Hindawi
Computational Intelligence and Neuroscience
Volume 2021, Article ID 8813806, 9 pages
https://doi.org/10.1155/2021/8813806

Research Article

Cost-Sensitive Classification for Evolving Data Streams with Concept Drift and Class Imbalance

Yange Sun,[1,2] **Meng Li**,[1] **Lei Li**,[1] **Han Shao**,[1] **and Yi Sun**[3]

[1]*School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, China*
[2]*Henan Key Lab of Analysis and Applications of Education Big Data, Xinyang Normal University, Xinyang, China*
[3]*Institute of Zhengzhou Information Science and Technology, Zhengzhou, China*

Correspondence should be addressed to Yange Sun; ygsun1982@126.com

Received 7 August 2020; Revised 4 July 2021; Accepted 21 July 2021; Published 2 August 2021

Academic Editor: Jussi Tohka

Class imbalance and concept drift are two primary principles that exist concurrently in data stream classification. Although the two issues have drawn enough attention separately, the joint treatment largely remains unexplored. Moreover, the class imbalance issue is further complicated if data streams with concept drift. A novel Cost-Sensitive based Data Stream (CSDS) classification is introduced to overcome the two issues simultaneously. The CSDS considers cost information during the procedures of data preprocessing and classification. During the data preprocessing, a cost-sensitive learning strategy is introduced into the ReliefF algorithm for alleviating the class imbalance at the data level. In the classification process, a cost-sensitive weighting schema is devised to enhance the overall performance of the ensemble. Besides, a change detection mechanism is embedded in our algorithm, which guarantees that an ensemble can capture and react to drift promptly. Experimental results validate that our method can obtain better classification results under different imbalanced concept drifting data stream scenarios.

1. Introduction

Data stream classification has attracted much attention in the scenario of big data mining due to its presence in many real-world fields, such as social network analysis, weather prediction, online medical diagnosis, and weblog mining [1–5]. Concept drift is a common feature of data streams [6–9], which refers to the phenomenon of target concepts of streams changing over time. Concept drift can deteriorate the performance of classification because the model trained on old concepts may be unsuitable for new concepts. For example, fashion trends in recommend systems may be influenced by customer behavior, and the weather forecast model may no longer be applicable as the season changes. Therefore, an efficient data stream learning model should have the capability of capturing drifts promptly and updating the model accordingly [7, 10].

A growing number of methodologies have been proposed for dealing with concept drift [9]. Among these techniques, the window-based method adopts a natural way of forgetting mechanism to add new instances and eliminate outdated instances. The sliding window is the most frequently used window technology. It adopts the first-in-first-out structure to move on processed instances and ensure that the current window stores the latest instances. Because ensemble algorithms have the advantage of modularity and can quickly adapt to changes, ensemble-based methods are the most common methods for handling concept drift.

Although much work has been done on concept drift [5–7], the class imbalance problem [11] (i.e., negative class instances are more extensive than other classes) further increases the difficulty of addressing concept drift [12]. Class imbalance commonly exists in the real world. Examples include cancer diagnosis, financial fraud detection, and geological disaster prediction. For binary classification, the class that has more instances is called the majority class (negative class), and the other is the minority class (positive class). For example, in the online fraud identification of automobile insurance, fraudulent customers accounted for only 1% of the total customers in 100 000 instances. Finding

a way to identify only 1% of fraudulent instances correctly can significantly reduce economic loss.

Several popular methods for dealing with the class imbalance issue [13–18] can be broken down into main groups: data-level techniques, cost-sensitive learning, and ensemble methods. Cost-sensitive learning methods aim to minimize the total cost. Some researchers argue that the cost-sensitive strategy is the most effective and frequent technique for dealing with class imbalance [11].

How to tailor the cost-sensitive learning strategy and adapt it to a nonstationary environment to enhance the capability of dealing with class imbalance is meaningful work. In practice, constructing classifiers under evolving data streams existing class imbalance is not a trivial task. It should address the following subproblems: (1) How can concept drift be handled? (2) How can class imbalance be managed?

A novel cost-sensitive learning scheme, named Cost-Sensitive based Data Stream (CSDS), is devised to tackle the combined issue to address these challenges. The contributions are threefold:

(1) A novel cost-sensitive variant of the ReliefF algorithm, named Cost-Sensitive based on ReliefF (CS-ReliefF), is proposed. The CS-ReliefF considers cost information in feature weighting to address the class imbalance issue at the data level.

(2) A dynamic cost-sensitive weighting mechanism is developed in the classification stage, incorporating cost value into the learning to alleviate the class imbalance at the algorithm level.

(3) The performance of our algorithm was implemented on different kinds of class imbalance data stream benchmarks. The results demonstrated that CSDS achieves the best overall performance in G-mean, running time, and concept drifts adaption.

2. Related Work

2.1. Class Imbalance Learning for Static Data. Researchers have done several works on class imbalance classification on static datasets [11]. The research work is mainly divided into three categories: data preprocessing techniques, cost-sensitive learning methods, and ensemble-based methods [13].

Data preprocessing techniques are mainly to alleviate the influence of class imbalance employing changing the original data distribution. Undersampling and oversampling are two common data preprocessing technologies. The undersampling method balances the classes by deleting the majority of instances, resulting in information loss [14]. The oversampling technique balances the data by duplicating a minority of instances. However, due to the uncertainty in the synthesis of new instances, it may weaken the classifier's performance [15]. SMOTE [16] is the most famous random oversampling algorithm, synthesizing new minority instances near the original minority instances. However, it often results in overfitting.

Cost-sensitive learning solutions [17] assign different costs to different classes, which seek to minimize the total cost. Suppose that the majority class is misclassified as a minority class. In that case, a lower misclassification cost is assigned, and when a minority class is misjudged as a majority class, a higher misclassification cost is assigned. In this way, we could balance the class distribution of the data. Most of these methods extend traditional machine learning methods to make them cost-sensitive. For example, literature [18] introduced cost-sensitive strategies into the SVM algorithm to minimize cost-sensitive hinge losses. AdaCost algorithm proposed in literature [19] reduces the weight of misclassified instances by introducing a cost-sensitive weight function into the AdaBoost. Sun et al. presented a series of algorithms based on cost-sensitive learning [20].

Bagging and boosting are two commonly used strategies in ensemble algorithms. Representative bagging-based ensemble algorithms used to deal with class imbalance include OverBagging [21], UnderBagging [22], UnderOverBagging [23], and DES-MI [24]. Data preprocessing techniques are often used in boost-based algorithms. Representative methods include SMOTEBoost [25] and RUSBoost [26]. Some ensemble methods combine both bagging and boosting strategies, including EasyEnsemble and BalanceCascade [27].

2.2. Data Streams Learning under Concept Drift and Class Imbalance. Although many efforts have been made focusing on class imbalance or concept drift separately [28–32], the combination of the two issues in data stream classification has not yet drawn enough attention.

Gao et al. proposed a general framework, called Sample and Ensemble (SE), for addressing class imbalance issues under streaming scenarios [28]. The SE divides the continuously arriving block into two groups: majority class instances and minority class instances. And then, SE collects the minority instances of the previous blocks and removes some of the majority class instances from the current chunk. Chen and He [29] introduced a novel ensemble solution, called Recursive Ensemble Approach (REA), for tackling class imbalance issues under a nonstationary environment. REA utilized the K-NN algorithm to measure the similarity between the minority class instances of the previous block and the minority class instances of the current block and chose the previous minority instances to balance the classes of the current block. Polikar et al. [30] presented an algorithm based on the Learn^{++} framework [31] to deal with class imbalance under a data stream environment named Learn^{++}.NIE. Recently, Mirza et al. introduced an online version of Extreme Learning Machine to solve the class imbalance issue [32].

In [33], a novel neural networks framework based on a cost-sensitive strategy was devised for handling the class imbalance issue. Li et al. introduce an ensemble algorithm using a multiwindow strategy to handle class imbalance issues [34]. More specially, three windows are designed in the algorithm: the current data block, the latest minority instances, and the pool of base classifiers. Lu et al. extended and improved the classic dynamic weighted majority (DWM) to effectively deal with the imbalance issue and

named Dynamic Weighted Majority for Imbalance Learning (DWMIL) [35]. Moreover, DWMIL used an underbagging strategy during data preprocessing to handle class imbalance. However, it has the drawback of overfitting. Zyblewski et al. proposed a dynamic classifier ensemble selection for imbalanced drifted data streams [36]. Most recently, Cano and Krawczyk proposed an algorithm called Kappa Update Ensemble (KUE) [37], which utilized the Kappa statistic for dynamically updating weights of base classifiers.

Simultaneously, some common problems exist in imbalanced data stream classification methods: these algorithms can deal with a specific type of concept drift. Besides, class imbalance often exists in the data stream together with concept drift. Most algorithms only focus on one problem and do not fully consider two issues simultaneously.

3. Our Method

3.1. Cost-Sensitive Based Data Stream Algorithm. A novel ensemble framework based on cost-sensitive feature selection is introduced to handle this study's joint issue. As shown in Figure 1, the proposed algorithm primarily consists of four steps:

Step 1: Data preprocessing: a cost-sensitive feature selection based on the ReliefF algorithm, named cost-sensitive ReliefF (CS-ReliefF), is devised. CS-ReliefF incorporates the cost information into feature selection, which selects a subset of features helpful in identifying the minority class. Hence, the feature set is more meaningful for effective prediction and has the effect of dimension reduction.

Step 2: Change detection: our algorithm employs concept detection to capture the changes explicitly, and when concept drift is detected, a new member classifier is built on the latest data.

Step 3: Classification module: a novel weighting scheme, that is, the weight of the base classifier, is updated based on accuracy and the total cost of misclassification on the latest data.

Step 4: Prediction: the weighted majority voting rule is used for predicting unknown instances.

3.2. Cost-Sensitive ReliefF Algorithm. A novel cost-based feature selection, named Cost-Sensitive ReliefF algorithm (CS-ReliefF), is proposed in this section. We adopt the ReliefF algorithm [38] mainly because it is simple, fast, and effective. More specially, we tailed the famous feature select algorithm ReliefF into a cost-sensitive learning model, which takes advantage of cost information into account during feature selection.

The main idea of the ReliefF algorithm is to weigh features according to their classification contribution. Specifically, the ReliefF randomly selects an instance x_i with class value y, finds its k nearest neighbors from the same class and different classes, and is denoted by H_j and $M_j(y)$, respectively. It updates the weights of all features based on their ability to distinguish neighboring instances.

Let x_i and x_j denote two instances, and their classes are y_i and y_j. The function diff (f, x_i, x_j) is defined as the difference between the value of feature f for two instances x_i and x_j, and it can be calculated according to

$$\operatorname{diff}\left(f, x_i, x_j\right) = \begin{cases} 0, & y_i = y_j, \\ 1, & y_i \neq y_j. \end{cases} \tag{1}$$

If the class is numerical, *i*.e., $y \in R$

$$\operatorname{diff}\left(f, x_i, x_j\right) = \left|\frac{y_i - y_j}{\max_f - \min_f}\right|, \tag{2}$$

where $\max_f$ and $\min_f$ represent the maximum and minimum values of f, respectively, and the diff (f, x_i, x_j) reflects the discrimination between x_i and x_j on f.

Let W_f denote the influence of feature f, where $W_f \in [-1, 1]$. ReliefF initializes the weights of all features to zero firstly. Then, the RelifF randomly selects an instance x_i and searches its k nearest neighbors. The ReliefF updates the weight of each feature according to

$$W_f = W_f - \frac{\sum_{j=1}^{k} \operatorname{diff}\left(f, x_i, H_j\right)}{r \cdot k} + \frac{\sum_{y \neq y_i} \left[\left(P(y)/1 - P(y_i)\right) \sum_{j=1}^{k} \operatorname{diff}\left(f, x_i, M_j(y)\right)\right]}{r \cdot k}, \tag{3}$$

where $P(y)$ is the prior probability of class y estimated from the training set, and r is a user-defined parameter indicating the number of iterations.

Unlike ReliefF, the proposed CS-ReliefF algorithm updates W_f considering cost information according to equation (4). In this way, the CS-ReliefF algorithm tends to select features with low costs.

$$W_f = W_f - \frac{\sum_{j=1}^{k} \operatorname{diff}\left(f, x_i, H_j\right)}{r \cdot k} + \frac{\sum_{y \neq y_i} \left[\left(P(y)/1 - P(y_i)\right) \sum_{j=1}^{k} \operatorname{diff}\left(f, x_i, M_j(y)\right)\right]}{r \cdot k} - \lambda \frac{\operatorname{Cost}_f}{r \cdot k}, \tag{4}$$

where $Cost_f$ is the test cost of f, and λ is the influence factor specified by the user.

$Cost_f$ is generated by a normal distribution, and the cost function is defined as follows:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left((x-\mu)^2/\sigma^2\right)}, \tag{5}$$

where μ and σ^2 are the mean and variance. To avoid the randomness of one sampling, the above process needs to be iterated r times. In our algorithm, the parameter p% is

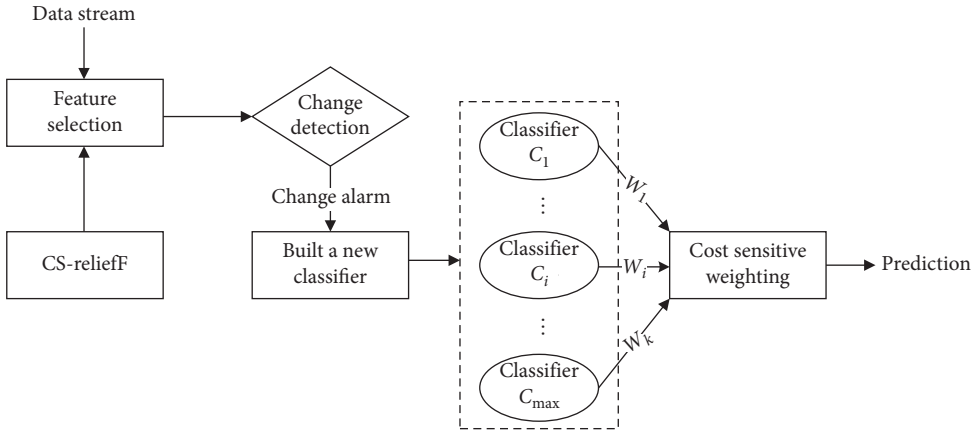

Figure 1: The framework of CSDS.

selected to adapt to the dynamically changing feature space. We adopt p as 75 in the following experiments. The pseudocode is shown in Algorithm 1.

3.3. Cost-Sensitive Ensemble Learning Method. Let $E = \{C_1, C_2, \ldots, C_k\}$ represent an ensemble with k base classifiers. The CSDS uses a sliding window technique to divide data stream S into data blocks B_i, $i = 1, 2, \ldots, n$, where $|B_i| = |B_j| = d$, $i \neq j$. More specially, whenever a new instance at time j is observed and inserted into the window, the $j - |W|$ is discarded. When a new block B_i arrives or a change occurs, we evaluate the effect of features according to equation (4) and use this to select an effective feature subset $F' \subseteq F$ (refer to Section 3.2). The CSDS uses B_i to build a new classifier C' and weights it according to the following equation:

$$W(C') = \frac{1}{\mathrm{MSE}_r + \alpha}. \quad (6)$$

The CSDS weights each base classifier $C_i \in E$ according to the following equation:

$$W(C_i) = \frac{1}{\mathrm{Cost}_i + \mathrm{MSE}_r + \mathrm{MSE}_i + \varepsilon}, \quad i = 1, 2, \ldots, k, \quad (7)$$

where MSE_r represents a randomly predicting classifier's performance of, and is employed as, the baseline for predicting the current distribution; MSE_i represents the mean square error of C_i on B_i at time t, respectively; $Cost_i$ is the total misclassification cost of C_i on B_i. When the ensemble is full, the worst classifier is removed, and the newly learned classifier is added to the ensemble. Our method adopts the weighted voting rule to make the final prediction. Moreover, the proposed algorithm utilizes the Drift Detection Method (DDM) [39], a change detection schema by detecting the classification model's error rate, as the change detector. In fact, it could be any change detection algorithm. The pseudocode of the cost-sensitive-based data stream algorithm is shown as follows (Algorithm 2).

4. Experiments and Analysis

The experiments were carried out on the Scikit-Multiflow framework [40], which is a python platform based on popular open-source frameworks including scikit-learn and MOA [41]. It includes data stream generators, classification algorithms, and evaluation methods.

4.1. Data Benchmarks. In the following experiments, we employed eight data stream benchmarks, including synthetic and real-world streams. The stream generators generated the synthetic streams in the Scikit-Multiflow framework. We adopted the ConceptDriftStream generator to simulate concept drift and used the ImbalancedStream generator to set the class imbalance rate ($^{\#}$majority instances: $^{\#}$minority instances). The description of the streams is shown in Table 1.

The Hyperplane that simulates a d-dimensional hyperplane is the most popular synthetic data to simulate gradual concept drift. A gradual drift stream with 1 m instances was generated in our experiments, and its imbalance rate was set to 5.

The SEA dataset is the most commonly used dataset representing sudden drift scenarios in data stream mining. We use the data stream generator to generate a data set of a sudden change in concept recurrence. The data set has a total of 1 m instances, which reappear every 250 K instances. Each instance is described by three attributes, which are used to represent one of the four concepts.

The LED dataset contains data used to predict the seven-segment LED display. We chose the 24 attributes version of the LED. We generate a mixed drifts stream containing 1 m instances, including sudden and gradual concept drifts.

The Rotating spiral is a dataset with the class imbalance and gradual concept drift. It is used to describe four types of spirals. The rotating spiral data stream contains 1 m instances, and the imbalance rate is 19.

The Spam dataset is a representative imbalanced real-world dataset, which collects e-mail messages from the Spam

Input: D: data in sliding window; F: feature set; k: number of neighbors; r: number of iterations
Output: Feature weight vector W

(1) Begin
(2) **for** all $f \in F$ do
(3) $W_f = 0$;
(4) **end for**
(5) **for** $i = 1$ to r do
(6) random select $\boldsymbol{x} \in D$;
(7) sampling $\boldsymbol{x_j} \in D$, if $y_j = y$ then add $\boldsymbol{x_j}$ to H_i, otherwise add to M_j, until $|H_i| = |M_j| = k$;
(8) **end for**
(9) **for** all $f \in F$ **do**
(10) Update W_f according to equation (5);
(11) **endfor**
(12) **endfor**
(13) Select the top p% of the features;
(14) **end.**

Algorithm 1: Cost-sensitive ReliefF feature select algorithm.

Input: data stream S, the maximum number of based classifiers max
Output: ensemble E with weighted classifiers $\{C_1, \ldots\ldots, C_{\max}\}$

(1) $E \longleftarrow \phi$;
(2) for each $x_i \in S$ do
(3) $W \longleftarrow W \cup \{\boldsymbol{x_i}\}$;
(4) apply CS-ReliefF on B_i;
(5) **if** $|W| \geq d$ or change is detected **then**
(6) learn a new classifier C' on data after feature selection;
(7) weight C' according to equation (6);
(8) **for** $C_i \in E$ **do**
(9) calculate the MSE_{ij} and MSE_r;
(10) update W (C') according to equation (7);
(11) **end for**
(12) **if** $|E| < \max$ **then**
(13) add C' into E;
(14) **else**
(15) replace the worst classifier with C';
(16) **end if**
(17) **for** $C_j \in E \backslash \{C'\}$ **do**
(18) train all $C_j \in E$ on B_i;
(19) **end for**
(20) end for
(21) **end.**

Algorithm 2: Cost-sensitive based data stream algorithm.

Assassin Collection. It has 9324 instances with 500 attributes. There are two classes (legitimate and spam). The imbalance rate of spam is 4. We simulate the spam into a stream with gradual drift by changing the features of spam over time.

The Sensor dataset has 2219803 instances, and five attributes describe each instance. The data is the information of 54 sensors of Intel Berkeley Research Lab in two months. Since attributes such as brightness and temperature change over time, the stream may contain concept drift.

The Electricity dataset is one of the most widely used real-world datasets. It was collected from the Electricity Market in Australia, containing 45312 instances, each described by seven attributes. The purpose of this dataset is to predict whether the price of electricity will increase (up) or decrease (down) with changes in market demand and supply. The classes are approximately balanced.

The Airlines stream contains 539383 instances, and eight attributes describe each instance. The class of Airlines is a delay, which indicates whether the flight is delayed.

4.2. Evaluation Metrics for Class Imbalance Learning. The G-mean is the geometric means of the recall of abnormal classes and that of normal classes, often used to measure the classifier's ability to handle unbalanced data [11]. It is often

applied in data streams with class imbalance to reduce the bias of the overall accuracy. For binary class classification, the G-mean is as follows [42, 43].

$$G - \text{mean} = \sqrt{\frac{\text{TP}}{\text{TP} + \text{FN}} \times \frac{\text{TN}}{\text{TN} + \text{FP}}}. \quad (8)$$

G-mean can be extended to multiclass cases. Assuming that there are m classes, G-mean is still the geometric average of various correct rates, defined as

$$G - \text{mean} = \left(\prod_{i=1}^{m} G - \text{mean}_i \right)^{1/m}, \quad (9)$$

where G-mean$_i$ is G-mean of ith class.

4.3. Experimental Results. We verified the effectiveness of CSDS using cost-sensitive strategies in evolving data stream scenarios involving different types of drifts and class imbalance. CSDS was compared with the following methods:

(i) VFDT: VFDT is an incremental decision tree classification based on the Hoeffding inequality theory, which can guarantee the constructed decision tree's accuracy with a certain probability.

(ii) AUE2: AUE2 is a block-based ensemble that combines the accuracy-based weighting mechanism with the incremental learning of the Hoeffding tree and aims to deal with various types of drift.

(iii) KUE: KUE is a dynamic weighting ensemble that utilized the Kappa statistic to update base classifiers' weight dynamically.

The evaluation can generate an incremental learning curve of metrics changing over time. For a fair comparison, the maximum number of the compared ensemble algorithms was set to 15. We chose the Hoeffding tree as the base classifier. The performance can be evaluated concerning G-mean and time (the averaged results of 10 runs), as shown in Tables 2 and 3.

4.3.1. G-Mean Analysis. Concerning the G-mean, CSDS achieved the best average ranking, as shown in Table 2. CSDS gained the best performance over four data streams: HyperPlane, SEA, Spam, and Electricity. We find that CSDS classification performs better in class imbalance data streams environment. VFDT obtains the worst performance. This is because it cannot solve the class imbalance challenge, but also incapable of dealing with concept drift. CSDS classification employs the cost-sensitive learning strategy during the data preprocessing and classification stages. CSDS uses the CS-ReliefF algorithm to incorporate cost information into feature selection to select a subset of features helpful in identifying minority classes. Therefore, the feature set is more meaningful for effective prediction and has the effect of dimension reduction. Simultaneously, a dynamic cost-sensitive weighting strategy is developed to reduce class imbalance at the algorithm level.

Table 1: Description of the datasets.

Data stream	# Inst	# Attrs	# Cls	IR	Drift
HyperPlane	1 m	10	2	5	Gradual
SEA	1 m	3	4	10	Sudden, recurring
LED	1 m	24	10	3	Mixted
Rotating spiral	1 m	40	3	19	None
Spam	9324	500	7	4	Unkown
Sensor	2219803	5	54	54	Unkown
Electricity	45312	10	10	1	Unkown
Airlines	539 383	7	2	2	Unknown

Table 2: Average G-mean of comparing algorithms.

	VFDT	AUE2	KUE	CSDS
HyperPlane	0.93 ± 0.04 (4)	0.95 ± 0.06 (3)	0.96 ± 0.05 (2)	$\mathbf{0.97 \pm 0.01}$ **(1)**
SEA	0.72 ± 0.10 (4)	0.83 ± 0.12 (3)	0.86 ± 0.22 (2)	$\mathbf{0.88} \pm 0.01$ **(1)**
LED	0.82 ± 0.02 (4)	$\mathbf{0.87} \pm 0.07$ **(1)**	0.86 ± 0.06 (2)	0.84 ± 0.02 (3)
Rotating spiral	0.73 ± 0.10 (4)	$\mathbf{0.79} \pm 0.05$ **(1)**	0.76 ± 0.02 (3)	0.78 ± 0.01 (2)
Spam	0.62 ± 0.08 (4)	0.66 ± 0.02 (3)	0.80 ± 0.04 (2)	$\mathbf{0.86} \pm 0.02$ **(1)**
Sensor	0.76 ± 0.11 (3)	0.74 ± 0.07 (4)	$\mathbf{0.86} \pm 0.06$ **(1)**	0.83 ± 0.04 (2)
Electricity	0.60 ± 0.05 (4)	0.64 ± 0.02 (2)	0.62 ± 0.02 (3)	$\mathbf{0.73} \pm 0.02$ **(1)**
Airlines	0.76 ± 0.07 (3)	0.74 ± 0.08 (4)	$\mathbf{0.83} \pm 0.07$ **(1)**	0.81 ± 0.03 (2)
Average rank	3.75	2.63	2.00	1.63

Table 3: Times of comparing algorithms (seconds).

	VFDT	AUE2	KUE	CSDS
HyperPlane	25.32 (2)	**15.33 (1)**	35.68 (4)	18.10 (3)
SEA	**8.32 (1)**	12.23 (3)	22.07 (4)	10.60 (2)
LED	**29.15 (1)**	36.03 (3)	43.29 (4)	34.65 (2)
Rotating spiral	**9.68 (1)**	16.01 (3)	21.35 (4)	11.04 (2)
Spam	**4.31 (1)**	15.47(4)	14.56(3)	7.45 (2)
Sensor	80.46 (2)	90.24 (3)	104.41 (4)	**66.79 (1)**
Electricity	48.79 (2)	69.41 (3)	77.46 (4)	**31.63 (1)**
Airlines	59.45 (3)	32.23 (1)	60.20 (4)	37.80 (2)
Average rank	1.63	2.63	2.88	1.88

4.3.2. Time Analysis. In terms of running time, VFDT performs best, followed by our algorithm, and KUE performed the worst. As shown in Table 3, we observe that the ensemble algorithms have certain advantages in G-mean, but it does not perform well in running time. Although the single decision tree classifier VFDT has apparent advantages in time, it performs the worst in overall performance. Overall, in most cases, CSDS can achieve a good compromise between G-mean and running time and adapt to drifts faster than other ensemble methods. Our algorithm benefits from the modular characteristics of ensemble learning, which can better deal with recurring gradual drifts. Meanwhile, the change

detection mechanism is embedded in our algorithm to capture sudden drift in time.

Next, we adopt graphical plots to visualize how the algorithm is affected by different kinds of change. X-axis and y-axis denote the number of processed instances and G-mean of the algorithms, respectively.

The SEA dataset is used to simulate scenes with sudden changes and to detect the ability to address sudden concept drift. In this scenario, the curves of the G-mean with the increment of processed increases are shown in Figure 2. The performance of the VFDT is the worst, followed by AUE2 and KUE, and CSDS is the best. Moreover, around the 50Kth instance, the G-mean values of all algorithms undergo rapid fluctuations except CSDS. This may benefit from the concept drift detection mechanism, which can promptly capture a sudden drift, thereby establishing a new classifier to adapt such drift.

The LED dataset simulates mixed concept drift, that is, scenes with gradual and sudden concept drift. It is intended to verify the algorithm's responsiveness to mixed drift. Specifically, the dataset is a stream containing two gradual drifts and one abrupt concept drift. When processing half of the stream, the target concept suddenly shifts from one concept to another. As shown in Figure 3, we have observed that all algorithms maintain a higher G-mean value when the data is relatively stable. When a sudden change occurs in the data stream, the performance of all algorithms except CSDS drops sharply. This may be because CSDS can capture different kinds of changes timely and reconstruct a new model to recover from concept drift quickly. Besides, CSDS provides the best overall performance utilizing cost-sensitive learning strategies in feature selection and classification.

Changes in real-world streaming scenarios are often complex and changeable, so simulating the real-world environment can better verify classifiers' performance. Figure 4 illustrates the G-mean curves on the Spam data stream change over time. All the curves show varying degrees of fluctuation, which implies that there may exist a drift in the stream. Since VFDT and AUE2 cannot deal with imbalances, the dataset simulates a real-world nonstationary scenario that includes class imbalance. They perform poorly in a scenario where class imbalance and concept drift coexist.

In contrast, the curves of CSDS and KUE are not significantly affected by concept drift, since CSDS is oriented to the data stream's changing characteristics to respond to these problems quickly and in real-time. Additionally, cost-sensitive learning strategies are adopted at the data level (feature selection) and algorithm level (classifier weighting) to effectively deal with imbalances.

Finally, we adopted the nonparametric Friedman test with a significance level $\alpha = 0.05$ to perform statistical tests on all competitive algorithms [44]. The statistical test results show that the null hypothesis is rejected. That is, there is no significant difference between the algorithms. After that, we further employ the Nemenyi test [45] to verify whether the performance of our method is statistically different from other algorithms. The result is shown in Figure 5. The results show that CSDS is significantly better than VFDT.

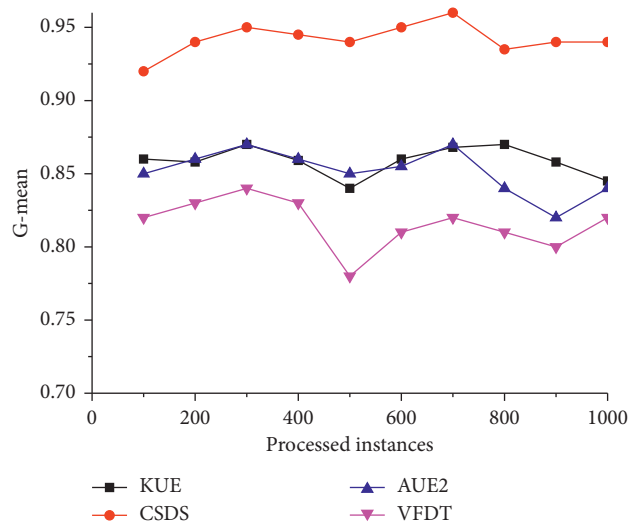

Figure 2: Comparison of G-mean on the SEA dataset.

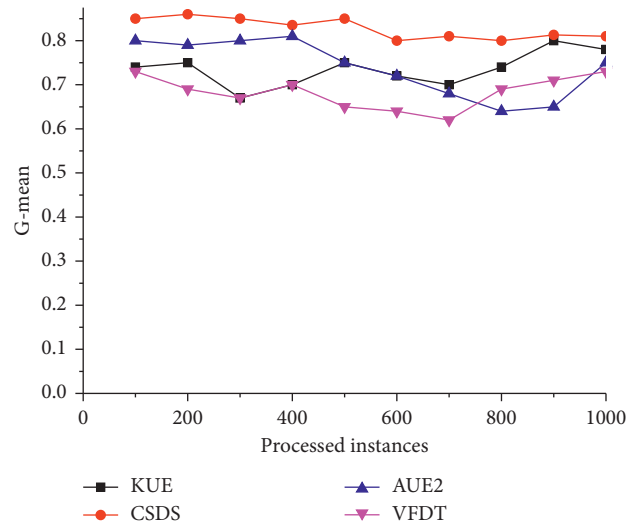

Figure 3: Comparison of G-mean on rotating spiral dataset.

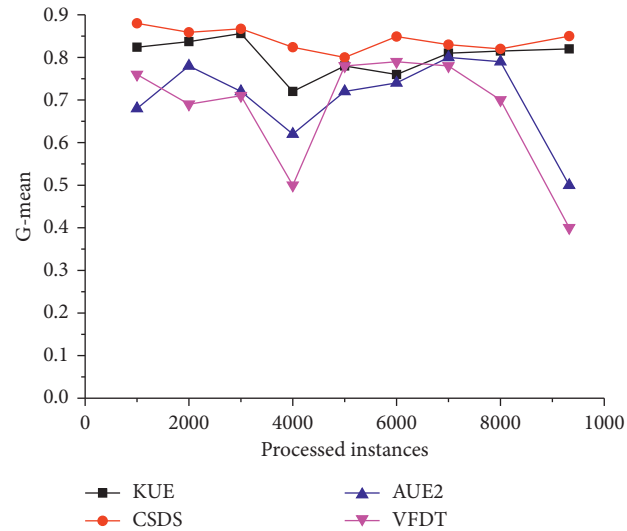

Figure 4: Comparison of G-mean on spam dataset.

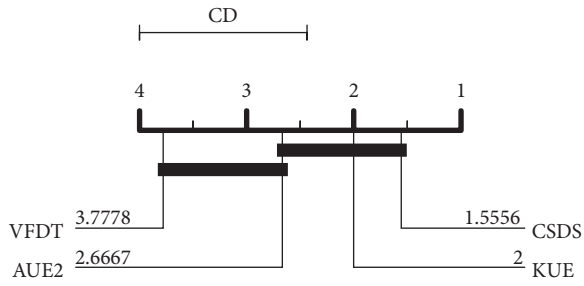

Figure 5: Nemenyi test for G-mean of all algorithms.

5. Conclusion

This study provides novel insight into how to utilize a cost-sensitive learning strategy to deal with class imbalance under dynamic streaming scenarios. An ensemble schema based on a cost-sensitive strategy is devised to handle the combination of the two issues. Firstly, a cost-sensitive version of the ReliefF algorithm that incorporates cost information during the data preprocessing is proposed to solve the class imbalance issue at the data level. Secondly, a cost-sensitive classifier weighting scheme utilizing cost information is devised in the ensemble stage. Moreover, a change detection module is embedded in the ensemble to capture drift in real-time. Finally, extensive experimental results show that our method is superior to the competitive algorithms and gains the best trade-off between performance and resources, especially for nonstationary data streams with imbalanced class environments. Furthermore, the results verified its statistical significance with a nonparametric Friedman test.

This study focuses on the topic of single-label data stream classification. Multilabel data streams are common in many real applications. In the future, we plan to extend cost-sensitive learning into the multilabel stream scenario.

Data Availability

The datasets and experimental codes can be downloaded from the websites: http://moa.cms.waikato.ac.nz/ and https://github.com/scikit-multiflow/scikit-multiflow.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported by National Natural Science Foundation of China (Nos. 62062004, 61702550, and 31900710), Teacher Education Curriculum Reform Projects of Henan Province (No. 2020-JSJYYB-034), the Innovation Team Support Plan of University Science and Technology of Henan Province (No. 19IRTSTHN014), Key Scientific Research Projects of Henan Province (No. 20B520030), Young backbone teachers program in Colleges and Universities of Henan Province, and Nanhu Scholars Program for Young Scholars of XYNU.

References

[1] C. C. Aggarwal, *Data Streams: Models and Algorithms*, Springer-Verlag, Berlin, Germany, 2007.

[2] J. Gama, *Knowledge Discovery from Data Streams*, Chapman & Hall/CRC, London, UK, 2010.

[3] I. Zliobaite, M. Pechenizkiy, and J. Gama, "An overview of concept drift applications," *Big Data Analysis: New Algorithms for a New Society, Studies in Big Data*, vol. 16, pp. 91–114, 2016.

[4] M. G. De Francisci, A. Bifet, L. Khan, J. Gama, and W. Fan, "IoT big data stream mining," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016)*, pp. 2119-2120, ACM Press, San Francisco, CA, USA, August 2016.

[5] H. M. Gomes, J. Read, A. Bifet, J. P. Barddal, and J. Gama, "Machine learning for streaming data," *ACM SIGKDD Explorations Newsletter*, vol. 21, no. 2, pp. 6–22, 2019.

[6] A. Tsymbal, *The Problem of Concept Drift: Definitions and Related Work*, Trinity College Dublin, Dublin, Ireland, 2004.

[7] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, pp. 231–238, 2014.

[8] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: a survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[9] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 964–994, 2016.

[10] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, and K. Ghédira, "Discussion and review on evolving data streams and concept drift adapting," *Evolving Systems*, vol. 9, no. 1, pp. 1–23, 2018.

[11] H. Haibo He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[12] H. He and Y. Ma, *Imbalanced Learning. Foundations, Algorithms, and Applications*, John Wiley & Sons, Inc, Hoboken, NJ, USA, 2013.

[13] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, "Learning from imbalanced data streams," *Learning from Imbalanced Data Sets*, Springer, Cham, Germany, 2018.

[14] X. Y. Xu-Ying Liu, J. Jianxin Wu, and Z. H. Zhi-Hua Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 539–550, 2009.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[16] H. He, B. Yang, E. A. Garcia, and S. Li, "ADASYN: adaptive synthetic sampling approach for imbalanced learning," in *Proeedings f the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, June 2008.

[17] A. Krishnamurthy, A. Agarwal, T. K. Huang, H. Daume III, and J. Langford, "Active learning for cost-sensitive classification," *Journal of Machine Learning Research*, vol. 20, pp. 1–50, 2019.

[18] P. Cao, D. Zhao, and O. Zaiane, "An optimized cost-sensitive SVM for imbalanced data learning," in *Proceedings of the*

Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 280–292, Springer, Gold Coast, Australia, April 2013.

[19] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "AdaCost: misclassification cost-sensitive boosting," in *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 97–105, Morgan Kaufmann Publishers Inc., Bled, Slovenia, June 1999.

[20] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognition*, vol. 40, no. 12, pp. 3358–3378, 2007.

[21] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2012.

[22] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," in *Proceedings of the IEEE Symposium on Computational Intelligence & Data Mining*, pp. 324–331, IEEE, Nashville, TN, USA, March 2009.

[23] B. Sun, H. Chen, J. Wang, and H. Xie, "Evolutionary under-sampling based bagging ensemble method for imbalanced data classification," *Frontiers of Computer Science*, vol. 12, no. 2, pp. 331–350, 2018.

[24] S. García, Z.-L. Zhang, A. Altalhi, S. Alshomrani, and F. Herrera, "Dynamic ensemble selection for multi-class imbalanced datasets," *Information Sciences*, vol. 445-446, pp. 22–37, 2018.

[25] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "SMOTEBoost: improving prediction of the minority class in boosting," in *Proceeding of the 7the European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 107–119, Springer, Cavtat-Dubrovnik, Croatia, September 2003.

[26] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: a hybrid approach to alleviating class imbalance," *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, vol. 40, no. 1, pp. 185–197, 2010.

[27] X. Liu, J. Wu, and Z. Zhou, "Exploratory under-sampling for class-imbalance learning," in *Proceeding of the 6th IEEE International Conference on Data Mining (ICDM 2006)*, pp. 965–969, IEEE, Hong Kong, China, December 2006.

[28] J. Gao, W. Fan, J. W. Han, and P. S. Yu, "A general framework for mining concept-drifting data streams with skewed distributions," in *Proceedings of the 7th SIAM International Conference on Data Mining*, pp. 3–14, Minneapolis, MN, USA, April 2007.

[29] S. Chen and H. He, "Towards incremental learning of non-stationary imbalanced data stream: a multiple selectively recursive approach," *Evolving Systems*, vol. 2, pp. 35–50, 2011.

[30] R. Polikar, L. Robi, S. S. Udpa, and V. Honavar, "Learn++: an incremental learning algorithm for supervised neural networks," *IEEE Transactions on Systems, Man & Cybernetics: Part C—Applications & Reviews*, vol. 31, no. 4, pp. 497–508, 2001.

[31] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, pp. 2283–2301, 2013.

[32] B. Mirza, Z. Lin, and N. Liu, "Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift," *Neurocomputing*, vol. 149, pp. 316–329, 2015.

[33] A. Ghazikhani, R. Monsefi, and H. Sadoghi Yazdi, "Ensemble of online neural networks for non-stationary and imbalanced data streams," *Neurocomputing*, vol. 122, pp. 535–544, 2013.

[34] H. Li, Y. Wang, H. Wang, and B. Zhou, "Multi-window based ensemble learning for classification of imbalanced streaming data," *World Wide Web*, vol. 20, no. 6, pp. 1507–1525, 2017.

[35] Y. Lu, Y. Cheung, and Y. Y. Tang, "Dynamic weighted majority for incremental learning of imbalanced data streams with concept drift," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 2393–2399, AAAI Press, Melbourne, Australia, August 2017.

[36] P. Zyblewski, R. Sabourin, and M. Woniak, "Preprocessed dynamic classifier ensemble selection for highly imbalanced drifted data streams," *Information Fusion*, vol. 66, pp. 138–154, 2020.

[37] A. Cano and B. Krawczyk, "Kappa updated ensemble for drifting data stream mining," *Machine Learning*, vol. 109, no. 1, pp. 175–218, 2020.

[38] I. Kononenko, "Estimating attributes: analysis and extensions of RELIEF," in *Proceedings of European Conference on Machine Learning*, F. Bergadano and L. D. Raedt, Eds., pp. 171–182, Springer, Catania, Italy, April 1994.

[39] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proceedings the 17th Brazilian Symposium on Artificial Intelligence (SBIA 2004, LNCS 3171)*, pp. 286–295, Springer-Verlag, Sao Luis, Brazil, September 2004.

[40] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, "Scikit-multiflow: a multi-output streaming framework," *Journal of Machine Learning Research*, vol. 19, pp. 1–5, 2018.

[41] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: massive online analysis," *Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

[42] R. P. Espíndola and N. F. F. Ebecken, "On extending *F*-measure and *G*-mean metrics to multi-class problems," *WIT Transactions on Information and Communication Technologies*, vol. 35, pp. 25–34, 2005.

[43] W. Shuo, L. L. Minku, and Y. Xin, "A systematic study of online class imbalance learning with concept drift," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 4802–4821, 2018.

[44] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[45] N. Settouti, M. E. A. Bechar, and M. A. Chikh, "Statistical comparisons of the top 10 algorithms in data mining for classi cation task," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 4, no. 1, pp. 46–51, 2016.