Knowledge and Information Systems (2023) 65:1105–1128
https://doi.org/10.1007/s10115-022-01791-5

REGULAR PAPER

Dynamic ensemble selection classification algorithm based on window over imbalanced drift data stream

Meng Han[1] · Xilong Zhang[1] · Zhiqiang Chen[1] · Hongxin Wu[1] · Muhang Li[1]

Received: 1 March 2022 / Revised: 31 October 2022 / Accepted: 7 November 2022 /
Published online: 27 November 2022

Abstract
Data stream classification is an important research direction in the field of data mining, but in many practical applications, it is impossible to collect the complete training set at one time, and the data may be in an imbalanced state and interspersed with concept drift, which will greatly affect the classification performance. To this end, an online dynamic ensemble selection classification algorithm based on window over imbalanced drift data stream (DESW-ID) is proposed. The algorithm employs various balancing measures, first resampling the data stream using Poisson distribution, and if it is in a highly imbalanced state then secondary sampling is performed using a window storing a minority class instances to achieve the current balanced state of the data. To improve the processing efficiency of the algorithm, a classifier selection ensemble is proposed to dynamically adjust the number of classifiers, and the algorithm runs with an ADWIN detector to detect the presence of concept drift. The experimental results show that the proposed algorithm ranks first on average in all five classification performance metrics compared to the state-of-the-art methods. Therefore, the proposed algorithm has better classification performance for imbalanced data streams with concept drift and also improves the operation efficiency of the algorithm.

Keywords Data stream · Imbalance data · Concept drift · Window sampling · Ensemble classification

1 Introduction

1.1 Motivation

In the traditional data mining domain, the entire data can be loaded in memory and the classification model can access the instances multiple times. But in a dynamic data stream environment, data items that appear in a temporal order cannot be stored down at once [1]. At the same time, during learning from streaming data, if the target concept of the data changes over time, concept drift is said to have occurred [2]. This phenomenon occurs in

✉ Meng Han
2003051@nmu.edu.cn

1 School of Computer Science and Engineering, North Minzu University, Yinchuan, China

real machine learning applications, for example, spammers constantly improve the quality of their spam messages posted on Twitter to avoid being intercepted by spam detection systems, and therefore, the characteristics and concepts of Twitter spam change frequently [3]. The appearance of concept drift in the data stream usually requires a drift detector to check the presence of concept drift then before doing further classification process, and the common ones are drift detection method DDM [4], single window method EDDM [5], and adaptive double window drift detection algorithm ADWIN [6].

In addition to the above problems of concept drift, class imbalance is more prevalent in real-data stream environments. Class imbalance exists in many real-world applications, such as network intrusion detection and credit card transactions [7]. Traditional classifiers are more inclined to majority class instances and poor performance of minority class instances. However, minority class occurrence is of more concern for researchers. With many proposed solutions to imbalanced classification problems, three categories can be classified based on the proposed processing methods. The first is the resampling technique, i.e., the use of data preprocessing, which attempts to equalize the number of samples from both classes by increasing the number of minority class instances or decreasing the number of majority class instances. In this case, the minority class instances can be simply oversampled or the majority instances can be undersampled. For example, the oversampling algorithm SMOTE [8] uses the difference to create synthetic minority instances instead of directly copying the instances to avoid overfitting, but this method may lead to changes in the characteristics of the minority class thus introducing new data concept. At the same time, Poisson distribution technology is one of the most commonly used methods for data preprocessing. Du et al. [9] achieved oversampling of the minority class and undersampling of the majority class by changing the λ parameter in the Poisson distribution of online Bagging. The second technology uses a cost-sensitive approach at the algorithm level, which considers a different loss function that assigns a higher cost to the misclassification of minority class instances. For example, Sun et al. [10] combined cost-sensitive and ensemble algorithms Boosting with AdaC1, AdaC2 and AdaC3. The third one is a hybrid approach which combines data preprocessing methods and classification methods. The most widely used is the combination of resampling techniques and ensemble learning. For example, the following classical algorithms SMOTEBoost [8], (GRE, Gradual Resampling Ensemble) [2], (SRE, Selected based Sampling Ensemble), etc.

1.2 Contribution

The above algorithm has the following shortcomings: (1) In the case of highly imbalanced data, the use of a simple Poisson distribution-based resampling technique may not be able to quickly balance the current data processing. (2) Using a fixed number of base classifier quantity may not be able to further reflect the previous diversity of the classifiers, thereby improving training efficiency. (3) The existing resampling algorithm SMOTE can deal with the problem of class imbalance, but in the environment of concept drift, the algorithm generates new minority samples through the difference, which can easily introduce new concepts and hinder classification.

In response to the analysis of the above literature, a new ensemble classification algorithm is proposed in this paper with the following main contributions.

(1) A new window sampling method based on Poisson distribution is studied and proposed to solve the class imbalance problem in the data stream. Three different Poisson distribution sampling settings are used in the window during the training process, and the method improves the true positive rate of a minority classes to some extent.

(2) A classification algorithm with dynamic ensemble selection based on window over imbalanced drift data stream (DESW-ID) is studied and proposed. In order to select the optimal number of classifier combinations, we use classification errors to sort the trained classifiers, and then use the reverse search algorithm to find the optimal number of classifiers. This dynamic selection strategy is proved to improve the efficiency of the algorithm.
(3) A new weighting equation is investigated and proposed to further consider the classification performance of both minority and majority classes by introducing the G-mean metric.

2 Related work

2.1 Imbalanced data stream classification

Boosting and Bagging algorithms are the most used ensemble frameworks in data stream classification, but this type of algorithm is basically designed for batch learning. In order to adapt to online learning more, Oza et al. [11] proposed Online Bagging and Online Boosting, the author uses Poisson distribution sampling to convert from batch processing to online learning mode. Since then, everyone has proposed various variations of it. Barros et al. [12] proposed BOLE's ensemble learning algorithm based on the heuristic modification. The algorithm is based on the modification of adaptive diversity online promotion (ADOB). Both are Online Boosting. In order to better adapt to imbalanced data, many researchers continue to improve the algorithm. For example, in the literature [13], batch mode algorithms such as AdaBoost, RUSBoost, SMOTEBoost, and EUSBoost are changed to online learning versions by using Poisson distribution adjust the weight of the samples in the data stream, and the samples in the data stream will be resampled. And Wang et al. [14] improved oversampling-based online Bagging (OOB) and undersampling-based online Bagging (UOB) based on the online Bagging ensemble algorithm.

2.2 Classifier ensemble selection

As scholars' research on ensemble classification deepens, the ensemble structure of classifiers gradually changes from a fixed number of classifiers to a dynamic classifier selection strategy. It has been experimentally verified that the dynamic classifier selection strategy can increase the diversity among classifiers and also increase the classification performance of classifiers. Dynamic ensemble selection is the selection of the best classifier on each test set, and the region where the instances of classifiers are usually evaluated is called the dynamic selection data set (DSEL) [15]. Many classical DES algorithms use KNN-based methods to select the desired samples from the DESL.

Based on the criterion of single classifier performance evaluation, KNORA-Eliminate (KNORAE) algorithm is proposed [16], which selects only all instances of the base classifier capability region that can be correctly classified. And the KNORA-Union (KNORAU) [16] algorithm makes a decision based on weighted voting, where all classifiers that can be correctly classified select one instance in the capability region. Where the DES-P [17] method uses a strategy of selecting classifiers with better performance than random classification DES-KNN [18] is to select some classifiers with the best classification accuracy. Then, some classifiers with the best diversity are selected as ensemble classifiers. Meta-DES [19] considers a variety of evaluation criteria, such as posterior probability and local accuracy. During the generalization phase, the meta-features are extracted from the query instance and passed down as input to the meta-classifier which estimates whether a base classifier is competent enough to be added to the ensemble. Recently proposed DES-MI [20] multi-class imbalance algorithm. In DES-MI, the classification ability of a classifier is evaluated based on weighted accuracy. The classifier capability is determined by weighting and summarizing the local accuracy of the instances in the capability region and assigning higher weights to a minority class instances. Therefore, the classifier performs well on a small group of instances that can be easily selected.

2.3 Concept drift detection method based on ADWIN

One of the biggest challenges in non-stationary data stream is the existence of concept drift. The occurrence of the current data distribution changes over time is called concept drift. Zhang et al. [21] described in a review of data stream ensemble classification that concept drift occurs when the joint probability of two-time points to $t1$ and $t2$ changes, denoted as $p_{to}(X, y_1) \neq p_{t1}(X, y_2)$. Four types are classified according to the rate of change in the data distribution. That is recurring drift, sudden drift, incremental drift, and gradual drift, which are shown in Fig. 1. Recurring concept drift is the situation where the data concept starts to drift repeatedly as soon as it appears in the cycle. Sudden drift refers to the rapid generation of new concepts as the cluster structure comes to change dramatically in a short period of time. Incremental drift is a slow evolution over time, and the process may be similar to gradual drift. Gradual drift may occur when data concepts change gradually over time and exhibit a low frequency and low magnitude of change.

With the in-depth exploration of concept drift by researchers, the current methods to deal with concept drift are mainly divided into two categories, namely passive adaptation and active detection. The most commonly used passive adaptation is the use of a dynamic weighted update method for the members of the ensemble classifier to delete poorly trained members. Active detection technology uses statistical methods and window methods to detect the existence of concept drift and then executes the current training model. This article mainly uses the ADWIN window proposed by Bifet [6] to detect the existence of concept drift. The window only saves the most recently seen samples. It reserves a variable-length window for the most recently seen. It can automatically detect and adjust its window to adapt to the current rate of change. ADWIN uses a threshold called delta in Eq. (1) to automatically

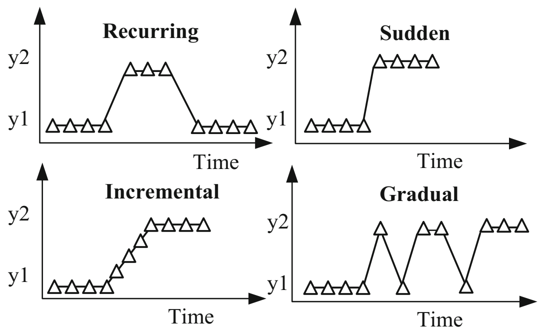

Fig. 1 Concept drift type

configure errors with two levels, called warning and change levels.

$$\text{LevelError} = \log\left(\frac{2\times\log n}{\text{delta}}\right) \tag{1}$$

where delta $\times$ 10 identifies the warning level, while the change level is identified using delta. Since delta appears in the denominator, using delta $\times$ 10 will produce a lower value than using delta. So the warning level will occur before the change, and n is the width of the window at that time.

3 Proposed DESW-ID algorithm

3.1 The training process of the DESW-ID algorithm

After a careful analysis of the literature in the above sections, most of the imbalanced data classification problems perform poorly, while many do not take into account the existence of concept drift in imbalanced data stream in a timely manner. The proposed solution should be innovative in terms of data processing itself and the structure of the algorithm to further improve the recognition rate of a minority classes. Therefore, this paper proposes the DESW-ID ensemble classification algorithm to solve the problem of both imbalance and concept drift in data streams. Figure 2 shows the overview of the proposed algorithm. We divide the training process of the algorithm into three phases. Firstly, the data stream continuously enters the data stream window Win_{Ins}, while another window Win_{Pos} stores the minority class instances. If the instances in the data stream window are in the imbalanced state, a certain percentage of the instances stored in the minority class window is selected to balance the data in the current state. In the training phase of the base classifier, if the ADWIN concept drift detector does not detect any concept drift in the current data, the data stream window Win_{Ins} Ins is enlarged to continue the training. Otherwise, the current training window is reduced and the current base classifier is reset and retrained. In the second phase of classifier training, the training uses Poisson sampling method in three cases in the data resampling phase to balance the current data by setting different values of λ parameter. The ensemble phase of the classifiers is performed using a dynamic classifier selection strategy to find the optimal quantity of ensemble classifiers at a time. The trained classifier weights are first ranked, and then, the search for the optimal quantity of ensemble classifiers K values is performed using an inverse search algorithm. The final generated ensemble classifiers are predicted using majority voting.

The DESW-ID algorithm considers only the binary classification problem. The samples are labeled as $Y = \{+1, -1\}$, where + 1 denotes the label of the minority class, and − 1 denotes the majority class. When evaluating the weights of the classifier, the classifier should give a greater penalty if misclassified minority class instances compared to misclassified majority class instances. $(x_i, y_i) \in S$ ith training instance, the cost of instance x_i misclassifier is denoted as C_i, as shown in Eq. (2).

$$C_i = \begin{cases} 1, & \text{if} = +1 \\ \text{ratio} = \dfrac{n\text{Postive}}{n\text{Negative}}, & \text{otherwise} \end{cases} \tag{2}$$

The ratio here is the imbalance ratio of the data stream window Win_{Ins}, which describes the ratio between the majority class instances and the minority class instances. nPostive and

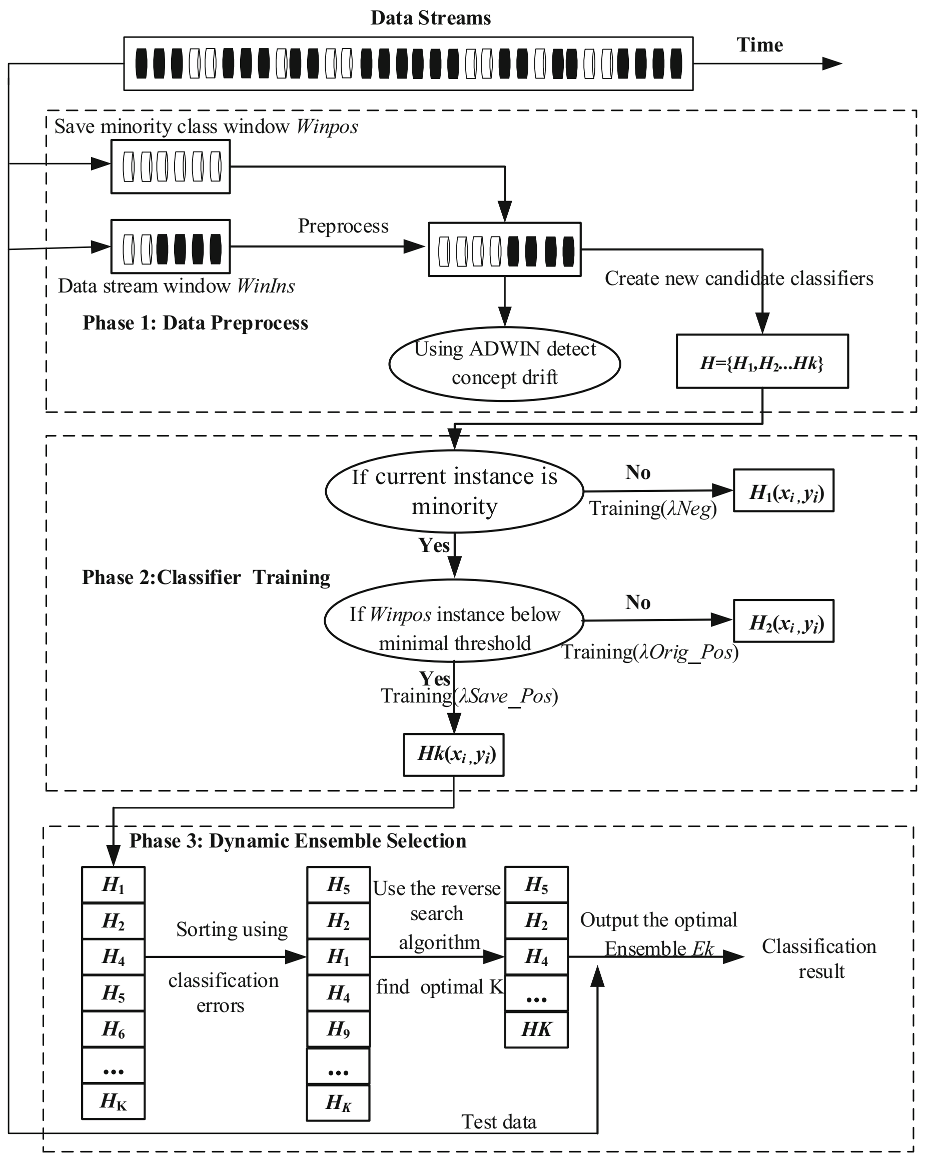
Data Streams
Time
Save minority class window Winpos
Preprocess
Data stream window WinIns
Create new candidate classifiers
Phase 1: Data Preprocess
Using ADWIN detect concept drift
H={H1,H2...Hk}
If current instance is minority
No
Training(λNeg)
H1(xi,yi)
Yes
Phase 2:Classifier Training
If Winpos instance below minimal threshold
No
H2(xi,yi)
Training(λOrig_Pos)
Yes
Training(λSave_Pos)
Hk(xi,yi)
Phase 3: Dynamic Ensemble Selection
H1 H2 H4 H5 H6 ... HK
Sorting using classification errors
H5 H2 H1 H4 H9 ... HK
Use the reverse search algorithm find optimal K
H5 H2 H4 ... HK
Output the optimal Ensemble Ek
Classification result
Test data

Fig. 2 Illustration of DESW-ID

nNegative in Eq. (2) are the number of minority class instances and the number of majority class instances of the current data stream window Win_{Ins}.

DESW-ID uses an online learning approach, which means that the classifier can be trained instantly as the data stream arrives. It is usually assumed that the instance in the nearest window is the most representative of the most recent data concept, so the newly created candidate classifier $H_m(1 \leq m \leq k)$ we consider as the perfect classifier because it is trained with the most current data and contains the most recent concept. At $i = 1$, k classifiers are created in the first window of data and all created classifiers weights are initialized to 1.

The mth component classifier H_m trained in the previous window is determined based on the classification performance of the instances on the latest window. A larger penalty will be given when a minority class instances are misclassified by the classifier. The weight of the component classifier is expressed as the equation as shown in Eq. (3) as the weight of the tth classifier at jth window, where $\mathrm{MSE_r}$ is the mean squared error and Eq. (4) represents the mean squared error of a readily predictable classifier. $p(y_i)$ denotes the class distribution share, i.e., the distribution of majority and minority classes. Avg_ε is the average error of the tth component classifier, where Eq. (5) ε_j is the error function of the tth component classifier, and the error function in this paper adopts the $G-$ mean, which considers the accuracy of both minority and majority classes, where C_i is the penalty factor of the classifier, if the minority class instances are misclassified will be given a bigger weight.

$$W_m = e^{\mathrm{MSE_r} - Avg_\varepsilon} \tag{3}$$

$$\mathrm{MSE_r} = \sum_{y_i} p(y_i)(1 - p(y_i))^2 \tag{4}$$

$$\mathrm{Avg}_\varepsilon = \frac{1}{|\mathrm{Win_{Ins}}|} \sum_{(x_i, y_i) \in S} C_i (1 - \varepsilon_i)^2 \tag{5}$$

$$\hat{y}_i = \arg\max \sum_{m=1}^{k} W_m^k * I\left(H_m\left(x_i'\right) = y_i'\right) \tag{6}$$

The $\hat{y}_i$ in Eq. (6) is the class label predicted by the final ensemble classifier at $x_i'(x_i' \in X, y_i' \in Y)$, where $H_m\left(x_i'\right)$ uses the label predicted by the classifier H_m.

In data stream classification, the way of updating the base classifier has been a key element of research, and most of the chunk based or online training methods are algorithmic innovations modeled on the AWE [22] algorithm framework. The algorithm itself uses a fixed ensemble size learning approach, and each time a new classifier is used to replace the worst classifier to maintain such a dynamic update process. However, there is a problem that replacing a worse classifier each time does not show that it is the best ensemble classifier overall, and the number of classifiers also affects the classification performance very much.

The DESW-ID algorithm uses the error of base classifier training for increasing ranking. Then, the threshold is used to search and traverse the base classifier set backward to find the optimal ensemble set. An advantage of this is that the number of classifiers can be adaptively adjusted to the number of ensemble classifiers according to the performance during training and testing, achieving another new way of dynamic updating, which the number of classifiers will eventually reach convergence behavior according to the training error. The specific pseudocode of the DESW-ID algorithm is shown in Algorithm 1.

Algorithm 1 DESW-ID

Input: Data streams S= {(x_1, y_2),.., (x_i, y_i)}, Number of classifiers K; Base classifier H_m ; Sorted set of classifiers $Sort_{H_m}$; Ensemble classifier E_k; Data stream window Win_{Ins}; threshold Δ; Drift detection $ADWIN$.

Output: Predictive label $\hat{y}_i$; Ensemble classifier E_k

```
1   Initialize $Win_{Ins} \leftarrow \emptyset$, Flag← False
2   While train $(x_i, y_i)$ do
3       ADWIN ← $y_i$ // Set $y_i$ into ADWIN
4       $Win_{Ins} \leftarrow (x_i, y_i)$//Cache data stream Instances to window
5     If ADWIN detect concept drift
6         Shrink ADWIN window size
7         Flag=True
8     Build new candidate classifiers $H_m$
9     For iterative ensemble classifier members do
10        Using algorithm 2 WS-PD to balanced data and train base classifiers
11        Compute classifier weight using Equation (3)
12        $Sort_{Hm} \leftarrow Sort_{classifiers}(\varepsilon_m, H_m)$ // Incremental sorting of classifiers by classification loss
13        End for
14    For iterative reverse traversal of a ranked ensemble classifier members
15    If $(0 < Sort_{H_m} - Sort_{H_{m-1}} \leq \Delta)$// IF the loss difference between two classifiers is within a specified interval, return current ensemble size
16    Return $K = m - 1$
17    Else
18    Return $K/2$ // If not found then return half the number of classifiers as the ensemble size
19    End for
20    For iterative test sets do
21      Assign $\hat{y}_i$ using Equation (6)
22    End For
```

Line 1 of the algorithm initializes the data stream window $\mathrm{Win}_{\mathrm{Ins}}$ and ADWIN drift detection. Lines 3–19 are the training plus updating process of the whole algorithm framework. Lines 3–4 store the y_i value of the current instance to the drift detector, while caching the incoming instances in the stream window. Lines 5–7 use the ADWIN drift detector to detect concept drift, shrink the ADWIN window if changes are detected, and set the Flag to True to reset the current trained classifier during the execution of Algorithm 2. In line 8, the candidate classifiers are constructed in the current data stream window. In lines 9–13, the newly constructed classifier is used to train the instances in the current window, and since the data stream itself is in an imbalanced state, Algorithm 2 WS-PD is called to balance the current data and train the classifier. Then, Eq. (5) is used to calculate the weights of the classifier to achieve the update of the classifier weights. To achieve dynamic selection of classifiers, the trained classifiers are sorted incrementally by loss in line 12 of the algorithm. In lines 15–19, the optimal ensemble size is found for the classifiers that have been sorted according to the inverse search, with the aim of finding the front from the classifier with the worst performance, which belongs to the learner with better classification performance. If the difference between the errors of two classifiers is within the set threshold, $m - 1$ is returned. If not found in this range, half of the number of classifiers is returned as the current ensemble size. Lines 20–22 use Eq. (7) to perform predictive classification of the examples.

3.2 The data balance process of the DESW-ID algorithm

DESW-ID is an online learning on based approach, which has the advantage of being able to train and test timely correction of classification models in real time. This algorithm proposes algorithm theory from Oza et al. [11] proposed an online learning framework, and the theory in the paper comes from the binomial distribution $\mathrm{Binomial}(p, N)$, which can be approximated as Poisson distribution $\mathrm{Poisson}(\lambda)$, and the condition that holds is that when $N \to \infty$ the parameters at this time can be expressed as $\lambda = Np$ where the successful probability p in the binomial distribution can be equated to the data distribution in the Bagging and Boosting algorithms. For example, the uniform sampling in Bagging algorithm can be approximated by $\mathrm{Poisson}(1)$, i.e., $\lambda = 1$, while for the online version of Boosting the parameter λ it can perform the weight calculation of correct and incorrect classification in the classification process. Also, according to Wang et al. [13] in their paper, the theory of Poisson distribution is used to realize the transformation from batch learning mode to online learning mode for both types of algorithms of Bagging and Boosting, which will eventually reach a state of convergence and make the online learning mode no less efficient than the batch learning mode.

The DESW-ID algorithm proposes three different resampling mechanisms in the preprocessing stage in order to balance the current imbalanced data stream. In the case of minority class, two sampling methods are used to increase the percentage of minority class, and since the percentage of minority class samples is severely under-represented in the initial stage of training, random oversampling will be performed by setting the Poisson distribution parameter $\lambda = 1$ for minority class to obtain high sample weights. As the minority class samples stored in the minority class window exceed the set minimum number threshold, the current imbalance will be balanced by adopting the samples in the minority class window with the sampling parameter $\lambda = (1 - a) * C$, where a is the current imbalance ratio, and C is the current share of the number of classifiers. In the case of majority class, the sample weights are reduced by setting the parameter $\lambda = a * C$ for random undersampling. A window is used in the DESW-ID algorithm to dynamically store continuously updated minority class instances, which is done to overcome the occurrence of concept drift that is easily triggered by the SMOTE algorithm when generating new minority class samples leading to the generation of new concepts. Also, to avoid that the instances generated by this algorithm are easily mixed with the majority class increasing the difficulty of classification, the window sampling based on Poisson distribution (WS-PD) method is proposed with specific details as shown in the pseudocode Algorithm 2.

Algorithm 2 WS-PD

Input: Data stream S= $\{(x_1, y_2),.., (x_i, y_i)\}$; Base classifier H_m, minimum number of minority class instances allowed to be stored in the minority class window $minSizeMinority$; minority class $nPositive$;majority class $nNegative$; save minority class instances window Win_{Pos} ;data stream window Win_{Ins}
Output: Ensemble classifier E_k;

```
1    $S$ ←incoming data stream
2    Initialize $Win_{Pos}, Win_{Ins} \leftarrow \emptyset$
3    Count $nPositive$ and $nNegative$
4    $C \leftarrow \frac{\text{nPositive}}{\text{nNegative}}$ // Calculate current imbalance ratio
5    If the current instance is a minority class, save it into the minority class window $Win_{pos} \leftarrow x_i^{pos}$
6    For each base classifier $m \leftarrow 1\ to\ K$
7        $a \leftarrow \frac{m}{K}$ // Calculate the percentage of current classifier
8      If current instance $y_i$ is minority class
9        If $|Win_{pos}| < minSizeMinority$ //If the current number of minority class window instances is below the set minority class instance threshold
10         $\lambda_{Original\_Pos} \leftarrow 1$ // Set the current minority class parameter
11         $k \sim Possion(\lambda_{Original\_Pos})$
12         Do $k$ times // repeat train $k$ times
13          Train $H_m(x_i, y_i)$
14         Else
15        $\lambda_{Save\_Pos} \leftarrow (1 - a) * C$// Set the currently stored minority class parameter
16        $k \sim Possion(\lambda_{Save\_Pos})$
17        Do $k$ times // repeat train $k$ times
18        $\bar{y}_i, \bar{x}_i \leftarrow Win_{Pos}(x_i, y_i)$ //adopt instances stored in a minority class window
19         Train $H_m(\bar{y}_i, \bar{x}_i)$
20   Else
21       $\lambda_{Neg} \leftarrow a * C$// Set the current majority class parameter
22       Do $k$ times
23         Train $H_m(x_i, y_i)$
24   If  Flag is  True
25    Reset current classifier $H_m$ to learning new instances
26   End For
27   Return $E_k \leftarrow H_m \cup H$
```

Lines 1–2 of the algorithm pass in the data stream and then initialize the current minority class and data stream windows. Lines 3–5 count the current minority class and majority class instances. Then calculate the current imbalance ratio, and cache the current instance to the current minority class window if it is a minority class sample. Lines 6–26 are the detailed process of balancing the current data stream for the classifier for training. In line 7, the percentage of each base classifier is calculated a. Lines 8–19 are the sampling process for the insufficient amount of minority class samples and the minority classes in the minority class window that exceed the set threshold number. If the current sample is the original minority class, the parameter is set $\lambda_{\text{Original\_Pos}}$=1. Then, the number of training is calculated by Poisson distribution to be k times. Base classifier will be repeatedly trained k times. And if the minority class stored in the current minority class window is larger than the set minority class threshold, the minority class in the Win_{Pos} window will be sampled, and it has $\lambda_{\text{Save\_Pos}} = (1 - a) * C$, while the k values calculated using Poisson distribution will be used to train the current base classifier for k times with the data in the minority class window. In lines 20-23, if the instance in the data stream window is majority class samples for undersampling, set the current parameter to $\lambda_{\text{Neg}} = a * C$. Then the samples are trained using the current base classifier. Lines 24-26, The Flag variable is set to True if concept drift occurs. Reset the current base classifier and train the classifier with the new data sample. Line 27 outputs the trained ensemble classifier set.

4 Experiment design

In this section, this article proves the effectiveness of the proposed algorithm through experiments. The DESW-ID algorithm is compared with the other 6 imbalance state-of-the-art techniques. Five artificial datasets and two real-world datasets are used for imbalance evaluation indicators.

4.1 Experimental dataset

Table 1 shows the detailed information of the artificial data set and the real data set used in the experiment. Due to the lack of proper real-world assessment of imbalanced data stream

Table 1 Dataset

Dataset	Instance (k)	Attributes	Classes	Noise (%)	IR (%)	Drifts	Drift type
SEAs	600	3	2	10	10	4	Sudden
SEAg	500	3	2	10	10	10	Gradual
Hyper	1000	10	2	5	10	1	Increment
RanRBF	800	20	2	10	10	4	Gradual recurrent
SEAsr	800	3	2	10	10	4	Sudden recurrent
Poker	360	10	2	–	10	–	–
CovType	266	54	2	–	10	–	–

classification, the artificial data set can control the concept drift and imbalance ratio control in the data stream. The setting process of the data set will be described in detail below.

SEA: Use the SEA generator [23] to create three data sets, each of which contains 10% noise. First, SEAS contains four sudden shifts (Sudden), the position of which is 150,000. Secondly, SEAsr is designed to contain four sudden changes in periodic drift (Sudden recurrent) with a position of 200,000. Third, ten gradual drifts are introduced on the SEAG data set, and the drift position is set to 50,000.

RanRBF: Random radial basis function (RBF) generator [2] creates a new instance by randomly selecting a center, where each center has a weight. The center with a high weight has a high probability of being selected. This generator is used to create a RanRBF data set described by 20 attributes and two classes. Simulate the drift of four gradual cycles by moving the center of mass at a constant speed (Gradual Recurrent).

Hyper: Hyperplane generator [24] can simulate incremental conceptual drift (Increment) by smoothly adjusting the direction and position of the rotating hyperplane. Use this generator to create a Hyper data set containing 100,000 observations, which are described by 10 attributes and two classes. The incremental drift is simulated by changing the modified weight of 0.1 for each instance, and then, 5% of the noise is added to the data.

Poker and CovType [25] are two real-world data sets, which contain 10 and 54 attributes, respectively, but the number and type of concept drift within them are unknown.

4.2 State-of-the-art methods

The experiment uses several advanced algorithms to compare their ability to learn concept drift from imbalanced data. All tested algorithms are implemented in Java. The tested algorithm is as follows:

Stratified Bagging (SBag) [26]: The algorithm proposes a new framework for hierarchical Bagging, where the data on the data blocks are balanced at the bottom layer using an oversampling technique, and the data are trained using dynamic classifier selection.

C-SMOTE (CS) [1]: The algorithm uses the continuous use of the SMOTE algorithm when the set imbalance reaches 0.5 before the next update, and the ADWIN drift detector is used to detect the drift.

Rebalance Stream (RS) [27]: The algorithm uses ADWIN to detect data streams with concept drift and trains four models in parallel. In this paper, SMOTE resampling technology is used to rebalance the data flow. If drift occurs, the best model is selected to reset other models.

Oversampling Online Bagging (OOB) and Undersampling Online Bagging (UOB) [14] are two other ensemble methods based on resampling. When the class imbalance is detected, the oversampling or undersampling embedded in the online Bagging is triggered to increase the chance of training minority samples or reduce the chance of training majority samples. Online AdaBoost(OzaBoost) [13]: It is the classic online learning version of Boosting algorithm. Each iteration of the algorithm pays more attention to the learning difficulty samples. In the imbalanced classification, the minority class is the difficult group that needs more attention, so it is better to use it as the comparison algorithm.

In order to make the comparison more meaningful, the algorithms in the experiment all use Hoeffding Tree (HOT) [28] as the base classifier, where the number of classifiers is set to $m = 15$, the experimental data set imbalance ratio is 10%, the search loss threshold is based on the experimental setting is $\Delta = 0.05$ and data stream size of window $\mathrm{Win}_{\mathrm{Ins}} = 1000$, store minority instances size of window $\mathrm{Win}_{\mathrm{Pos}} = 200 * \mathrm{Win}_{\mathrm{Ins}}$.

Table 2 Confusion matrix

Class	Positive	Negative
Positive	TP	FN
Negative	FP	TN

4.3 Evaluation indicator

In the evaluation index, since the accuracy rate describes the overall recognition performance of the algorithm on the test observation, it is usually used for traditional classification. Accuracy is mainly determined by the majority class, so it is not an appropriate indicator for imbalanced datasets. The metrics used in this experiment are balanced accuracy, F-measure, G-mean, recall, and AUC. Following confusion matrix will be used to show how these metrics are computed, mainly with a binary classification.

The confusion matrix is shown in Table 2, where TP indicates that the positive class sample prediction is still positive, FN indicates that the positive class sample prediction is negative, FP indicates that the negative class sample prediction is positive, and TN indicates that the negative class prediction is still negative.

$$A^{+} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}, \quad A^{-} = \frac{\mathrm{TN}}{\mathrm{TN+FP}} \tag{7}$$

$$\text{Balanced Accuracy} = \frac{A^{+}+A^{-}}{2} \tag{8}$$

$$G - \text{mean} = \sqrt{A^{+} * A^{-}} \tag{9}$$

$$\text{Recall} = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \tag{10}$$

$$F - \text{measure} = \frac{2\mathrm{TP}}{2\mathrm{TP+FP+FN}} \tag{11}$$

where the AUC value is the area under the ROC curve (Receiver Operating Characteristic), AUC is a comprehensive evaluation of classification models, which can provide more useful information than accuracy measurement.

4.4 Experiment environment

In order to verify the performance of the DESW-ID algorithm, the experiment uses a data stream mining analysis platform MOA (Massive Online Analysis) [29]. The hardware environment of this experiment is a PC with Intel Core i5 1 T + 128RAM, and the operating system is Windows10 Professional Edition, and the programming language is Java.

5 Analysis of experiment result

5.1 The influence of imbalance ratio on DESW-ID algorithm

The purpose of this experiment is to analyze the effect of different imbalance ratios on the performance of the proposed algorithm. The imbalance ratio of the data, i.e., the ratio of the number of minority classes to the number of majority classes, can directly affect the

Table 3 AUC value of the imbalance ratio on artificial dataset

Datasets	6%	7%	8%	9%	10%
SEAs	0.8747 (5)	0.8904 (3)	0.8886 (4)	0.8970 (2)	**0.8987 (1)**
SEAg	0.8706 (5)	0.9039 (4)	0.9015 (3)	0.9068 (2)	**0.9116 (1)**
SEAsr	**0.9054 (1)**	0.9019 (5)	0.9028 (4)	0.9030 (3)	0.9037 (2)
Hyper	0.9127 (5)	0.9194 (4)	0.9274 (3)	0.9287 (2)	**0.9340 (1)**
RanRBF	0.9612 (5)	0.9623 (4)	0.9632 (1)	0.9629 (3)	**0.9761 (1)**
Average rank	4.20	4.00	3.00	2.40	1.40

performance of the model. A smaller imbalance ratio means that the probability of obtaining a minority class sample is smaller, and therefore, the classification task becomes more difficult. In this experiment, five artificial datasets with five levels of imbalance equivalence, i.e., 6, 7, 8, 9 and 10% minority class ratios are considered and the best performers are bolded, while the average ranking of AUC performance is given.

Table 3 shows the performance of the proposed algorithm DESW-ID for AUC on datasets with different minority class ratios, and the best results are marked using bold. It can be seen from the table that the performance of AUC improves as the minority class ratio increases, where it can be seen that the algorithm performs best with the minority class ratio at 10% and has the highest average ranking. With more minority class ratio in the data stream, the classifier can fit more minority class instances. Also, as the imbalance ratio decreases, the performance of the algorithm does not decrease much, due to the fact that the algorithm uses preserved trained minority class. Therefore, it can guarantee enough number to reach balance with the majority class. Therefore, from the above experimental data, it can be concluded that the algorithm can achieve the equilibrium state without affecting the classification performance by using a window to store the constantly updated minority classes for sampling.

5.2 The influence of base classifier on DESW-ID algorithm

The experiment in this section explores the influence of the base classifier type on the ensemble structure. Since the structure of each classifier model is different, the efficiency and classification performance of their training examples are also different. This experiment uses three commonly used base classifiers Naïve Bayes (NB) [30], KNN [30] and Hoeffding Tree. NB is a classification algorithm proposed by Bayesian theory, which has the advantage of low training and prediction time complexity. KNN is a lazy learning method based on distance measurement, and HOT is a decision tree algorithm, which is incremental from the data stream. The decision tree generated by the equation has fast processing and can predict new samples at any point in time.

Figure 3 shows the AUC performance of different classifier types on all data sets, where the AUC value is the average of all training times. It can be seen from the figure that the KNN classifier has a lower AUC value than the other two classifiers, indicating that the classifier cannot quickly obtain robust training during the training process of the data stream. The NB classifier can train the incoming data smoothly due to its own advantages. The HOT classifier is the most used classifier on the MOA platform. Compared with the other two classifiers, it can quickly adapt to the data stream instance and perform incremental training, processing

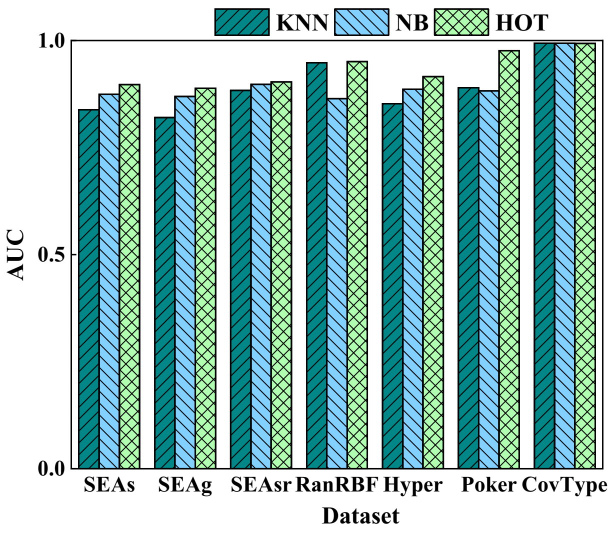

Fig. 3 AUC value of different base classifiers

the elements in the data stream only once. Therefore, other experiments in this article use HOT as the base classifier to combine into the final ensemble classifier.

5.3 Comparison of DESW-ID with other DES algorithms

Since the algorithm proposed in this paper uses a dynamic selection ensemble strategy in the composition structure of the classifier, it is necessary for the experiments to compare with other advanced dynamic ensemble selection strategies to show the advantageous performance of the algorithm. The experiments are also compared with six other advanced DES algorithms. Where the area under the ROC curve (AUC) is used as the metric for this experiment, higher AUC values indicate that the algorithm has better classification results, and Table 4 shows the best results of the algorithm on all datasets, while black bold is used to indicate the best AUC values on each dataset. From Table 4, we can see that the DESW-ID algorithm proposed in this paper is better than the other six DES algorithms on most of the datasets.

Table 4 The classification performance of various methods on all datasets

AUC	KNORAU	KNORAE	DES-P	DES-MI	DES-KNN	Meta-DES	DESW-ID
SEAs	0.7185	0.6873	0.7163	0.6660	0.7194	0.7297	**0.8987**
SEAg	0.7277	0.6915	0.7232	0.6664	0.7236	0.7396	**0.9015**
SEAsr	0.7202	0.6880	0.7153	0.6658	0.7208	0.7353	**0.9028**
RanRBF	**0.9851**	0.9821	0.9846	0.9729	0.9799	0.9716	0.9761
Hyper	0.7334	0.6828	0.7273	0.6700	0.7196	0.7450	**0.9274**
Poker	0.9663	0.9746	0.9692	0.9288	0.9715	0.9714	**0.9763**
CovType	0.9751	0.9771	0.9736	0.9617	0.9776	0.9749	**0.9837**

Table 5 Results of the Wilcoxon signed-rank test comparison of the proposed algorithm with well know DES algorithms

Comparison	Hypothesis	*P* value
DESW-ID versus KNORAU	Rejected at 5%	0.046875*
DESW-ID versus KNORAE	Rejected at 5%	0.046875*
DESW-ID versus DES-P	Rejected at 5%	0.046875*
DESW-ID versus DES-MI	Rejected at 5%	0.015625*
DESW-ID versus DES-KNN	Rejected at 5%	0.03125*
DESW-ID versus Meta-DES	Rejected at 5%	0.03125*

A "$*$" near the p value means that there are statistical differences with $\alpha = 0.05$ (95% confidence)

The experimental dataset is with concept drift, so the algorithm DESW-ID is adaptable to most types of concept drift. The selection strategy in this paper is to use error weighting while giving a larger penalty to a minority classes of misclassified classifiers. The weighted weights are then ranked and a fixed threshold is set to adaptively select the best number of classifiers. To certify the performance of the algorithm more rigorously the Wilcoxon signed-rank test was also performed, and it can be shown from the statistical results in Table 5 that all hypotheses are rejected, set at $\alpha = 0.05$ (95% confidence level). Because the p values obtained were lower than $\alpha = 0.05$. We can indicate that the DESW-ID algorithm proposed in this paper is superior to the comparison algorithms based on the rejection in the Table 5.

5.4 Comparative analysis of DESW-ID versus state-of-the-art methods

In this section, DESW-ID is compared with six other state-of-the-art methods on five artificial and two real datasets. Each compared algorithm is evaluated using five metrics, balanced accuracy (Balanced Acc), F-measure, G-mean, and recall, AUC. The results of the algorithm performance comparison are shown in Table 6, and the best-performing algorithm performance metrics are bolded. Also, the ranking of the metrics of the compared algorithms on all datasets is shown in Table 7. Due to space limitations, Fig. 4 provides performance comparison charts for only one representative dataset of SEAs.

In Table 7, we can observe the ranking of indicators for each algorithm on all datasets. The DESW-ID algorithm proposed in this paper is in the first place in all data sets. At the same time, CS and RS algorithm ranked second and third. In Table 6, the DESW algorithm is in the optimal position in the gradual SEA dataset and sudden SEAsr dataset under the five evaluation indexes. This is due to the fact that the resampling technique of the DESW-ID algorithm takes into account both data imbalance factors and conceptual drift and is used to improve the recognition rate of the model for a small number of classes, and it performs well on most of the datasets through the ADWIN drift detection method and the dynamic selection of the number of classifiers in a way that updating the ensemble members can quickly respond to different types of drift. While the CS and RS algorithms themselves use online learning, they use the SMOTE sampling technique to generate a minority classes for balancing on a continuous stream of imbalanced data. CS simultaneously removes obsolete data to continuously learn new knowledge, and RS is selected to train four models using K-statistics to pick the best one. Among them, CS has good performance in these three metrics, while RS's AUC improvement on the SEAsr dataset is at the cost of G-mean, recall and F-measure, which cannot quickly cope with the conceptual drift of abrupt repetition. SBag algorithm uses

Table 6 Performance comparison of different algorithms on all datasets

Dataset	Methods	Balanced Acc	F-measure	G-mean	Recall	AUC
SEAs	UOB	0.8326 (3)	0.4428 (4)	0.8159 (4)	0.8259 (3)	0.8257 (5)
	OOB	0.7312 (5)	0.3974 (6)	0.7081 (6)	0.5374 (6)	0.7359 (7)
	RB	0.6833 (6)	0.4501 (3)	0.8393 (3)	0.7905 (4)	0.8736 (4)
	CS	0.8757 (2)	**0.6255 (1)**	0.8493 (2)	0.8879 (2)	0.8780 (3)
	OzaBoost	0.6736 (7)	0.4345 (5)	0.6097 (7)	0.3936 (7)	0.8851 (2)
	SBag	0.7756 (4)	0.3000 (7)	0.7182 (5)	0.7184 (5)	0.7783 (6)
	DESW-ID	**0.8769 (1)**	0.6198 (2)	**0.8762 (1)**	**0.8992 (1)**	**0.8987 (1)**
SEAg	UOB	0.8391 (3)	0.4576 (3)	0.8242 (4)	0.8141 (3)	0.8214 (5)
	OOB	0.7307 (5)	0.4248 (5)	0.6752 (6)	0.4876 (6)	0.7124 (7)
	RB	0.6856 (6)	0.4516 (4)	0.8326 (3)	0.7665 (4)	0.8652 (4)
	CS	0.8716 (2)	0.6175 (2)	0.8414 (2)	0.8327 (2)	0.8702 (3)
	OzaBoost	0.6685 (7)	0.4232 (6)	0.6016 (7)	0.3838 (7)	0.8931 (2)
	SBag	0.7692 (4)	0.2929 (7)	0.7079 (5)	0.7078 (5)	0.7719 (6)
	DESW-ID	**0.8808 (1)**	**0.6204 (1)**	**0.8501 (1)**	**0.8629 (1)**	**0.9015 (1)**
SEAsr	UOB	0.8326 (3)	0.4430 (4)	0.8591 (3)	0.8344 (3)	0.8606 (5)
	OOB	0.7321 (5)	0.3985 (6)	0.7493 (5)	0.6023 (5)	0.7678 (7)
	RB	0.6776 (6)	0.4432 (3)	0.6136 (6)	0.3995 (6)	0.8905 (3)
	CS	0.8885 (2)	0.6380 (2)	0.8868 (2)	0.8969 (2)	0.8868 (4)
	OzaBoost	0.6691 (7)	0.4332 (5)	0.6021 (7)	0.3821 (7)	0.8928 (2)
	SBAG	0.8324 (4)	0.3715 (7)	0.7972 (4)	0.7972 (4)	0.8333 (6)
	DESW-ID	**0.8924 (1)**	**0.6454 (1)**	**0.8880 (1)**	**0.9021 (1)**	**0.9028 (1)**
RanRBF	UOB	0.7423 (6)	0.3221 (7)	0.7422 (7)	0.7239 (5)	0.6388 (7)
	OOB	0.3548 (7)	0.7070 (4)	0.7439 (5)	0.6049 (6)	0.7609 (6)
	RB	**0.9529 (1)**	0.9298 (2)	0.9165 (2)	0.8456 (2)	0.9809 (2)
	CS	0.8067 (3)	0.4706 (6)	0.7942 (4)	0.8000 (3)	0.8817 (5)
	OzaBoost	0.7784 (5)	0.7053 (5)	0.7427 (6)	0.5580 (7)	0.9548 (4)
	SBag	0.7792 (4)	**0.9754 (1)**	**0.9762 (1)**	**0.9918 (1)**	**0.9948 (1)**
	DESW-ID	0.8078 (2)	0.7119 (3)	0.8013 (3)	0.7338 (4)	0.9761 (3)
Hyper	UOB	**0.9354 (1)**	0.6617 (2)	**0.9348 (1)**	**0.9492 (1)**	0.5272 (7)
	OOB	0.7834 (4)	0.6158 (3)	0.6747 (5)	0.4686 (6)	0.7218 (6)
	RB	0.7181 (6)	0.5299 (5)	0.7711 (4)	0.7206 (3)	0.8716 (4)
	CS	0.8772 (2)	0.6150 (4)	0.8407 (2)	0.7792 (4)	0.9145 (2)
	OzaBoost	0.6509 (7)	0.4258 (6)	0.5567 (7)	0.3197 (7)	0.9171 (3)
	SBAG	0.7792 (5)	0.3405 (7)	0.6329 (6)	0.6945 (5)	0.7845 (5)
	DESW-ID	0.8009 (3)	**0.6833 (1)**	0.8019 (3)	0.7908 (2)	**0.9274 (1)**
Poker	UOB	0.6512 (3)	0.2328 (6)	0.6405 (2)	0.6674 (5)	0.5533 (7)
	OOB	0.6113 (6)	0.2659 (5)	0.5095 (5)	0.3799 (6)	0.6061 (5)
	RB	0.6239 (5)	0.9682 (3)	0.4393 (6)	0.9824 (3)	0.8311 (4)
	CS	0.6763 (4)	0.9078 (4)	0.5915 (3)	0.9008 (4)	0.8723 (3)
	OzaBoost	0.6805 (2)	0.9851 (2)	0.5780 (4)	0.9862 (2)	0.9605 (2)
	SBag	0.3569 (7)	0.2001 (7)	0.2672 (7)	0.2121 (7)	0.5931 (6)

Table 6 (continued)

Dataset	Methods	Balanced Acc	F-measure	G-mean	Recall	AUC
	DESW-ID	**0.8336 (1)**	**0.9873 (1)**	**0.8113 (1)**	**0.9894 (1)**	**0.9763 (1)**
CovType	UOB	0.7473 (7)	0.5013 (7)	0.7162 (6)	0.6059 (6)	0.6715 (7)
	OOB	0.7596 (6)	0.5139 (6)	0.6702 (7)	0.5657 (7)	0.7392 (6)
	RB	0.9556 (2)	**0.9915 (1)**	0.9539 (2)	0.9865 (2)	0.9825 (3)
	CS	0.8817 (5)	0.9142 (4)	0.8557 (4)	0.8970 (4)	0.9145 (4)
	OzaBoost	0.9554 (3)	0.9880 (3)	0.9533 (3)	0.9807 (5)	0.9808 (2)
	SBag	0.8390 (4)	0.6861 (5)	0.8029 (5)	0.7920 (4)	0.8510 (5)
	DESW-ID	**0.9667 (1)**	0.9895 (2)	**0.9662 (1)**	**0.9894 (1)**	**0.9837 (1)**

Table 7 Mean ranks of seven comparative on all datasets

Rank	UOB	OOB	RB	CS	OzaBoost	SBag	DESW-ID
Balanced Acc	3.71	5.43	4.57	2.85	5.43	4.57	1.42
F-measure	4.71	5.00	3.00	3.28	4.57	5.85	1.43
G-mean	3.86	5.57	3.71	2.71	5.86	4.71	1.43
Recall	3.71	6.00	3.43	3.00	6.71	4.43	1.14
AUC	6.61	6.29	3.43	3.43	2.43	5.00	1.28

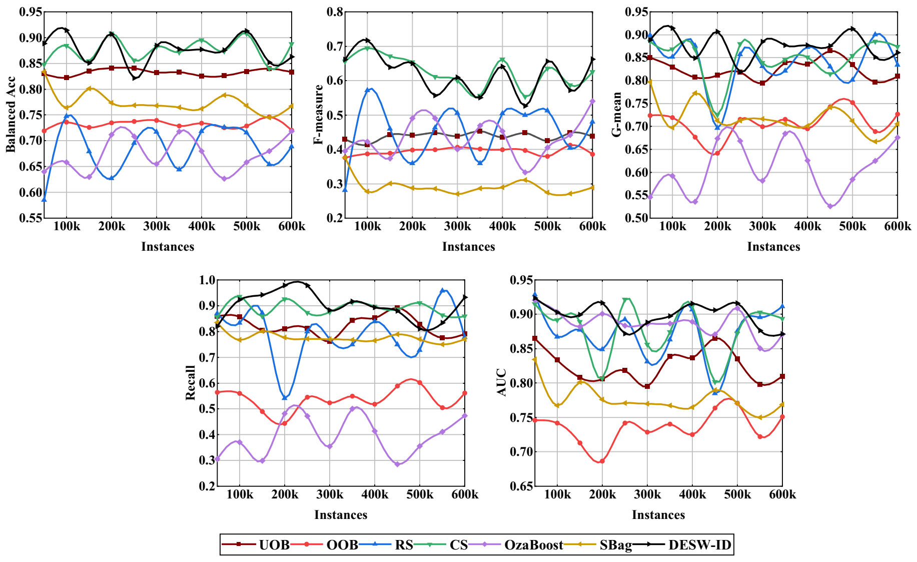

Fig. 4 The classification performance of comparative algorithms on SEAs dataset

resampling technique to balance the dataset at the bottom, and then uses hierarchical Bagging to train the balanced set, and finally uses dynamic selection for ensemble. It performs best on the RanRBF dataset with F-measure, G-mean, recall, and AUC. Meanwhile, OOB and UOB algorithms using resampling and time decay techniques have better performance on SEAs, SRAg, SEAsr and Hyper datasets, where UOB ranks first in Hyper dataset on Balanced Acc, G-mean, and Recall datasets. However, these two perform poorly on the remaining three datasets because their conceptual drift and imbalance processing mechanisms are too simple to accommodate complex situations. As for OzaBoost, due to its lack of handling mechanism for concept drift and imbalance, it shows low F-measure, G-mean and Recall values in most of the datasets. Based on the above analysis, it can be concluded that DESW-ID can have a good performance on a minority class instances without sacrificing the performance on most class instances.

Figure 4 shows the performance variation of each metric of the compared algorithms on the SEAs dataset. The OzaBoost algorithm has a high value of AUC variation in Fig. 4, but the other metrics have a poor performance because the algorithm itself is not able to adapt to abrupt concept drift and does not have the ability to handle imbalanced data. At the same time, we see that the comparison algorithm CS fits the data well in the trend of all five indicators. In the figure, the DESW-ID proposed in this paper is compared with other comparison algorithms in that it can quickly recover from the stable period and maintain a high-performance variation after the concept drift occurs.

5.5 Analysis of DESW-ID algorithm time efficiency

In this section, the running time efficiency of the comparison algorithm is discussed. Table 8 shows the time consumption of the comparison algorithms on all datasets. First, it can be seen that the SBag and CS algorithms consume a significant amount of time during the training process. Since the SBag algorithm needs to perform stratified Bagging sampling each time, as well as the oversampling process eventually using a dynamic selection strategy for ensemble, the whole process takes some time. CS algorithm also needs the process of data sampling, but its sampling process consumes more memory. Because it requires oversampling with SMOTE every time to get the balance ratio of 0.5. The DESW-ID algorithm proposed in this paper is similar in time consumption compared to the rest of the advanced algorithms and does not consume much time. Because the algorithm does not generate minority class samples, where the time for classifier ensemble and training is greatly reduced due

Table 8 Running time of comparison algorithms

Dataset	DESW-ID	SBag	OzaBoost	CS	RS	OOB	UOB
SEAs	16.19	10,518.51	56.776	3624.00	60.00	63.02	80.54
SEAg	13.94	8306.01	39.20	2672.00	43.95	51.67	53.82
SEAsr	24.28	13,323.73	70.03	7448.00	108.00	83.68	86.06
RandRBF	74.00	30,737.03	164.45	13,488.96	310.00	94.79	97.25
Hyper	54.61	25,969.38	293.06	20,182.59	253.00	109.95	110.99
Poker	109.80	7027.05	29.70	66.00	4.80	101.86	84.02
CovType	181.68	13,563.75	144.23	114.00	10.59	315.63	297.87

to the number of ensemble classifiers that are dynamically pruned. Therefore, the proposed algorithm has satisfactory time efficiency and is suitable for mining imbalanced data stream.

6 Conclusion

Since there may be both class imbalance and concept drift in the data stream, which can greatly hinder the performance of classification models, this paper proposes a classification algorithm based on classifier ensemble selection. The problem of insufficient number of minority classes is solved by using a Poisson distributed window sampling method, while avoiding the introduction of new concepts. For the trained classifiers are sorted incrementally using error and the optimal number of ensemble classifiers are found using the inverse search algorithm for the sorted classifiers using differences. The proposed algorithm is experimentally verified to improve the recognition rate of minority classes, and the time efficiency is also improved due to the online learning approach. With the frequent occurrence of bank card fraud in recent years, where the occurrence of fraud is a minority class event, with the continuous renovation of fraudulent means, there is a great need for an algorithm to help banks to identify the occurrence of fraud, and the proposed one in this paper can provide meaningful guidance for imbalanced data stream mining with concept drift.

Due to the complexity of imbalanced data stream, resampling past samples not only suffers from concept drift, but also from overlap between classes, high local imbalance, and how to update and balance the trained classifiers in the past. These challenges of imbalanced data stream classification will be the focus of future research.

Author contributions MH completes the main work; XZ completed the coding of the model, some experiments and the writing of the main paper; ZC completed some experiments and the production of experimental diagrams; HW and ML participated in the coordination of the study and reviewed the manuscript. All authors read and approved the final manuscript.

Funding This work was supported by the National Nature Science Foundation of China (62062004), the Ningxia Natural Science Foundation Project (2020AAC03216, 2022AAC03279) and the Graduate Innovation Project of North Minzu University (YCX21085).

Declarations

Conflict of interest No potential conflict of interest was reported by the authors.

Human or animal rights With the unanimous consent of all our authors, the paper is only about a research on a machine learning algorithm and does not involve Human Participants and/or Animals. All data are open source and do not involve the interests of others.

References

1. Bernardo A, Gomes HM et al (2020) C-SMOTE: continuous synthetic minority oversampling for evolving data streams. In: Proceedings of the IEEE international conference on big data, pp 483–492
2. Ren SQ, Zhu W, Li Z et al (2018) The Gradual Resampling Ensemble for mining imbalanced data streams with concept drift. Neurocomputing 286:150–166
3. Li H, Wang Y, Wang H (2017) Multi-window based ensemble learning for classification of imbalanced streaming data. World Wide Web 20(6):1507–1525
4. Gama J, Medas P (2004) Learning with drift detection detection. Adv Artif Intell 3171:286–295

5. Baena-Garc M, Campo-Ávila JD, Fidalgo-Merino R et al (2006) Early drift detection method. In: International workshop on knowledge discovery from data streams, vol 6, pp 77–86
6. Bifet A, Gavalda R (2007) Learning from time-changing data with adaptive windowing. In: Proceedings of the seventh SIAM international conference on data mining, pp 443–448
7. Ren SQ, Zhu W, Liao B et al (2019) Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning. Knowl Based Syst 163:705–722
8. Chawla NV, Lazarevic A, Hall LO et al (2003) SMOTEBoost: improving prediction of the minority class in boosting. In: Proceedings of knowledge discovery in databases: PKDD 2003, vol 2838. Springer, Berlin, pp 107–109
9. Du HL, Zhang Y, Gang K et al (2021) Online ensemble learning algorithm for imbalanced data stream. Appl Soft Comput 107:107378
10. Sun Y, Kamel MS, Wong AKC et al (2007) Cost-sensitive boosting for classification of imbalanced data. Pattern Recogn 40(12):3358–3378
11. Oza NC, Russell S (2005) Online bagging and boosting. In: Proceedings of artificial intelligence and statistics, pp 105–112
12. Barros RSM, Santos SGT (2016) A boosting-like online learning ensemble. In: Proceedings of the 26 international joint conference on neural networks, pp 1871–1878
13. Wang BY, Pineau J (2016) Online bagging and boosting for imbalanced data stream. IEEE Trans Knowl Data Eng 28(12):3353–3366
14. Wang S, Minku LL, Yao X (2015) Resampling-based ensemble methods for online class imbalance learning. IEEE Trans Knowl Data Eng 27(5):1356–1368
15. Hou WH, Wang XK, Zhang HY et al (2020) A novel dynamic ensemble selection classifier for an imbalanced dataset: an application for credit risk assessment. Knowl Based Syst 208:106462
16. Ko AHR, Sabourin R, Britto AS et al (2008) From dynamic classifier selection to dynamic ensemble selection. Pattern Recognit 41:1718–1731
17. Woloszynski T, Kurzynski M, Podsiadlo P et al (2012) A measure of competence based on random classification for dynamic ensemble selection. Inf Fusion 13:207–213
18. Soares RGF, Santana A, Canuto AMP et al (2006) Using accuracy and diversity to select classifiers to build ensembles. In: Proceedings of IEEE international joint conference on neural network, Vancouver, Canada, pp 1310–1316
19. Cruz RMO, Sabourin R, Cavalcanti GDC et al (2015) META-DES: a dynamic ensemble selection framework using meta-learning. Pattern Recogn 48:1925–1935
20. García S, Zhang ZL, Altalhi A et al (2018) Dynamic ensemble selection for multi-class imbalanced datasets. Inf Sci 445:445–446
21. Zhang XL, Han M, Chen ZQ, Wu HX, Li MH (2021) An overview of complex data stream ensemble classification. J Intell Fuzzy Syst 41(2):3667–3695
22. Wang H, Fan W, Yu PS (2003) Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining, pp 226–235
23. Street WN, Kim Y (2001) A streaming ensemble algorithm (sea) for large-scale classification. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining, pp 377–382
24. Hulten G, Spencer L, Domingos P (2001) Mining time-changing data streams. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining, pp 97–106
25. Cano A, Krawczyk B (2020) Kappa updated ensemble for drifting data stream mining. Mach Learn 1(109):178–218
26. Zyblewski P, Sabourin R, Wozniak M (2021) Preprocessed dynamic classifier ensemble selection for highly imbalanced drifted data streams. Inf Fusion 66:138–154
27. Bernardo A, Valle DE, Bifet A (2020) Increment rebalancing learning on evolving data streams. In: Proceedings of the 20th international conference on data mining workshops (ICDM), pp 844–850
28. Domingos P, Hulten G (2000) Mining high-speed data streams. In: Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining, pp 71–80
29. Bifet A, Holmes G, Kirkby R, Pfahringer B (2010) Moa: massive online analysis. J Mach Learn Res 11:1601–1604
30. Lemaire V, Salperwyck C, Bondu A (2015) A survey on supervised classification on data streams. Lecture Notes Bus Inf Process 205:88–125

Meng Han born in 1982, PhD, professor, main research area is data mining.

Xilong Zhang born in 1996, is a master's student, whose main research area is data mining.

Zhiqiang Chen born in 1998, is a master's student, whose main research area is data mining.

Hongxin Wu born in 1998, is a master's student, whose main research area is data mining.

Muhang Li born in 1997, is a master's student, whose main research area is data mining.