excavate the amount of data for the preset period within the time of day. If the amount of data is greater than the average amount of data in the preset period, then the current data volume is considered to be in a rising or high tide stage, and the sliding window will be expanded at this time; otherwise, the data is considered to be in a falling or low tide stage, and the window size will be reduced.

5.2. Sliding Window Size Adjustment. In the window adjustment in Algorithm 2, according to the concept drift and the amount of data, the next sliding window size is processed as follows:

Detect the concept drift of the current window data and determine whether the data in the window has a conceptual drift. If concept drift occurs, obtain a window reduction degree value L according to the current window that is reduced by its size to get the window size width.

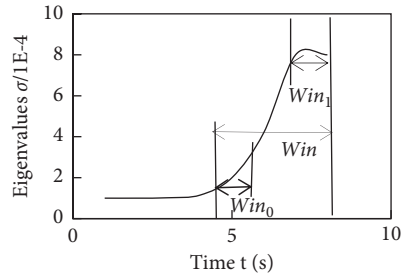

Figure 2: Stream data characteristic mutation trend analysis diagram.

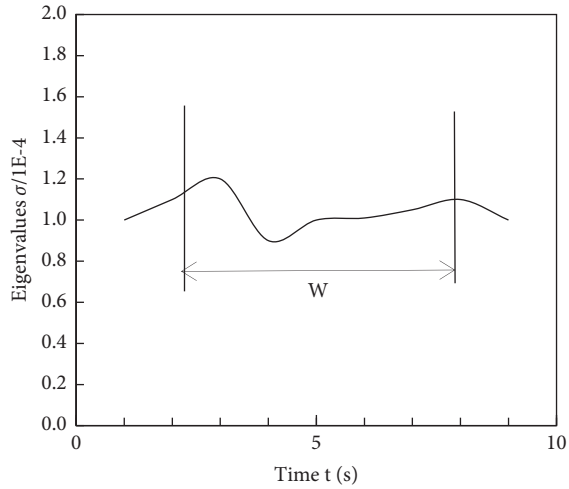

Figure 3: Stream data feature gradient trend analysis diagram.

$$L = n\frac{(a_{k0} - a_{l0}) + \cdots + (a_{k,n-1} - a_{l,n-1})}{\sqrt{\left((a_{k0} - a_{l0})^2 + \cdots + (a_{k,n-1} - a_{l,n-1})^2\right)/n}}. \quad (7)$$

According to the increase or decrease of the data amount of the preset period, the width is adjusted again to obtain the size of the next sliding window. Details are as follows:

(1) If the amount of predicted data is greater than the threshold, the window size will be increased predicted data by $1/w$ based on the concept drift to reduce the window without changing the trend of the data window becoming smaller.

(2) If it is less than the preset amount of data, the window is reduced according to the amount of predicted data by $1/s$ based on the concept drift to reduce the window (not less than the minimum window length of 800).

If no concept drift occurs, the window is linearly expanded:

(1) If the amount of predicted data is greater than the preset amount of data, the window is increased according to the amount of predicted data by $1/w$ based on the expansion of the concept drift on the window.

(2) If it is smaller than the preset amount of data, the $1/s$ of the predicted data amount is reduced based on the concept drift to reduce the window without changing the tendency of the data window to become large.

According to the above situation in the classification process, the corresponding processing is performed. This process is repeated over and over in classifier iterations and updates.

6. Experiment and Analysis

The experiment is implemented based on Windows 10 operating system, Eclipse, and Weka development environment. The data used in the experiment is the synthetic data formed on the Weka development tool. The synthetic data set has 50,000 pieces of data, each of which has 20 attribute dimensions and 2 class labels. About experimental analysis, this paper introduces the algorithm in terms of accuracy and efficiency.

6.1. DVFDTC Algorithm Classification Accuracy Analysis. The verification method of the stream data classification algorithm is different from the verification method of the traditional static classification algorithm. In this experiment,

Initialize: length = 0;/∗length of slide window ∗/
(1) Win_1 = Data from training set, Win_2 = Data from a classifier created by a training set
(2) While(d_i ! = $Null$)
(3) $Width_{\text{time}}$ = Predict the amount of data for a given time period
(4) If(Concept Drift)
(5) length = $Wi\ dt\ h_{\text{time}} * L$
(6) Else
(7) Length = $Wi\ dt\ h_{\text{time}} + Wi\ dt\ h_{\text{time}} * (1/4)$
(8) If($Width_{\text{time}} > \tau$)
(9) length+ = $Wi\ dt\ h_{\text{time}} * 1/w$
(10) Else
(11) length− = $Wi\ dt\ h_{\text{time}} * 1/s$

Algorithm 2: Sliding window dynamic adjustment algorithm.

the ten-fold cross-validation algorithm is improved to verify the stream data classification algorithm more effectively.

First, the data set is declassified, and only the class label of the first 1/10 of the data set is reserved for training. The instance of the window in each iteration process is regarded as the data block $\{B_0, B_2, \ldots, B_{10}\}$. In this experiment, the VFDT algorithm and DVFDTc algorithm are compared in algorithm classification accuracy, and the classification accuracy of the two algorithms is shown in Table 1.

It can be seen from Table 1 that the classification accuracy of DVFDTc is higher than that of the VFDT algorithm. However, the accuracy of DVFDTc is lower than that of VFDT at the beginning, because the initial amount of data is insufficient; in the concept drift detection process, the selection of the threshold is inexperienced, and the accuracy is low. However, as the classifier continues to iteratively update, the accuracy rate has generally kept rising.

DVFDTc obtained experimental results in the process of traversing 10 data blocks, the average accuracy increased by 2.95%, and the accuracy rate is increased by a maximum of 7.83%.

6.2. DVFDTC Algorithm Execution Efficiency Analysis. The performance of the DVFDTc algorithm is mainly analyzed from the perspective of the execution time of the algorithm. The experiment compares DVFDTc and VFDT. The results are shown in Figure 4.

From the above figure, the execution times of the DVFDTc algorithm and the VFDT algorithm are both longer than the execution time of the VFDT algorithm, and the DVFDTc algorithm has the longest execution time. The reason for this phenomenon is that the DVFDTc algorithm and the VFDT algorithm have iterations and updates to the classifier in the process of classification. A large number of intermediate results are written locally during the execution of the program, which greatly increases the read and writes of the disk IO. If the intermediate results of the algorithm are stored in memory, the running time of the algorithm can be greatly reduced. Another reason for the long execution time of the DVFDTc algorithm is that the algorithm constructs multiple classifiers and also has timing detection of the classifiers. In fact, in the case where the classification accuracy requirement is not very high, the number of voting classifiers can be reduced to effectively reduce the running time of the algorithm.

6.3. SWDA Performance Analysis. The performance of the SWDA algorithm is analyzed from two angles. One is to analyze whether the algorithm helps improve the accuracy of the DVFDTc algorithm; the other is to analyze the influence of the SWDA algorithm on the execution time of the DVFDTc algorithm. The experiment is carried out by comparing the DVFDT algorithm with the SWDA algorithm and the DVFDT algorithm without the SWDA algorithm. Count their classification accuracy and execution time.

The accuracy statistics of DVFDTc with the SWDA algorithm are shown in Table 2.

It can be seen from the lifting degree in Table 2 that the DVFDTc algorithm with the SWDA algorithm has a certain improvement inaccuracy, which proves that the results of concept drift detection and data volume prediction are reasonable for the sliding window size adjustment strategy. However, the reason why the efficiency improvement of the DVFDTc algorithm is small is that the SWDA algorithm and the DVFDTc algorithm have overlapping effects on the accuracy of the stream data classification.

The execution time statistics of DVFDTc with integrated sliding window dynamic adjustment algorithm are shown in Table 3.

In the last column of Table 3, the execution time of the DVFDTc algorithm and DVFDTc with the SWDA algorithm is similar to the execution time of the SWDA algorithm. The execution time of the SWDA algorithm is much lower than the execution time of the DVFDTc algorithm.

6.4. Summary of the Experiment. Through the verification of the above experiments, it can be seen that the classification algorithm based on a decision tree for stream data has good performance. The average accuracy is 3.66% higher than that of the VFDT algorithm. However, due to the addition of classifier iteration, timing detection, and the concept drift detection process, the execution time is lower than VFDT.

Table 1: Comparison of VFDT and DVFDTc algorithm accuracy statistics.

Data block	VFDT (%)	DVFDTc (%)	Lifting degree
B_0	74.03	70.63	−3.40%
B_2	76.54	73.85	−2.69%
B_4	78.76	80.43	1.67%
B_6	80.53	84.65	4.12%
B_8	80.89	85.79	4.90%
B_{10}	80.07	87.19	7.12%

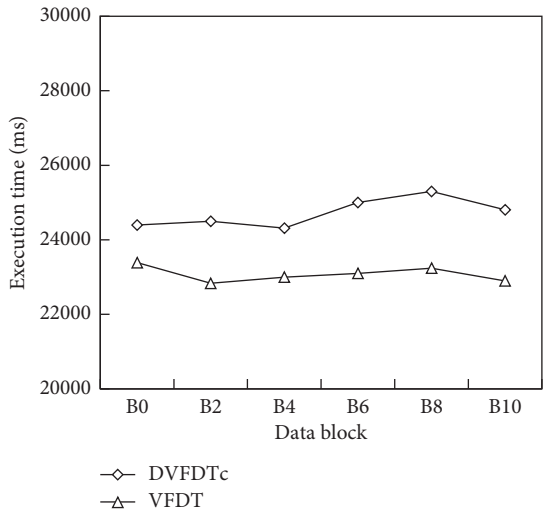

Figure 4: Comparison of execution time.

Table 2: The accuracy rate statistics table of DVFDTc with SWDA algorithm.

Data block	DVFDTc (%)	DVFDTc with SWDA algorithm (%)	Lifting degree (%)
B_0	70.63	70.63	0
B_2	73.85	77.38	3.53
B_4	80.43	81.04	0.61
B_6	84.65	85.13	0.48
B_8	85.79	86.27	0.48
B_{10}	87.19	88.10	0.91

Table 3: The execution time statistics table of DVFDTc algorithm with SWDA algorithm.

Data block	DVFDTc/ms	DVFDTc with SWDA algorithm/ms	Difference/ms
B_0	24400	24450	50
B_2	24500	24575	75
B_4	24315	25069	754
B_6	25000	26035	1035
B_8	25300	25467	167
B_{10}	24805	25001	196

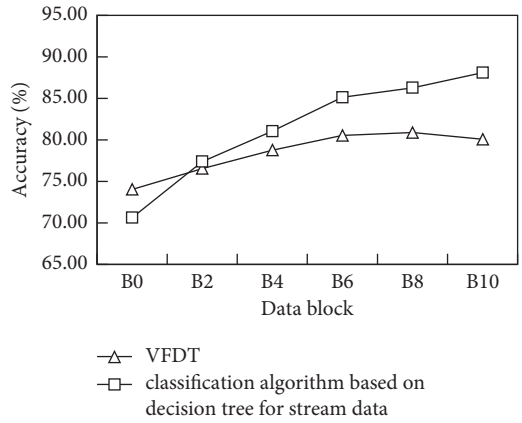

Figure 5: Comparison of the accuracy of the flow data classification algorithm based on decision tree and VFDT algorithm.

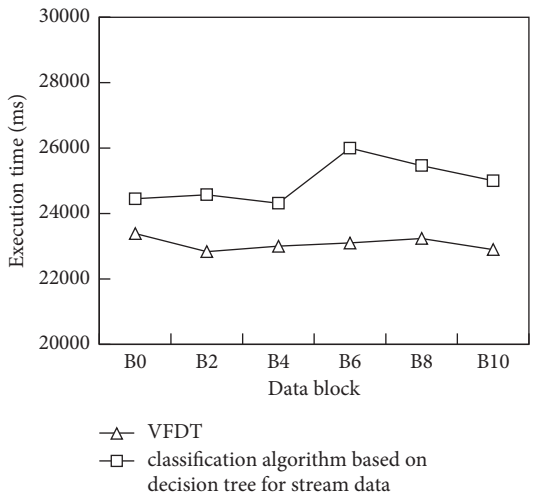

Figure 6: Comparison of the execution time of flow data classification algorithm based on decision tree and VFDT algorithm.

The accuracy ratio of the flow data classification algorithm based on the decision tree and VFDT algorithm is shown in Figure 5.

The execution time comparison of DVFDTc and VFDT algorithm is shown in Figure 6.

7. Conclusion

Aiming at the problem of incompletely labeled stream data classification, this paper designs and optimizes the stream data classification algorithm based on a decision tree. The algorithm solves the problem of concept drift to a certain extent and has a great improvement in accuracy.

As the iterative update of the classifier increases the read and write of IO, the algorithm is still lacking in execution efficiency. In the next stage, we will study how to implement a decision tree-based stream data classification algorithm on a distributed platform to improve the execution efficiency of the algorithm.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant nos. 61402090 and 61806218.

References

[1] X. Hu, P. Li, and Y. Zhang, *Data Stream Classification*, pp. 205-206, Tsinghua University Press, Beijing, China, 2015.

[2] Z. Yu, *Research on Related Issues of Massive Data Mining*, M.S. thesis, 2015.
[3] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the ACM SIGKDD*, pp. 71–80, Boston, MA, USA, August 2000.
[4] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, 1962.
[5] J. Gama, R. Rocha, and P. Medas, "Accurate decision trees for mining high-speed data streams," in *Proceedings of the ACM SIGKDD*, pp. 523–528, New York, NY, USA, August 2003.
[6] M. Han, Z. Wang, and J. Ding, "A frequent pattern decision tree for processing variable data streams," *Chinese Journal of Computers*, vol. 39, no. 8, pp. 1514–1554, 2016.
[7] S. Y. Ooi, S. C. Tan, and W. P. Cheah, "Temporal Sleuth Machine with decision tree for temporal classification," *Soft Computing*, vol. 22, no. 9, 2017.
[8] W. Street and Y. Kim, "A streaming ensemble algorithm(sea) for large-scale classification," in *Proceedings of the ACM SIGKDD*, pp. 377–382, San Francisco, CA, USA, July 2001.
[9] H. Wang, W. Fan, E. Yu, and J. Han, "Mining concept-drifting data stream using ensemble classifiers," in *Proceedings of the ACM SIGKDD*, pp. 226–235, New York, NY, USA, July 2003.
[10] W. Pan, G. Cheng, and X. Guo, "Adaptive network flow concept drift classification method based on information entropy," *Journal of Computers*, vol. 7, 2017.
[11] O. A. Mahdi, E. Pardede, and J. Cao, "Combination of information entropy and ensemble classification for detecting concept drift in a data stream," in *Proceedings of the ACSW*, pp. 1–5, Brisbane, Queensland, Australia, January 2018.
[12] P. Li, L. He, X. Hu, Y. Zhang, L. Li, and X. Wu, "Concept-based short text stream classification with topic drifting detection," in *Proceedings of the IEEE ICDM*, Barcelona, Spain, December 2016.
[13] P. Zhang, X. Zhu, J. Tan, and L. Guo, "Classifier and cluster ensembles for mining concept drifting data streams," in *Proceedings of the IEEE ICDM*, Sydney, Australia, December 2010.
[14] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing Data streams," in *Proceedings of the ACM SIGKDD*, pp. 97–106, NewYork, NY, USA, July 2001.
[15] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive window," in *Proceedings of the ICDM*, pp. 443–448, Minneapolis, MN, USA, April 2007.
[16] A. Bifet and R. Gavaldá, "Adaptive learning from evolving data Streams," in *Proceedings of the ISIDA*, pp. 249–260, Lyon, France, August 2009.
[17] Q. Zhu, Y. Zhang, and X. Hu, "A two-dimensional window-based concept drift data stream classification algorithm," *Journal of Automation*, vol. 37, no. 9, 2011.
[18] J. Yim, Y. Zhang, X. Lang, D. Zhang, and R. Wang, "Parallel of decision tree classification algorithm for stream data," *Journal of Computer Research and Development*, vol. 54, no. 9, pp. 1945–1957, 2017.