IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 8, 2020, accepted October 13, 2020, date of publication October 16, 2020, date of current version November 2, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3031603*

# Two-Stage Cost-Sensitive Learning for Data Streams With Concept Drift and Class Imbalance

**YANGE SUN[1,2], YI SUN[3], AND HONGHUA DAI[4,5]**

[1]School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, China
[2]School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China
[3]Zhengzhou Information Science and Technology Institute, Zhengzhou 450004, China
[4]Institute of Intelligent Systems and Innovation, Deakin University, Melbourne, VIC 3125, Australia
[5]Cooperative Innovation Center of Internet Healthcare, Zhengzhou University, Zhengzhou 450000, China

Corresponding author: Yange Sun (ygsun1982@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 62062004, Grant 31900710, and Grant 61702550; in part by the Innovation Team Support Plan of University Science and Technology of Henan Province under Grant 19IRTSTHN014; in part by the Teacher Education Curriculum Reform Projects of Henan Province under Grant 2020-JSJYYB-034; in part by the Key Scientific Research Projects of Henan Province under Grant 20B520030; and in part by the Nanhu Scholars Program for Young Scholars of XYNU.

**ABSTRACT** Most methods for classifying data streams operate under the hypothesis that the distribution of classes is balanced. Unfortunately, the phenomenon of class imbalance widely exists in many real-world applications. In addition, the underlying concept of data stream may change in a certain way over time, and attacks increase the difficulty of data stream mining. Motivated by this challenge, a Two-Stage Cost-Sensitive (TSCS) classification is proposed for addressing the class imbalance issue in non-stationary data streams. We propose a novel two-stage cost-sensitive framework for data stream classification by utilizing cost information in both feature selection stage and classification stage. Moreover, a window adaptation and drift detection mechanism, which guarantees that an ensemble can adapt promptly to concept drift, is embedded in our method. Our algorithm is compared with competitive algorithms on different kinds of datasets. The result demonstrates that TSCS obtains significant improvement in terms of class imbalance data stream metrics.

**INDEX TERMS** Data streams, classification, class imbalance, concept drift, cost-sensitive, ensemble learning.

## I. INTRODUCTION

Extracting knowledge from data stream environment has gained growing attention owing to its wide applications, such as credit card fraud detection, spam filtering, intrusion detection and data analysis in Internet of Things networks [1]–[5]. Concept drift [6]–[9], i.e., the distribution of data stream evolving over time, is the crucial characteristic of data streams, which deteriorates the classification performance due to data distribution evolving. For example, the characteristics of spam often change with users' preferences, and weather prediction models affected by atmospheric dynamics. Therefore, classifications should have the capacity of detecting the stability-plasticity dilemma caused by concept drift [10], [11].

Although much work has been put forward to focus on concept drift issue [5]–[7]. In practice, class imbalance [12] is the mining case that the instances of one class (normally of interesting) is much smaller than that of other classes which poses the well-known challenge in machine learning. Learning from data stream with class imbalance and concept drift becomes a more challenging task. The phenomenon of class imbalance presences widely in many applications, including severe weather forecasting, rare event monitoring, text classification, medical diagnosis, networks intrusion detection and fault identification. In fact, the learning with the simultaneous occurrence of the two issues in data stream classification is largely unexplored.

Class imbalance issue has extensively studied in static learning scenarios [13]–[18]. These methods can be organized into three main groups: a) data preprocessing oriented approaches, b) cost-sensitive oriented approaches, and c) ensemble oriented approaches. Cost-sensitive oriented approaches assign different misclassification cost values for each class in classification. Research has shown that cost-based strategies are an effective method. Moreover, research

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wei.

indicates that cost-sensitive learning strategy can be adopted to solve the class imbalance issue naturally [19].

Both cost-sensitive learning and online classification have been studied extensively in data mining community, respectively. Unfortunately, there are relatively few studies consider the application of cost-sensitive learning strategy in data stream scenarios due to its nature of dynamic changes [19].

Finding a way to perform cost-sensitive learning in non-stationary environment and to adopt effective learning strategies to deal with class imbalance to pursuit better performance of classification with concept drifting is the main aim of this research. In addition, the task of learning from evolving data streams with class imbalance is to tail the cost-sensitive learning strategy into the evolving online scenarios. The most challenge of our work is to devise an effective cost-sensitive online learning paradigm that can effectively extract knowledge from data stream with class imbalance and concept drift.

This study is on the topic of adaptive ensemble, which is considered as the most popular technique for handling concept drifts. Differs from existing ensemble architectures, we tail the cost-sensitive learning strategy into the evolving data stream classification scenarios, and a novel cost-sensitive online ensemble, named Two-Stage Cost-Sensitive (TSCS) scheme, is devised seeking towards both issues simultaneously. The main contribution of this article is to generalize cost-sensitive learning algorithms to their online versions. For each newly arrived data block, TSCS first uses a cost-sensitive learning mechanism to preprocess the feature set, that is, effectively use cost information to select a feature subset that helps to improve the model's performance on minority instances; then, define the feature subset in the feature space, use TSCS to train the classifier on the current data block; finally, TSCS evaluates the accuracy and misclassification cost of the existing classifier on the current data block, and weights the classifier based on the evaluation result. Compared with existing algorithms, TSCS can achieve better performance on artificially synthesized and real-world class imbalance concept drift data streams.

To summarize, there are four key original contributions of our algorithm.

(1) In the feature selection stage, a Cost-Sensitive Principal Component Analysis (CSPCA) by continuously selecting the optimal set of features according to the cost information is employed for performing feature selection on imbalanced data streams. The exploration of feature subspace can improve the generalization ability of the algorithm, and can better adapt to various concept drifts.

(2) In the classification stage, a cost-sensitive weighting schema is developed, which introduces cost information into the learning framework to effectively manage the improvement of the learning performance.

(3) An adaptive window change detection mechanism is also employed in the framework to react promptly to different kinds of changes.

(4) An extensive experimental study was carried out on a variety of data stream benchmarks. The results indicated that the proposed method obtained better performance than the competitive methods, particularly under dynamic data streams with class imbalance environments.

The structure of the paper is organized as follows. In Sections II, we briefly retrospect some closely related work. Next, we describe our algorithm in detail in Section III. Section IV demonstrates the experiments results. Finally, a conclusion is drawn.

## II. RELATED WORK

This section reviews the recent work of the major contributions that are closely related to our research presented in this article which covers approaches in handling concept drift and in dealing with the class imbalance.

### A. HANDLING CONCEPT DRIFT IN DATA STREAMS

Concept drift is a hot research topic in the field of data mining community and a lot of classification algorithms have been developed. These methods have been reviewed by Tsymbal [6], Gama [7], Dilter [8], Khamassi [10] and Lu [11] *et al*, providing valuable insight into addressing the concept drift.

In the literature [6], the surveyed concept drift adaptation methods are divided into three main categories: instance selection methods, weight-based approaches and ensemble-based methods. The method based on instance selection is the most widely used technique for processing concept drift. Specifically, it refers to selecting a part of the data to be processed, that is, selecting the most relevant instance of the current concept for learning, so as to improve the efficiency of algorithm learning. Such methods are often implemented through sliding window technology (Sliding Window). The method based on instance weighting increases its influence in the new model by assigning larger weights to the data that is most relevant to the current data. Many methods for setting weights have been proposed. The more commonly used ones are based on the age of the instance (such as the time when the data arrives, etc.), or based on the degree of relevance of the data in the processing of the new concept. The ensemble learning refers to the use of multiple base classifiers to form a set of classifiers by means of model averaging, and then use voting or weighted voting to combine them to predict unknown data.

Dilter *et al*. divided the methods into two main categories [8]: active approaches and passive approaches. The active approaches are mainly by adding a concept drift detection mechanism to the classifier to actively discover detection drift. The single classification model uses the concept drift detection mechanism to actively detect the concept drift in the data stream. Once the concept drift is detected, the current model will be adjusted to delete outdated concepts and adapt to the new concepts in time. However, for the slow concept drift with gentle changes, it may be difficult to detect once it appears, causing the old concepts not be deleted in time,

thus affecting the performance of model. It is notable that the passive adaptation method does not actively detect concept drift, nor consider the occurrence of concept drift in the data block, and is not sensitive enough to capture the drift. When sudden conceptual drift occurs, it is difficult to deal with it in time.

In this article, we group methods for handling concept drift in data streams into three categories, namely single classifiers, drift detectors and ensemble-based methods. Most single-classifier methods improve and extend traditional classification algorithms to adapt to the changing environment of data streams. In addition, single-classifier methods generally provide implicit drift processing mechanisms (such as concept drift detection, sliding window, and instance weighting). William *et al*. proposed a method called Paired Classifiers including two classifiers for complementary consideration: using the more accurate classifier as the candidate for prediction [20]. Much earlier than Paired Classifiers methods, Zhuang and Dai introduced an inexact approach for Dual Imbalance Text Classification [21]. Hulten *et al*. extended the classic VFDT algorithm [22], and proposed an algorithm named CVFDT [23] that can handle concept drift. CVFDT maintains a window for storing the most recent data, and uses a candidate sub-tree to train on the window data. If its performance exceeds the original. For the sub-tree corresponding to the tree, the candidate sub-tree is used instead of the original sub-tree. Gama *et al*. also improved VFDT and proposed the VFDTc algorithm [24]. The VFDTc algorithm can process continuous attribute data and detect concept drift by comparing the data distribution between two windows. Most of the work is based on decision tree algorithms, including Hoeffding Adaptive Tree (HAT) [25], Adaptive-Size Hoeffding Tree (ASHT) [26], and Ultra Fast Forest of Trees (UFFT) [27].

Concept drift detection algorithm, also known as concept drift detector, refers to an algorithm that detects input data information according to data distribution. It monitors the changes in the data stream distribution, and once it finds a concept drift, it will warn the classifier and take corresponding actions to adjust it to the new data distribution.

In [28], Gama *et al*. proposed a method called Drift Detection Method (DDM) by monitoring the error-rate of the algorithm. Although DDM can effectively capture sudden concept drifts, it cannot detect the gradual drift in time. To solve this problem, Baena-Garcia *et al*. improved the DDM algorithm and presented the Early Drift Detection Method (EDDM) method [29], which detects the standard deviation between the error rates of two connections. Bifet and Gavalda [30] proposed the ADWIN method to adapt to concept drift, which cuts each window into two sub-windows, representing the new data and the old data respectively, and determines the best split according to the rate of change between the sub-windows, and then the new data use the training model to replace the old model. In [31], Philipp *et al.* proposed a parallelized version of ADWIN algorithm. Gomes *et al.* [32] proposed an online version of random forest classifier, named Adaptive Random Forest classification (ARF), which uses a concept detector to decide when to replace trees in the forest to deal with concept drift.

The third solution is a common strategy for addressing concept drift issue. Ensemble-based approaches [33], [34] have proved to be an efficient and powerful technique of handling concept drift due to their flexible structure. Generally, constructing ensemble can be divided into three main steps: Training a set of base classifiers (or component classifiers), selecting a subset of base classifiers, and making the final prediction. It is considered to be an effective way to update knowledge without changing the model structure. On one hand, ensemble learning can learn new knowledge very efficiently. It only needs to add new classifier members to the ensemble model. On the other hand, integrated learning provides a natural mechanism for forgetting irrelevant knowledge.

The above work only focuses on the issue of concept drift. The following subsections review some major work in learning from data with class imbalance.

### B. LEARNING FROM CLASS IMBALANCE DATA

Ensemble approaches have the capability of easily adapting to changing data stream scenarios due to their modular structure. Thus becoming one of the most popular methods used for dealing with drift. The ways of processing data streams can be roughly divided into two types: data block-based processing and single instance online processing. Ensemble-based methods for data streams classification can be categorized into the following three types: block-based ensembles, online ensembles, and hybrid ensembles [35].

(1) Specifically, in block-based ensemble, data stream is divided into fixed-size blocks. Block-based ensemble adapts to change by periodically updating its components and replacing the weakest members with new ones. Streaming Ensemble Algorithm (SEA) [36] is a block-based ensemble, which maintains a fixed number of base classifiers based on blocks and employs a heuristic classifier replacement strategy. Wang *et al.* present a method named Accuracy Weighted Ensemble (AWE) [37], which is the best-known representative block-based ensemble. AWE maintains the top-$k$ classifiers learnt from sequential blocks of instances and dynamically weights each classifier according to the most recent block. In [38], an algorithm named Accuracy Updated Ensemble (AUE1) was proposed. AUE1 incrementally trains and updates component classifiers after each block. Its improved version AUE2 [39], by periodically weighting ensemble members, could obtain better response to gradual drifts. At the meanwhile, it improves performance on abruptly changing streams environment. In [40], the algorithm dynamically weights each classifier according to time-adjusted accuracy on change of distribution. The performance of block-based ensembles is greatly affected by the size of block. Another disadvantage is that their delay in response to sudden change in time because true labels can be entirely available after each full data block.

(2) For online ensembles, incremental classifiers are maintained and updated as soon as a new instance arrives. Kolter and Maloof proposed the most cite online-based ensemble, named Dynamic weighted majority (DWM) [41]. When there is an error in algorithm classification, DWM dynamically adds a new expert, when there is an error in the expert, it reduces the weight of the expert, when the expert performance is poor, DWM removes an expert, and incrementally trains exist component experts through online learning. Similar to DWM, Gomes and Enembreck presented a dynamic ensemble classification called Streaming Ensemble Algorithm (SEA2) [42], which is represented by an undirected graph. More specifically, SEA2 is an ensemble algorithm represents as a network in which connections are created between two classifiers if they have similar predictions. The weights of edges in the graph represent the similarity of the two classifiers. Recently, Lu *et al.* proposed a Dynamic Weighted Majority for Imbalance Learning (DWMIL) based on the DWM framework [43]. DWMIL processes the data incrementally and dynamically weighting the base classifiers to ensure timely processing of concept drifts. Moreover, DWMIL uses under-bagging technology before the base classifier training, to balance the class distribution. However, it has the shortcoming of over-fitting.

Online-based ensembles update the component classifiers after each new arriving instance. Hence, they have the capacity of adapting to sudden changes rapidly. Unfortunately, online ensembles have the disadvantage of higher computational costs compared with block-based ensembles.

(3) The hybrid ensembles combine the advantages of weighting mechanism of block-based and online processing. It continuously updates base classifiers online according to the coming instances. Nishida *et al*. proposed an adaptive ensemble algorithm, named Adaptive Classifier Ensemble (ACE) [44]. The ACE algorithm is based on the data block period weighting mechanism and online learning method. It detects mutation concept drift by monitoring the error rate of classifiers, and builds a base classifier on the data block to cope the gradual concept drift. When there are large data blocks, training the classifier will consume too much time. Brzezinski *et al*. proposed a hybrid ensemble named Online Accuracy Updated Ensemble (OAUE) [45]. However, it has a potentially drawback of unable to adapt to sudden changes due to the fixed window size. Recently, Gama *et al*. introduced a dynamic weighting ensemble algorithm, named Kappa Update Ensemble (KUE) [46], which adopts Kappa metric to guide the member classifiers' selection and weighting.

In this subsection, we surveyed recent advances in strategies for learning from class imbalance data in general. In the next subsection, we will provide a review on approaches for learning from class imbalance data streams in particular.

### C. LEARNING FROM CLASS IMBALANCE DATA STREAM

Basically, both class imbalance data and data involving concept drift are special types of low quality data [47]. Dai [48] pointed out three strategies for learning from low quality data: preprocess data before learning; improve data quality while learning and introduce learning algorithms that tolerate low quality data. All these three strategies are applicable in guiding the creation of new learning approaches.

The main recent developed approaches for learning from class imbalance can be classified into three categories [12], [13]: 1) data preprocessing oriented approaches; 2) cost-sensitive oriented approaches, and 3) ensemble oriented approaches.

1) Data preprocessing oriented approaches are fit into the first category in fixing the imbalance before learning. This type of approaches aims to reduce the imbalance of class distribution by changing the distribution of data. Specifically, during data preprocessing, the distribution of data is balanced by adding minority instances (oversampling) or reducing majority instances (undersampling). The Synthetic Minority Oversampling TEchnique (SMOTE) [13] proposed by Chawla *et al*. is the most famous preprocessing oriented algorithm adopting the random oversampling technology. More specifically, SMOTE generates new instances by performing random linear interpolation between a small number of instances and their similar neighbors, which increases the possibility of overlapping between classes. The experimental results show that compared with other standard approaches, classification accuracy of the minority class is improved. However, the duplication or synthetic creation of the minority class instances often leads to over-fitting.
2) Cost-sensitive learning methods [14] fit into the second type of learning strategy fixing imbalance data issue during the learning. This type of methods assigns distinct misclassification costs to different classes and obtains the optimal decision boundary by minimizing the total misclassification cost. Most of the cost-sensitive methods are improved to traditional machine learning methods. For example, applying the cost-sensitive strategy to the SVM algorithm [15], the cost-sensitive Hinge loss function is minimized. AdaCost algorithm [16] introduces cost- sensitive strategy into the weight function of AdaBoost to reduce the weight of misclassified instances. Subsequently, Sun *et al*. proposed a family of AdaBoost algorithms based on cost-sensitive strategy: AdaCl, AdaC2, and AdaC3 [17].
3) Ensemble methods belong to the third type strategy which is capable of tolerating class imbalance. These methods are mainly classified into three categories [18]: bagging-based approaches, boosting-based approaches, and hybrid ensembles. Bagging-based methods mostly combine bagging technology, such as OverBagging, UnderBagging, UnderOverBagging, and DES-MI. Data preprocessing is introduced to the boosting algorithm, and the weights of misclassified instances are adjusted by updating the

distribution weights. Some prominent boosting-based methods include SMOTEBoost and RUSBoost. SMOTEBoost [49] introduces SMOTE to create synthetic instance technology into the AdaBoost iteration. Similar to SMOTEBoost, RUSBoost [50] randomly randomizes the majority of instances in each iteration. The hybrid ensemble method combines both bagging and boosting strategy. Some prominent methods include EasyEnsemble and BalanceCascade [21].

Data streams also exhibit class imbalance problem. In fact, addressing the two issues simultaneously is a nontrivial task due to the need to adhere the following requirements: 1) detecting changes as soon as possible; 2) adapting to changes quickly; and 3) recovering to the accuracy level before changes.

In [51], Gao *et al*. proposed an ensemble framework using oversampling technology, named Sample and Ensemble (SE). In such framework, incoming data block is divided into two parts: one represents the positive class instances ($P$) and another represents the negative class instances ($Q$). To take all these instances into account, we choose some positive instances from $P$ and obtain a subset of $Q$ then merge them together. Then we use all these instances to learn a new classifier to obtain a better performance. Chen and He [52] proposed a novel algorithm, named SERA (SElectively Recursive Approach), which used a similar measure to select the previous minority instances. Similar to SERA, Gao *et al*. proposed an algorithm that selects the ''best'' $n$ minority class according to Mahalanobis distance. Unfortunately, it is not strictly incremental, as it requires to access previous data. Mirza and Lin [53] introduced a method based on Extreme Learning Machine (ELM), which incorporates sampling and cost-sensitive weighting technique to solve class imbalance and use window technology to deal with concept drift. It is the first meta-cognitive framework to solve class imbalance and concept drift problem. Ditzler and Polikar [54] presented an extension of Learn$^{++}$.NSE, named Learn$^{++}$.NIE. It employs subensemble and class-independent error weighting mechanism with a penalty constraint strategies to address the two issues.

Ghazikhani *et al*. introduced cost-sensitive learning to neural networks and proposed an online neural network ensemble algorithm [55] for dealing with the class imbalance issue. Recently, Li *et al.* [56] presented a novel ensemble algorithm based on multi-window technology to address the issue. Lu *et al.* [43] proposed an algorithm, called Dynamic Weighted Majority for Imbalance Learning (DWMIL), which dynamically weights the performance of base classifier on each newly arriving block to ensure timely processing of concept drift. Moreover, DWMIL uses underbagging technology before the base classifier training; that is, during each bagging iteration, undersampling technology is used to ensure that the training instance class distribution is balanced. However, it has the shortcoming of overfitting.

Most recently, Zhao *et al.* [57] propose to introduce a novel cost-sensitive online classification framework, named Adaptive regularized Cost-sensitive Online Gradient descent algorithms (ACOG), which employs the adaptive regularization during the classification. Wang and Pineau [58] proposed a cost-sensitive home algorithm using an online gradient descent algorithm to solve online optimization tasks by maximizing weights or minimizing weighted classification error costs; and theoretically analyzed cost sensitivity. Recently, Wang *et al.* [59] convert a series of batch-based cost-sensitive ensembles into online versions, including new online extensions of UnderOverBagging, SMOTEBagging, AdaC2, CSB2, RUSBoost, and SMOTEBoost.

In [58], Khanchi *et al*. proposed an active learning algorithm named StreamGP, which adapt over time as genetic programming (GP) individuals improve. Korycki *et al.* [60] present an online framework called Active Learning Strategy (MD-OAL) for imbalanced data streams with partially labeled.

Most of the above algorithms are based on techniques for static data that are used to reduce the imbalance in data. These methods have the following disadvantages: SE only builds a classifier for the current data block, and cannot save the concepts that have been learned, and most of the data block use real-time update mechanisms that cause excessive time consumption. The base classifier in IMDWE increases linearly, resulting in a sharp increase in time consumption. Learn$^{++}$.CDS and Learn$^{++}$.NIE lead to excessive time consumption; and SERA requires constant access to previous data, so it is not a strictly incremental algorithm. While, StreamGP and MD-OAL are specially designed for partially labeled data stream scenarios.

## III. OUR METHOD

This section firstly introduces a change detector based on adaptive window strategy to deal with concept drift, and then our cost-sensitive principal component analysis method is described in detail. After that our proposed two-stage cost-sensitive ensemble method is presented.

### A. CHANGE DETECTOR BASED ON ADAPTIVE WINDOW

In our method, a novel change detection algorithm, which utilizes a two-window change detection schema to monitor the change of distribution, is presented. In order to quantitatively measure whether the data in the sliding window has concept drift, it utilizes Kullback-Leibler divergence as the metric to compare the distance between the windows. The Kullback-Leibler divergence of two distributions $p(x)$ and $q(x)$ is defined as Equation (1).

$$KL(p\|q) = \sum_{x\in X} p(x)\log\frac{p(x)}{q(x)} \tag{1}$$

where $x$ is the space of the events.

More formally, let $W_1 = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ and $W_2 = \{\boldsymbol{x}_{n+1}, \ldots, \boldsymbol{x}_{2n}\}$ represent the reference window and the current window. Our change detection can be expressed as hypothesis testing uses the following methods to detect whether the data

sequence has concept drift:

$$\begin{cases} H_0 \ KL(W_1, W_2) \leq \varepsilon \\ H_1 \ KL(W_1, W_2) > \varepsilon \end{cases}$$

where $KL(W_1||W_2)$ is a metric that measures the dissimilarity of $W_1$ and $W_2$, $\varepsilon$ is a threshold. If the Kullback-Leibler divergence of the data in the new window and the data in the historical window is greater than the threshold, it indicates that a concept drift has occurred.

Adaptive window change detection can detect sudden concept drift, gradual drift and recurring concept. A concept drift is detected when the value of distance function exceeds the threshold. A recurring concept is recognized when the measure is zero. $\varepsilon$ is a threshold calculated according to Bernstein inequality, refer to our previous work [61]. The change detector alarms a signal when a change is detected. Then the outdate data is discarded. If the stream is stationary, the windows slide step by step. The pseudo-code of the adaptive window detector is specified in the Algorithm 1.

**Algorithm 1** Change Detection based on Adaptive Windowing

**Input:** $S$ is a data stream, $n$ is maximum size of windows;
**Output:** alarm signal
**Procedure**:
01: Set $t$ to 0;
02: Set $W_1$ to$\{x_{t+1}, \ldots, x_{t+n}\}$;
03: Set $W_2$ to $\{x_{t+n+1}, \ldots, x_{t+2n}\}$;
04: **while** not at the end of $S$ **do**
05: **if**$KL(W_1, W_2) > \varepsilon$ **then**
06: Alarm a drift signal at $t$;
07: Discard the outdate instances and goto step 02;
08: **else if** $KL(W_1, W_2) == 0$ **then**
09: Alarm a recurring concept signal;
10: **end if**
11: **else** $W_2$ is slide by 1 step;
12: **end if**
13: **end while**
14: **end.**

### B. COST-SENSITIVE FEATURE SELECTION BASED ON PRINCIPAL COMPONENT ANALYSIS

In our algorithm, we apply a cost-sensitive learning strategy in the feature selection. Principal component analysis (PCA) [62] is one of the most popular feature selection techniques, which uses orthogonal transformations to gain a low-dimensional representation of data called the principal components. In order to deal with class imbalance data, we tailor the traditional PCA algorithm into cost-sensitive mode, named Cost-Sensitive Principal Component Analysis (CSPCA). More specially, we apply cost-sensitive strategy in feature selection stage, by incorporating cost value into the process of feature selection.

The majority and minority classes in training instances use different cost ratios:

$$C_i = \begin{cases} C_i^- = \dfrac{1-\alpha}{N^-} & If\, y_i = -1 \\ C_i^+ = \dfrac{1+\alpha}{N^+} & If\, y_i = +1 \end{cases} \tag{2}$$

where $i = 1, \ldots, n$ instances; $N^-$ and $N^+$ are the total number of negative and positive instances, respectively. $\alpha \in [0, 1]$ is a parameter.

Consider the data matrix $X \in \mathbb{R}^{m \times n}$, where the $n$ represents the total number of instances, and the $m$ are the number of features. $\mathrm{w}_1$ is the first principal component of $p$ dimensions and $p \ll m$. $\mathrm{w}_1$ is calculated according to Equation (3):

$$\mathrm{w}_1 = \arg\max \sum_{i,j} \left(\mathbf{X}_{i,j} \cdot \mathbf{w}_{1j}\right)^2 \tag{3}$$

where $i$ and $j$ are the rows and columns of $\mathbf{X}$ respectively.

Geometrically, the first step of PCA is centered data by subtracting the average of the data from all points. However, in class imbalance case, covariance matrix usually represents the variance of most of the class instances, and the maximum variance direction of the data may be largely captured from most spaces. For this reason, we introduced the cost-sensitive learning into the PCA algorithms to solve imbalance problems. For binary classification, it is assumed that the negative and positive instances are discounted by imbalance cost ratios $C^-$ and $C^+$ respectively. Thus, the weighted $r$ th principal component is:

$$\begin{aligned} \mathrm{w}_1 = \arg\max & \sum_{i:y_i=1,j} \left(C_i^- \mathbf{X}_{i,j} \cdot \mathrm{w}_{1\,j}\right)^2 \\ & + \sum_{i:y_i=+1,j} \left(C_i^+ \mathbf{X}_{i,j} \cdot \mathrm{W}_{1j}\right)^2 \end{aligned} \tag{4}$$

where $j$ is the index of the dimension, $C_i^-$ and $C_i^+$ are calculated according to Equation (2).

Using different cost ratios for negative and positive classes can reduce the dominant role of negative instances in selecting features. CSPAC selects a relatively small subset of relevant features from the original set of features. For this reason, the CSPCA can address the class imbalance at the feature selection level without changing the distribution of data or modifying the algorithm.

### C. TWO-STAGE COST-SENSITIVE ENSEMBLE LEARNING

In this section, a novel two-stage cost-sensitive ensemble framework is put forward for effectively handling the joint issue of class imbalance and concept drift. TSCS provides a combination of online and block-based approaches, both continuously updating base classifiers and replacing them with new ones when necessary. More specifically, TSCS relearn a classifier based on the instances in latest window. Another strategy is to use hybrid ensemble learning scheme, in which the base classifiers are weighted their performance and cost of the latest block, and incremental learning the classifiers on the latest block. Hence, it is a hybrid method that offers both

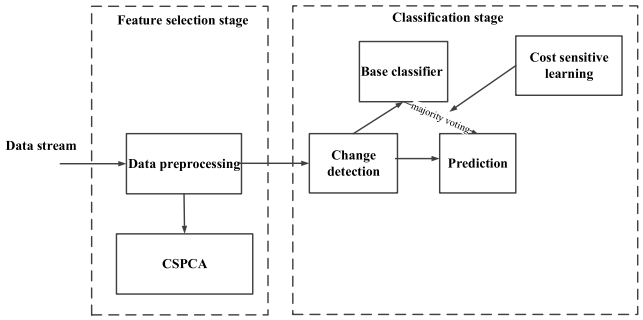


**FIGURE 1.** The framework of TSCS algorithm.

implicit (change detection mechanism) and explicit mechanisms (ensemble weighting mechanism) to deal with concept drifts.

The basic framework of the TSCS algorithm is shown in Figure 1, which mainly involves two stages:

1) PCA-base feature selection stage: It employs a cost-sensitive learning mechanism in the feature selection process to filter out feature sets that are more meaningful for effective prediction and classification of minority samples, and at the same time. It also has the effect of reducing data dimensions.

2) Ensemble classification stage: A cost-sensitive weighting schema is designed, that is, based on both the accuracy and the total cost of misclassification of the base classifiers on the latest data block to update its weight, and then the weighted voting schema is employed for prediction.

Assume there are

—$d$ Stream data window $W_j$; contains the current data $(1 \le j \le d)$;

—$k$ base classifiers $h_i = f_i(W_j)$ $(1 \le i \le k)$;

—$m$ instances $I_t$ $(1 \le t \le m)$;

To implement our two stage cost-sensitive ensemble learning, we need to introduce the weight $weight_{ij}$ and the cost $ct_{ij}$. The more important a base classifier outcome is, the higher weight is assigned. Similarly the higher cost of a base classifier, the less important the base classifier would be.

Specifically, a data stream can be described as $S = \{s_1, s_2, \ldots, s_t, \ldots\}$, where $s_t = (\boldsymbol{x}_t, y_t)$ represents the instance at time $t$. $E = \{h_1, h_2, \ldots, h_k\}$ represents the ensemble containing $k$ weighted base classifiers. First, TSCS continuously uses CSPCA algorithm to select effective feature subset, and learns a base classifier $h_i$ base on the new feature space of the latest window $W_j$, i. e., $h_i = f_i(W_j)$. And then, for each incoming instance $\boldsymbol{x}_t$, instead of evaluating base classifiers every $d$ instances, each classifier $h_i \in E$ is weighted based on the data in the latest window according to Equation (5).

$$\begin{aligned} weight_{ij} &= \frac{1}{ct_{ij} + MSE_{ij} - MSE_r + \Delta} \\ MSE_r &= \sum_y p(y)(1 - p(y))^2 \\ MSE_{ij} &= \frac{1}{|W_j|} \sum_{(x,y) \in W_j} (1 - f_y^i(\mathbf{x}))^2 \end{aligned} \tag{5}$$

where $MSE_{ij}$ is the prediction error of base classifier $h_i$ on the recent window $W_j$, while $MSE_r$ represents the mean square error of a randomly predicting classifier and is employed as the baseline for predicting the current distribution. Additionally, $\Delta$ is a very small positive value. $f_y^i(x)$ represents the probability of using $h_i$ to classify $x$ as $y$.

$ct_{ij}$ is the total cost of the misclassification of classifier $h_i$ on current window $W_j$, which can be computed using Equation (6):

$$ct_{ij} = \frac{1}{|W_j|} \sum_{\{x,y\} \in W_j} \sum_{y'} \cos t(y, y') f_{y'}^i(\mathbf{x}) \tag{6}$$

where $\cos t(y, y')$ represents the cost of instance $x$ of class $y$ being classified as $y'$.

The intuition underlying of Equation (5) is that the weight of the classifier $h_i$ is inversely proportional to the error and the cost value, that is, the larger the classification error rate and cost of the classifier on the data of latest window, it indicates that the classifier is not suitable for the current data distribution, and its weight in the ensemble classification should be weakened.

After the generation of base classifiers, a new classifier $h'$ is trained then added into the ensemble when drift is detected. When the number of base classifiers reaches the specified maximum value, the worst classifier is replaced by the new classifier. The final prediction of ensemble is based on the weighted majority voting rule according to Equation (7).

$$E(\mathbf{x}_t) = \sum_{i=1}^{k} weight_i f_i(\mathbf{x}_t) \tag{7}$$

where $\sum_{i=1}^{k} weight_i = 1, 0 < weight_i < 1$.

To assume $\sum_{i=1}^{k} weight_i = 1$ and $0 < weight_i < 1$ hold true. We normalize the weights of base classifiers in the following way:

Assume $k$ individual weights obtained via formula (5) is $< w_1, w_2, \ldots, w_k >$ (Here we fix the window on current one so we omit the subscript $j$). We let

$$\overline{w} = \sum_{i=1}^{k} w_i \tag{8}$$

and we let the weights we used in Equation (7) as follows

$$\text{weight}_i = \frac{w_i}{\bar{w}} \tag{9}$$

When the algorithm detect a drift or the instances in the long-term buffer exceeds the specified max value (line 4), a classifier that represents the new concept is built according to the instances in $B$ (line 5), weighted (line 6), and added to the ensemble (line 8). The final prediction is based on the weighted majority voting rule. The pseudo-code of TSCS is given in Algorithm 2.

**Algorithm 2** Two-Stage Cost-Sensitive Ensemble

**input:** $D$: change detector, $S$: data stream, $k$: max number of classifiers, $d$: maximum size of windows
**output:** $E$: ensemble with $k$ base classifiers
**Procedure:**
01: $E \leftarrow \varphi$
02: **for** each current $W_j$ in $S$
03: apply CSPCA algorithm;
04: **if** $|W_j| > d$**or** detected a change **then**
05: train a new candidate classifier $h'$ based on new feature subset of $W_j$;
06: weight new classifier $h$';
07: weight all classifiers $h_i$ in $E$ according to Equation (5);
train all classifiers $h_i$ in $E$ incrementally;
08: **if** $E$ is not full **then** add the classifier $h'$ to ensemble;
09: **else** the worst base classifier is replaced by $h'$;
10: reset $D$;
11: reset current window;
12: **end if**
13: **end for**
14:**end.**

## IV. EXPERIMENTAL STUDY

The experiments are carried out with Massive Online Analysis (MOA) [63] software package. MOA is the most famous open source framework for data stream mining and it provide an environment for implementing the state-of-the-art algorithms of data stream mining.

### A. DATA BENCHMARKS

A thorough experimental study, comparing TSCS to 14 state-of-the-art algorithms over 6 synthetic and 8 real-world data stream benchmarks. The synthetic data streams were simulated by the MOA Generators.

The imbalance rate (IR) represents the ratio between the majority class instances and the minority class instances. Two types of class imbalance benchmarks are selected in the experiment: (a) static class imbalance benchmarks; (b) dynamic class imbalance datasets, whose IR values change dynamically over time. The detail of the datasets is described in Table 1.

#### 1) SYNTHETIC DATASETS

We first generate data streams through the generators in MOA, and then use the ConceptDriftStream generator to simulate concept drifts. Finally, we set the imbalance rate of the ImbalancedStream generator to simulate data streams with class imbalance. For multi-class classification, the first class is set as the major class, the remainder classes are the minority classes.

The Agrawal data stream is used to generate one of ten specified functions, including a total of 9 attributes. We use the MOA Agrawal generator to produce a stream with 1000, 000 instances.

**TABLE 1. Characteristic of the datasets.**

| Stream | #Inst | #Attrs | #Clas | IR | Drift |
|---|---|---|---|---|---|
| Agrawal | 1M | 9 | 2 | 1 | gradual |
| STAGGER | 1M | 9 | 2 | 1 | sudden |
| HyperPlane | 1M | 10 | 2 | 1~20 | gradual |
| LED | 1M | 24 | 10 | 3 | mixed |
| Rotating spiral | 1M | 40 | 3 | 19 | gradual |
| SEA | 1M | 3 | 4 | 10 | sudden, recurring |
| Spam | 581K | 53 | 7 | 4 | unkown |
| German | 1K | 24 | 2 | 7 | unknown |
| Poker | 1M | 10 | 10 | 13 | unkown |
| Electricity | 1M | 10 | 2 | 5 | unkown |
| Airlines | 539K | 7 | 2 | 2 | unknown |
| IJCNN1 | 49, 990 | 22 | 2 | 5 | unknown |
| Sensor | 2M | 5 | 54 | 54 | unkown |
| Weather | 18, 159 | 8 | 2 | 10 | unkown |

The STAGGER data set was proposed by Schlimmer *et al*. It is a widely used dataset of simulated concept drift.. In the experiment, we utilized the STAGGERGenerator in MOA to generate a dataset having 1, 000, 000 instances. It contains 3 concepts. The concept 1 is set as the majority class. The imbalance ratio with the other two categories is 5.

Hyperplane is the most popular synthetic dataset in data stream classification experiments. It is a classic two class classification problem that simulates a $d$-dimensional hyperplane. We simulate a stream with gradual drift containing 1, 000, 000 instances.

The LED dataset is to predict the numbers on LED display. In the experiment, we produced 24 binary attributes version of the LED, of which 17 are irrelevant. Simulate concept changes by exchanging related attributes. We generate a LED with mixture drift. It contains 1, 000, 000 instances.

The Rotating Spiral stream is used to describe four types of spirals. It contains 1, 000, 000 instances, of which about 5% are positive instances. Therefore, the Rotating spiral is a dataset with class imbalance and gradual concept drifts.

The SEA stream has three attributes, only two of them are relevant. The dataset contains four concepts which represent a block of data. The dataset uses $f_1 + f_2 \leq O$ to classify the instances in the blocks. 9, 8, 7 and 9.5 are the most common used threshold. We generate a SEA dataset with sudden recurring concept drift, containing 1, 000, 000 instances. We use the RecurrentConceptDriftStream generator to simulate recurrent concept drifts.

#### 2) REAL-WORLD DATASETS

It is impossible for real-world data sets to know exactly when drift begins, what kind of drift exists, or even whether it does

exist. Therefore, it is more meaningful to verify the adaptability of algorithms. The static datasets can be downloaded at MOA website,[1] and then use the MOA generators to simulate them into streams.

Spam dataset represents information of a message, and is divided into two types: spam (only 20%) and legitimate messages, so it is a dataset with class imbalance. It contains 9, 324 instances, each instance have 500 attributes. The characteristics of the spam slowly evolving over time. And then simulate static dataset into stream by the generator in MOA.

The Electricity dataset is the most frequently used real-world data stream. It has 45, 312 instances, and each described by 7 attributes. The purpose of Electricity dataset is to predict the trend of electricity price changes in Australia.

The German credit dataset contains 1000 instances, and each instance is described by 24 attributes. The purpose of the dataset is to predict the tendency of loan default based on bank loan information and the occurrence of overdue loans of applied customers. It can be obtained from LIBSVM website.[2]

The Poker Hand dataset comes from UCI repository [64]. It indicates the issue of recognizing hand in a poker game. The Poker dataset has 1000, 000 instances. Each instance is described by 10 attributes, representing a group of five cards in hand. The class is described as ''Poker Hand''.

The purpose of Airlines dataset is to predict whether a flight will be delayed based on the given information of scheduled departure. The dataset contains 539, 383 instances, and each instance consists of three numeric attributes and five nominal attributes. The class of the Airlines is delay, indicating whether a flight is delayed.

IJCNN1 dataset comes from the IJCNN competition in 2001, and uses the pre-processed data of LIBSVM as experimental data. It has 49, 990 instances.

The Sensor dataset comes from 54 sensors information in the Intel Berkeley Research Laboratory, as shown in Figure 2. It consists of 2, 219, 803 instances and each instance with 5 attributes. Brightness and temperature change over time, leading to concept drifts.

The Nebraska Weather Prediction (Weather) dataset contains the daily weather measurements of the Offutt Air Force Base in Bellevue. It consists of 18, 159 instances. It is a dataset with diverse weather patterns and concept drifts.

### B. EVALUATION METRICS

Since the classification accuracy is not suitable for imbalanced data distribution, we adopt $G$-mean [67, 68] as alternate metrics for evaluating the performance of classifiers in class imbalance scenarios.

(1) Precision and Recall

Precision and recall can be calculated as:

$$precision = \frac{TP}{TP + FP} \tag{10}$$

[1]http://moa.cms.waikato.ac.nz/datasets/.

[2]http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

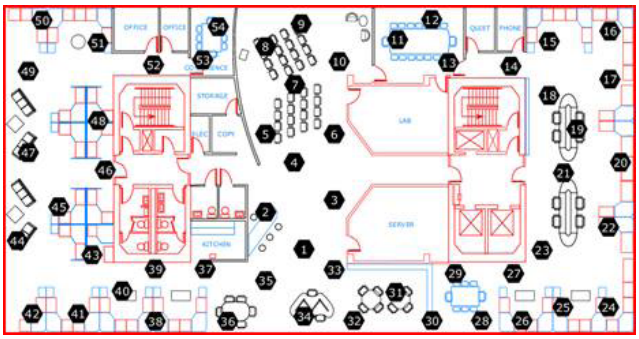

**FIGURE 2.** The sensor distribution map in the inter Berkeley Research Lab.

$$recall = \frac{TP}{TP + FN} \tag{11}$$

(2) $F_{-\text{measure}}$ $F_{-\text{measure}}$ is the harmonic mean of precision and recall, as shown in Equation (12).

$$F\text{ -measure} = \frac{2\ \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{12}$$

(3) $G$-mean

$G$-mean is the most commonly used evaluation measure in class imbalance environments. $G$-mean is the geometric mean of the recall of abnormal classes and normal classes. Formally, the $G$-mean can be calculated as follows [65].

$$G - mean = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \tag{13}$$

### C. EXPERIMENTAL SETUP

All algorithms are written in Java and executed under the MOA framework. The experiments were carried out on a 3.0GHz Pentium CPU with 32GB of RAM and Microsoft Windows 10.

In the experiments, we use the prequential evaluation strategy in MOA to evaluate the performance of all compared methods. Therefore, a classifier is evaluated before all instances are obtained. This evaluation method can generate an incremental learning curve, where the two-dimensional curve corresponding to the horizontal axis to the number of instances that have been classified, and the vertical axis are the dynamic changes of evaluation indicators.

In the study, all tested ensemble methods set $k$ to 15. The Hoefffding tree was selected as the basic classifier. This is mainly because the decision tree is probably the most widely used algorithm equipped with ensemble technology, and can effectively deal with concept drift. For the Hoeffding tree algorithm, we set grace period $n_{min}$ to 100, set tie-threshold $\tau$ to 0.05, and set split confidence $\delta$ to 0.01.

### D. RESULTS AND ANALYSIS

The proposed method was evaluated against ten state-of-the-art algorithms. The first five is non-stationary learning algorithms and the last five are designed for evolving environments with class imbalance.

**TABLE 2.** *G*-mean comparison of 10 algorithms.

| | ARF | AWE | Oza | Lev | AUE2 | KUE | OSB | NIE | ACOG | TSCS |
|---|---|---|---|---|---|---|---|---|---|---|
| Agrawal | **0.53 (1)** | 0.45 (7) | 0.47 (6) | 0.48 (5) | 0.43 (8) | 0.38 (10) | 0.49 (4) | 0.52 (3) | 0.41 (9) | 0.52 (2) |
| STAGGER | 0.74 (9) | 0.73 (10) | 0.76 (7) | 0.75 (8) | **0.89 (1)** | 0.86 (3) | 0.79 (6) | 0.83 (5) | 0.88 (2) | 0.85 (4) |
| HyperPlane | 0.83 (7) | 0.79 (10) | 0.85 (4) | 0.84 (5) | 0.82 (8) | **0.93 (1)** | 0.82 (8) | 0.89 (2) | 0.86 (3) | 0.84 (5) |
| LED | 0.63 (8) | 0.67 (3) | 0.67 (3) | 0.68 (2) | 0.63 (8) | 0.66 (6) | 0.61 (10) | 0.67 (3) | 0.66 (6) | **0.69 (1)** |
| Rotating spiral | 0.62 (10) | 0.66 (9) | 0.80 (6) | 0.84 (3) | 0.69 (8) | 0.81 (5) | **0.86 (1)** | 0.74 (7) | 0.85 (2) | 0.83 (4) |
| SEA | 0.76 (3) | 0.64 (6) | 0.65 (5) | 0.63 (8) | 0.66 (4) | 0.78 (2) | 0.60 (10) | 0.64 (6) | 0.62 (9) | **0.79 (1)** |
| Spam | 0.66 (8) | 0.64 (9) | 0.62 (10) | 0.73 (6) | 0.68 (7) | 0.76 (3) | 0.75 (4) | 0.74 (5) | **0.83 (1)** | 0.81 (2) |
| German | 0.76 (6) | 0.74 (8) | 0.83 (3) | 0.81 (4) | 0.76 (6) | **0.86 (1)** | 0.62 (10) | 0.66 (9) | 0.80 (5) | 0.85 (2) |
| Poker | 0.84 (4) | 0.65 (10) | 0.67 (8) | 0.68 (7) | 0.81 (6) | 0.83 (5) | 0.86 (3) | 0.66 (9) | 0.87 (2) | **0.89 (1)** |
| Electricity | 0.73 (8) | 0.83 (3) | 0.84 (2) | 0.80 (4) | 0.69 (10) | 0.73 (8) | **0.85 (1)** | 0.80 (4) | 0.79 (6) | 0.76 (7) |
| Airlines | 0.58 (10) | 0.63 (7) | 0.66 (3) | 0.67 (2) | 0.62 (9) | 0.63 (7) | 0.64 (5) | 0.65 (4) | **0.68 (1)** | 0.64 (5) |
| IJCNN1 | **0.79 (1)** | 0.72 (5) | 0.70 (7) | 0.69 (8) | 0.62 (10) | 0.76 (2) | 0.63 (9) | 0.72 (5) | 0.73 (4) | 0.74 (3) |
| Sensor | 0.84 (3) | 0.78 (6) | 0.76 (8) | 0.75 (9) | 0.83 (4) | 0.86 (2) | 0.71 (10) | 0.77 (7) | 0.82 (5) | **0.87 (1)** |
| Weather | 0.41 (8) | 0.40 (9) | 0.47 (4) | 0.42 (5) | 0.36 (10) | 0.42 (5) | 0.49 (3) | 0.42 (5) | 0.53 (2) | **0.55 (1)** |
| Average | 0.69 (6) | 0.67(10) | 0.70 (4) | 0.70 (4) | 0.68 (9) | 0.73 (3) | 0.69 (6) | 0.69 (6) | 0.74 (2) | **0.76 (1)** |

**TABLE 3.** $F_{\text{measure}}$ comparison of 10 algorithms.

| | ARF | AWE | Oza | Lev | AUE2 | KUE | OSB | NIE | ACOG | TSCS |
|---|---|---|---|---|---|---|---|---|---|---|
| Agrawal | 0.057 (9) | 0.099 (5) | 0.068 (8) | 0.039(10) | 0.086 (6) | **0.194 (1)** | 0.157 (2) | 0.137 (3) | 0.078 (7) | 0.135 (4) |
| STAGGER | **0.118 (1)** | 0.106 (4) | 0.080 (6) | 0.062 (7) | 0.045 (9) | 0.035(10) | 0.110 (3) | 0.056 (8) | 0.089 (5) | 0.112 (2) |
| HyperPlane | 0.451 (4) | 0.345 (5) | 0.221 (7) | 0.207(10) | 0.477 (2) | **0.479 (1)** | 0.454 (3) | 0.345 (5) | 0.221 (7) | 0.217 (9) |
| LED | 0.097 (8) | 0.187 (5) | 0.141 (6) | 0.189 (4) | 0.247 (2) | 0.235 (3) | 0.089 (9) | 0.129 (7) | **0.268 (1)** | 0.086(10) |
| Rotating spiral | 0.098 (5) | 0.036(10) | 0.069 (8) | 0.058 (9) | 0.156 (4) | 0.224 (2) | 0.098 (5) | 0.075 (7) | 0.210 (3) | **0.262 (1)** |
| SEA | 0.027 (9) | 0.110 (7) | 0.169 (4) | 0.029 (8) | 0.248 (2) | 0.216 (3) | 0.027 (9) | 0.113 (6) | 0.169 (4) | **0.273 (1)** |
| Spam | 0.039 (9) | 0.049 (6) | 0.078 (4) | 0.047 (7) | 0.057 (5) | 0.043 (8) | **0.239 (1)** | 0.033(10) | 0.178 (3) | 0.181 (2) |
| German | 0.120 (9) | 0.135 (6) | 0.136 (5) | 0.073(10) | 0.147 (3) | 0.169 (2) | 0.121 (8) | 0.127 (7) | 0.146 (4) | **0.186 (1)** |
| Poker | 0.046 (6) | 0.034(10) | 0.055 (4) | 0.045 (8) | 0.047 (5) | 0.059 (3) | 0.046 (6) | 0.037 (9) | **0.068 (1)** | 0.067 (2) |
| Electricity | **0.156 (2)** | 0.097 (3) | 0.078 (7) | 0.052(10) | 0.086 (6) | 0.094 (4) | **0.157 (1)** | 0.093 (5) | 0.078 (7) | 0.066 (9) |
| Airlines | 0.118 (9) | 0.156 (8) | 0.280 (3) | 0.262 (5) | 0.245 (6) | 0.236 (7) | 0.118 (9) | **0.356 (1)** | 0.281 (2) | 0.274 (4) |
| IJCNN1 | 0.450 (3) | 0.345 (6) | 0.220(10) | 0.232 (8) | 0.407 (4) | **0.489 (1)** | 0.451 (2) | 0.345 (6) | 0.221 (9) | 0.364 (5) |
| Sensor | 0.089(10) | 0.127 (9) | 0.140 (8) | 0.189 (5) | 0.237 (2) | 0.235 (3) | 0.149 (6) | 0.197 (4) | 0.141 (7) | **0.274 (1)** |
| Weather | 0.081 (7) | 0.160 (5) | 0.061(10) | 0.112 (6) | 0.076 (8) | 0.062 (9) | **0.173 (1)** | 0.167 (2) | 0.162 (4) | 0.165 (3) |
| Average | 0.139 (8) | 0.142 (7) | 0.128 (9) | 0.114(10) | 0.183 (3) | **0.198 (1)** | 0.171 (4) | 0.158 (6) | 0.165 (5) | 0.190 (2) |

- ARF: ARF [32] is an online version of random forest classifier, which uses a concept detector to deal with concept drift.
- AWE: AWE [37] is the most representative block-based ensemble, which maintains the top-k classifiers learnt from sequential blocks of instances and dynamic weights each classifier according to the most recent block.
- OzaBagAdwin: OzaBagAdwin is the online bagging version equipped with ADWIN as the change detector.
- LeverageBagging (Lev Bagging): Lev Bagging extends the traditional bagging algorithm to the online mode, and randomizes the weights of instances in the input data stream to enhance the performance of the ensemble.
- AUE2: AUE2 [39] is a block-based ensemble. It uses a non-linear base classifier weighting scheme, incrementally updates each data block after it arrives to address sudden concept drift.
- KUE: KUE [46] is a hybrid ensemble which using the Kappa metric for dynamically weighting base classifiers in ensemble.
- Online SMOTEBagging (OSB): OSB [58] is an online version of bagging based cost-sensitive learning algorithm.
- Learn++.NIE: Learn++.NIE [54] is an ensemble algorithm that adopts bagging-based subensembles to generate subensemble of classifiers and weighting strategy based on recall, $F_{\text{measure}}$, or $G_{-\text{mean}}$.
- ACOG: ACOG [57] is the state-of-the-art online classification based on adaptive regularization.

The results are evaluated by the metrics of $G$-mean, $F_{-\text{measure}}$, and running time, as shown in Tables 2-4.

***G*-mean Analysis.** As shown in Table 2, in terms of $G$-mean, on average TSCS achieved the best, AWE is the worst. In the case of data streams, the best average performance is obtained on LED, SEA, Poker, Sensor and Weather datasets. Compared with other algorithms, TSCS can achieve better performance under class imbalance data streams environment. This is because TSCS has a concept drift detection mechanism and uses the cost-sensitive strategy, so it can handle class imbalances and concept drift well. Due to AWE unable to deal with the class imbalance problem, so its average performance is the worst. Meanwhile, the cost sensitive learning strategy is adopted in TSCS classification stage, and

**TABLE 4.** Times comparison of 10 algorithms (seconds).

| | ARF | AWE | Oza | Lev | AUE2 | KUE | OSB | NIE | ACOG | TSCS |
|---|---|---|---|---|---|---|---|---|---|---|
| Agrawal | 35.54 (5) | **5.33(1)** | 65.03 (8) | 48.11 (7) | 45.54 (6) | 25.33 (3) | 765.03(10) | 115.47(9) | 14.56 (2) | 27.45 (4) |
| STAGGER | 18.64 (3) | 52.03 (5) | 82.29 (7) | 14.60 (2) | **11.64 (1)** | 52.23 (6) | 82.29 (7) | 40.24 (4) | 84.41 (9) | 96.89(10) |
| HyperPlane | 39.67 (2) | 76.13 (8) | **30.29 (1)** | 74.69 (6) | 69.67 (5) | 76.03 (7) | 82.29 (9) | 156.41(10) | 47.36 (3) | 51.23 (4) |
| LED | 12.11 (2) | 36.01 (5) | 43.35 (7) | 31.34 (4) | 18.71 (3) | 36.01 (5) | 43.35 (7) | 52.23 (9) | 68.29 (10) | **7.89 (1)** |
| Rotatingspiral | 24.31 (4) | 25.47(6) | 44.56 (8) | 37.45 (7) | 17.46 (3) | 15.47 (2) | **14.56 (1)** | 25.33 (5) | 765.03(10) | 48.11 (9) |
| SEA | 50.47 (5) | **40.24 (1)** | 84.41 (9) | 76.89 (8) | 46.29 (4) | 41.23 (2) | 84.41 (9) | 55.47 (6) | 44.56 (3) | 67.45 (7) |
| Spam | 58.69 (6) | 56.41 (4) | 247.75(10) | 51.34 (3) | 59.60 (7) | 56.41 (4) | 47.58 (2) | 140.24 (9) | 84.41 (8) | **36.89 (1)** |
| German | 58.05 (8) | 46.27(3) | 50.29 (5) | 67.89(10) | 55.05 (6) | **42.23 (1)** | 59.29 (9) | 56.41 (7) | 47.40 (4) | 44.75 (2) |
| Poker | 45.54 (4) | 52.33 (7) | 75.03 (10) | 48.11 (5) | 45.03 (2) | 45.33 (3) | 65.03 (8) | 52.23 (6) | **39.29 (1)** | 67.89 (9) |
| Electricity | 21.64 (4) | 52.23 (9) | 29.29 (6) | 14.60 (3) | **11.64 (1)** | 12.23 (2) | 37.29 (8) | 25.33 (5) | 75.03 (10) | 35.02 (7) |
| Airlines | **68.13 (1)** | 76.03 (4) | 86.29 (10) | 74.69 (3) | 69.67 (2) | 76.03 (4) | 80.29 (7) | 85.47 (9) | 84.56 (8) | 77.45 (6) |
| IJCNN1 | 52.11 (4) | 56.01 (7) | 63.35 (8) | 71.34 (9) | 52.39 (5) | 46.01 (2) | 53.35 (6) | 50.24 (3) | 84.41 (10) | **45.89 (1)** |
| Sensor | **45.54 (1)** | 55.33 (3) | 165.03 (9) | 148.11(7) | 47.54 (2) | 59.33 (4) | 65.03 (6) | 156.41 (8) | 247.37(10) | 61.73 (5) |
| Weather | 41.23 (3) | 84.41 (9) | 55.76 (6) | 43.54 (5) | 67.45 (8) | **14.56 (1)** | 43.35 (4) | 63.34 (7) | 118.71(10) | 31.69 (2) |
| Average | **40.83 (1)** | 51.02 (5) | 80.19 (8) | 57.34 (6) | 44.12 (3) | 42.75 (2) | 108.80 (9) | 76.77 (7) | 128.96(10) | 50.02 (4) |

a feature subset space which can effectively balance the data distribution is selected. During the training stage, a new base classifier is learned in the feature subspace when concept drift is occurred. In the prediction process, TSCS uses the cost-sensitive ensemble method to make prediction.

***F*$_{-\text{measure}}$ Analysis.** As shown in Table 3, on most scenarios, Lev performs the worst compared with other algorithms. This is because there is no mechanism in Lev to deal with class imbalanced issue, thus it performs poorly. TSCS is ranked the second directly next to the best. For TSCS, cost-sensitive feature selection strategy is implemented by deleting irrelevant and redundant features from the original feature set, and then classification is performed based on this feature set, which further affects the overall classification performance.

**Time Analysis.** As shown in Table 4, we observe that OSB and ACOG are the most time-consuming methods. The better performance of our algorithm might be partially because our algorithm benefits from the adaptive window change detection strategy to capture drift in a timely manner. In most datasets, TSCS adapts to concept changes faster than most of the competitive techniques.

In conclusion, TSCS achieves better average performance with respect to *G*-mean, time and $F$-measure on different types of concept drift scenarios compared to other ensemble approaches that use the same base learner. This is principally for the following reasons: (1) The two-stage cost-sensitive learning schema injected the information of cost into the procedures of feature selection and classification, which can effectively deal with class imbalance issue. (2) The management of the change detection mechanism improves the generalization of classification in different situations, particularly in non-stationary environments.

Figure 3-5 shows the changes of the *G*-mean of the algorithms as the number of instances processed increases. Such graphical plots can display the algorithms' adaptability to different kinds of drifts intuitively.

The scenario of LED data stream simulates a complex change by joining two gradually evolving streams. As shown

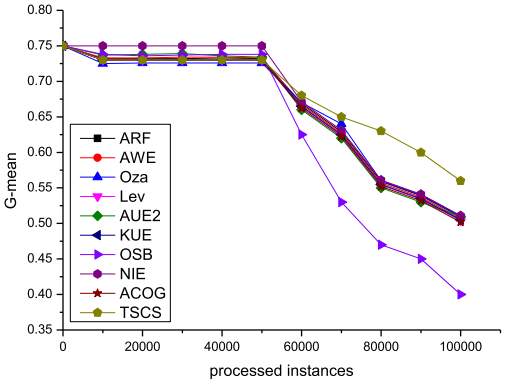


**FIGURE 3.** *G*-mean of algorithms on the LED dataset.

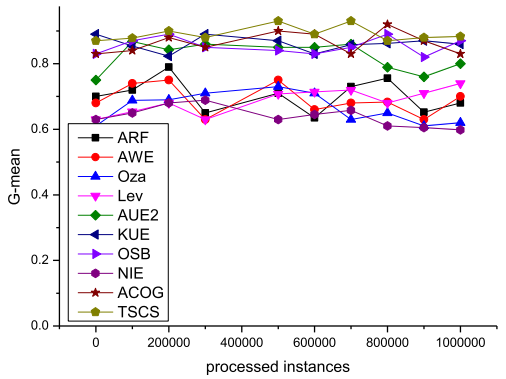


**FIGURE 4.** *G*-mean of algorithms on Poker dataset.

in Figure 3, all the algorithms show relatively smooth curves at the beginning up to 60, 000 instances processed. When concept drift occurs, all the *G*-mean curves suffer from instantaneous accuracy rates fluctuation, including that of TSCS. Since TSCS can track the sudden change immediately using the change detector, it can update the ensemble classifier to adapt to gradual changes in a timely manner. These results confirm that our algorithm provide a certain guarantee for the stability of different kinds of concept drift.

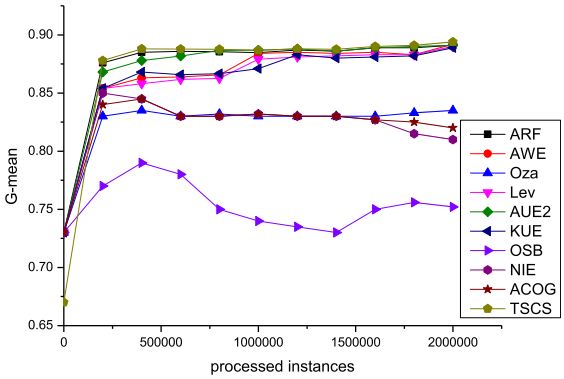


**FIGURE 5.** *G*-mean of algorithms on Sensor dataset.

Conceptual changes in real-world stream environments has the characteristics of uncertainty, so it can verify the adaptability of data stream algorithms. Figure 4 shows *G*-mean of the algorithms on the Poker dataset. The curves of all methods suffer a sudden drop, which indicates that there may exist concept drift. Since this is a real dataset with class imbalance, the AWE and ARF do not have the ability to deal with class imbalance, so the performance is poor. In contrast, the curves of TSCS and Learn++.NIE are relatively stable. Our method adapts the classification model to non-stationary environments passively, incorporates two-stage cost-sensitive mechanisms to address class imbalance issue.

Specially, as shown in Figure 5, TSCS is superior to the other algorithms on the Sensor dataset. More importantly, we can observe that the curves of AUE2, ADOB and ARF decrease at the period of after learning 200K instances to learning 300K instances. Thus, we can infer that there may exist concept drift in this period. However, TSCS performs well in this period. The ensemble methods (TSCS, OSB, Lear$^{++}$.NIE, ACOG) achieve better generalization performance than the other five algorithms. AUE2 and AWE are not designed for dynamic environments and therefore performs poorly on this dataset. TSCS gained the best performance, followed by KUE. The curve of TSCS is relatively stable compared to that of other algorithms. This is due to the fact that TSCS employs the hybrid ensemble algorithm uses an online manner to update the model.

**Sensitivity to Imbalance Ratio.** To further verify the adaption of the five algorithms (KUE, OSB, NIE, ACOG and TSCS) are affected by the imbalance rate (IR). In the experiment, we vary the value of IR from 1 to 20 on the HyperPlane data stream. As the imbalance rate increases over time, the classification task becomes more and more difficult. As shown in Figure 6, the curves of all algorithms drop sharply as the imbalance ratio increase, except TSCS. This might be attributed to the combination of cost-sensitive feature selection and cost-sensitive weighting strategies that provides a good trade-off between improved robustness to class imbalance features and applicability to dynamic data streams scenario. The results indicate that TSCS has strong adaptability to dynamic environment of class imbalance rate.

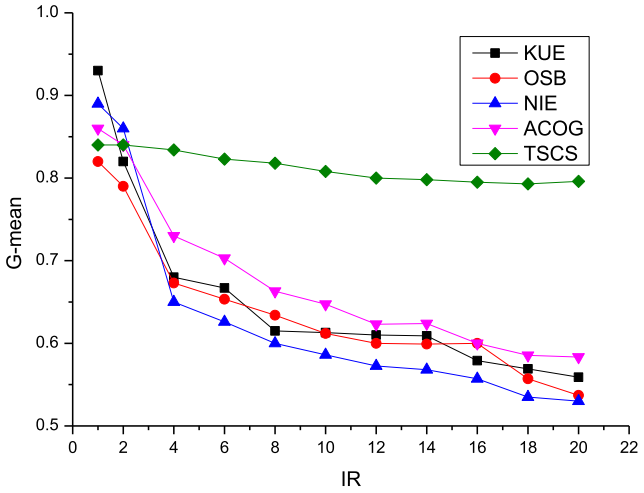


**FIGURE 6.** Sensitivity to imbalance ratio on hyperPlane dataset.

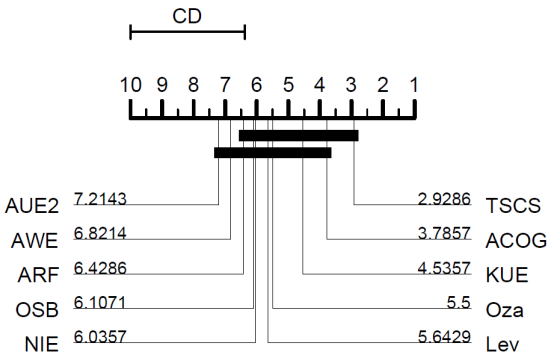


**FIGURE 7.** Critical-different diagram for all the algorithms.

Finally, we did a non-parametric Friedman test for all competing algorithms [67]. In the statistical test, we set the significance level $\alpha$ to 0.05. The test result rejects the null hypothesis, which indicates that there is no significant difference between the performances of all algorithms. Next, we adopt the Nemenyi post-hoc test [68] ($p = 0.05$) to further verify the results. The results shown in Figure 7 reveal that our method is significantly better than AUE2 and AWE.

In summary, all of the above experimental results confirm that TSCS is superior to the competitive existing methods mainly in the following aspects: (1) The cost-sensitive feature selection strategy is implemented by deleting irrelevant and redundant features from the original feature set, and then classification is performed based on this feature set, which further affects the overall classification performance; (2) TSCS can deal with concept drifts quickly and appropriately; and (3) TSCS provides a good performance in both static and dynamic class imbalance environments.

## V. CONCLUSION

This study seeks to understand the capability and explain the role of cost-sensitive learning in dealing with the class

imbalance issue under non-stationary data stream. This research introduced a novel and efficient learning approach to deal with the classification of data streams with class imbalance and concept drifts. It provides an effective way to tackle the learning challenge when concept drifts and class imbalance occurs simultaneously using a two-stage cost-sensitive learning scheme. First, in feature selection process, CSPAC algorithm incorporates cost information into feature selection, which can not only delete redundant features but also be capable of handling class imbalance issue in data stream. Secondly, a cost-sensitive weighting ensemble scheme is devised. In addition, the ensemble is equipped with an adaptive window change detect mechanism to determine when to build a new candidate classifier to adapt drift quickly. Experimental results manifest that our proposed TSCS approach outperforms other methods and achieves the best performance assessed with measurement metrics commonly applied by researchers in the research area, especially for evolving data streams with class imbalance environments.

## REFERENCES

[1] C. C. Aggarwal, *Data Streams: Models and Algorithms*. Berlin, Germany: Springer-Verlag, 2007.
[2] J. Gama, *Knowledge Discovery from Data Streams*. New York, NY, USA: CRC Press, 2010.
[3] I. Zliobaite, M. Pechenizkiy, and J. Gama, "An overview of concept drift applications," in *Big Data Analysis: New Algorithms for a New Society*. Cham, Switzerland: Springer, 2016. 91-114.
[4] G. De Francisci Morales, A. Bifet, L. Khan, J. Gama, and W. Fan, "IoT big data stream mining," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2016, pp. 2119–2120.
[5] H. M. Gomes, J. Read, and A. Bifet, "Machine learning for streaming data: State of the art, challenges, and opportunities," *ACM SIGKDD Explor. Newslett.*, vol. 21, no. 2, pp. 6–22, Feb., 2019.
[6] A. Tsymbal, "The problem of concept drift: Definitions and related work," Dept. Comput. Sci., Trinity College, Dublin, Ireland, Tech. Rep. TCD-CS-2004-15, 2004.
[7] J. Gama, I. Žliobaitė, and A. Bifet, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 231–238, Apr. 2014.
[8] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Comput. Intell. Mag.*, vol. 10, no. 4, pp. 12–25, Nov. 2015.
[9] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining Knowl. Discovery*, vol. 30, no. 4, pp. 964–994, Jul. 2016.
[10] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, and K. Ghédira, "Discussion and review on evolving data streams and concept drift adapting," *Evolving Syst.*, vol. 9, no. 1, pp. 1–23, Mar. 2018.
[11] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2019.
[12] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.
[14] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Trans. Syst., Man, Cybern., B (Cybern.)*, vol. 39, no. 2, pp. 539–550, Apr. 2009.
[15] P. Cao, D. Zhao, and O. Zaiane, "An optimized cost-sensitive SVM for imbalanced data learning," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining* Berlin, Germany: Springer, 2013, pp. 280–292.
[16] W. Fan, S. J. Stolfo, and J. Zhang, "AdaCost: Misclassification cost-sensitive boosting," in *Proc. 16th Int. Conf. Mach. Learn.* San Mateo, CA, USA: Morgan Kaufmann, vol. 1999, pp. 97–105.
[17] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognit.*, vol. 40, no. 12, pp. 3358–3378, Dec. 2007.
[18] 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 Trans. Syst., Man, Cybern., C (Appl. Rev.)*, vol. 42, no. 4, pp. 463–484, Jul. 2012.
[19] I. Žliobaitė, M. Budka, and F. Stahl, "Towards cost-sensitive adaptation: When is it worth updating your predictive model?" *Neurocomputing*, vol. 150, pp. 240–249, Feb. 2015.
[20] J. M. Williams, A. Samal, P. K. Rao, and M. R. Johnson, "Paired trial classification: A novel deep learning technique for MVPA," *Frontiers Neurosci.*, vol. 14, p. 417, Apr. 2020.
[21] L. Zhuang and H. Dai, "A novel field learning algorithm for dual imbalance text classification," in *Proc. 2nd Int. Conf. Fuzzy Syst. Knowl. Discovery* Berlin, Germany: Springer-Verlag, 2005, pp. 39–48.
[22] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2001, pp. 97–106.
[23] I. Frías-Blanco, J. D. Campo-Ávila, G. Ramos-Jiménez, A. C. P. L. F. Carvalho, A. Ortiz-Díaz, and R. Morales-Bueno, "Online adaptive decision trees based on concentration inequalities," *Knowl.-Based Syst.*, vol. 104, pp. 179–194, Jul. 2016.
[24] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2000, pp. 71–80.
[25] J. Gama, R. Rocha, and P. Medas, "Accurate decision trees for mining high-speed data streams," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2003, pp. 523–528.
[26] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2009, pp. 139–148.
[27] A. Bifet and R. Gavaldá, "Adaptive learning from evolving data streams," in *Proc. 8th Int. Symp. Intell. Data Anal.* Berlin, Germany: Springer-2009, pp. 246–260.
[28] J. Gama, P. Medas, and G. Castillo, "Learning with drift detection," in *Proc. 17th Brazilian Symp. Artif. Intell. (SBIA)* (Lecture Notes in Computer Science), vol. 3171. Berlin, Germany: Springer-Verlag, 2004, pp. 286–295.
[29] M. Baena-García, D. J. Campo-Ávila, and R. Fidalgo, "Early drift detection method," in *Proc. 4th Int. Workshop Knowl. Discovery Data Streams (KDD)*, New York, NY, USA, 2006, pp. 77–86.
[30] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proc. SIAM Int. Conf. Data Mining*. Philadelphia, PA, USA: SIAM, Apr. 2007, pp. 443–448.
[31] G. Philipp, "Scalable detection of concept drifts on data streams with parallel adaptive windowing," in *Proc. 21st Int. Conf. Extending Database Technol. (EDBT)*, 2018, pp. 1–4.
[32] H. M. Gomes, A. Bifet, and J. Read, "Adaptive random forests for evolving data stream classification," *Mach. Learn.*, vol. 6, pp. 1–27, Oct. 2017.
[33] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Inf. Fusion*, vol. 37, pp. 132–156, Sep. 2017.
[34] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 1–36, Jun. 2017.
[35] D. Brzezinski, "Block-based and online ensembles for concept-drifting data streams," Ph.D. dissertation, Dept. Comput. Sci., Poznań Univ. Technol., Poznań, Poland, 2015.
[36] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2001, pp. 377–382.
[37] H. Wang, W. Fan, P. S. Yu, and J. W. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2003, pp. 226–235.
[38] D. Brzeziński and J. Stefanowski, "Accuracy updated ensemble for data streams with concept drift," in *Proc. 6th Int. Conf. Hybrid Artif. Intell. Syst. (HAIS)* (Lecture Notes in Computer Science), vol. 6678. Berlin, Germany: Springer-Verlag, vol. 2011, pp. 155–163.
[39] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 81–94, Jan. 2014.

[40] R. Elwell and R. Polikar, "Incremental learning of concept drift in non-stationary environments," *IEEE Trans. Neural Netw.*, vol. 22, no. 10, pp. 1517–1531, Oct. 2011.
[41] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, Dec. 2007.
[42] H. M. Gomes and F. Enembreck, "SAE2: Advances on the social adaptive ensemble classifier for data streams," in *Proc. 29th Annu. ACM Symp. Appl. Comput.*, 2014, pp. 798–804.
[43] Y. Lu, Y.-M. Cheung, and Y. Y. Tang, "Dynamic weighted majority for incremental learning of imbalanced data streams with concept drift," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 2393–2399.
[44] K. Nishida, K. Yamauchi, and T. Omori, "ACE: Adaptive classifiers-ensemble system for concept-drifting environments," in *Proc. 6th Int. Workshop Multiple Classifier Syst. (MCS)* (Lecture Notes in Computer Science), vol. 3541. Berlin: Springer-Verlag, 2005, pp. 176–185.
[45] D. Brzezinski and J. Stefanowski, "Combining block-based and online methods in learning ensembles from concept drifting data streams," *Inf. Sci.*, vol. 265, pp. 50–67, May 2014.
[46] A. Cano and B. Krawczyk, "Kappa updated ensemble for drifting data stream mining," *Mach. Learn.*, vol. 109, no. 1, pp. 175–218, Jan. 2020.
[47] H. Dai, X. Hang, and G. Li, "Inexact field learning: An approach to induce high quality rules from low quality data," in *Proc. IEEE Int. Conf. Data Mining*, Nov./Dec. 2001, pp. 586–588.
[48] H. Dai, "Learning of Forecasting Rules from Very Large Noisy Real Observational Meteorological Data Bases," Ph.D. dissertation, Dept. Comput. Sci., Roy. Melbourne Inst. Technol., Melbourne VIC, Australia, 1994.
[49] N. V. Chawla, A. Lazarevic, and L. O. Hall, "SMOTEBoost: Improving prediction of the minority class in boosting," in *Proc. 7th Eur. Conf. Princ. Data Mining Knowl. Discovery* Berlin, Germany: Springer, 2003, pp. 107–119.
[50] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: A hybrid approach to alleviating class imbalance," *IEEE Trans. Syst., Man, Cybern., A, Syst. Humans*, vol. 40, no. 1, pp. 185–197, Jan. 2010.
[51] J. Gao, W. Fan, J. Han, and P. S. Yu, "A general framework for mining concept-drifting data streams with skewed distributions," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2007, pp. 3–14.
[52] S. Chen and H. He, "Towards incremental learning of nonstationary imbalanced data stream: A multiple selectively recursive approach," *Evolving Syst.*, vol. 2, no. 1, pp. 35–50, Mar. 2011.
[53] B. Mirza and Z. Lin, "Meta-cognitive online sequential extreme learning machine for imbalanced and concept-drifting data classification," *Neural Netw.*, vol. 80, pp. 79–94, Aug. 2016.
[54] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2283–2301, Oct. 2013.
[55] 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, Dec. 2013.
[56] 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, Nov. 2017.
[57] P. Zhao, Y. Zhang, M. Wu, S. C. H. Hoi, M. Tan, and J. Huang, "Adaptive cost-sensitive online classification," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 214–228, Feb. 2019.
[58] B. Wang and J. Pineau, "Online bagging and boosting for imbalanced data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 12, pp. 3353–3366, Dec. 2016.
[59] J. Wang, P. Zhao, and S. C. H. Hoi, "Cost-sensitive online classification," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 10, pp. 2425–2438, Oct. 2014.
[60] L. Korycki, A. Cano, and B. Krawczyk, "Active learning with abstaining classifiers for imbalanced drifting data streams," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Los Angeles, CA, USA, Dec. 2019, pp. 2334–2343.
[61] Y. Sun, Z. Wang, Y. Bai, H. Dai, and S. Nahavandi, "A classifier graph based recurring concept detection and prediction approach," *Comput. Intell. Neurosci.*, vol. 2018, pp. 1–13, Jun. 2018.
[62] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 2, no. 4, pp. 433–459, Jul. 2010.
[63] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive online analysis," *J. Mach. Learn. Res.*, vol. 11, pp. 1601–1604, May 2010.
[64] D. Dua, and C. Graff. (2019). *UCI Machine Learning Repository*. School of Information and Computer Science, University of California, Irvine, CA, USA. [Online]. Available: http://archive.ics.uci.edu/ml
[65] D. Brzezinski, J. Stefanowski, R. Susmaga, and I. Szczech, "On the dynamics of classification measures for imbalanced and streaming data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 2868–2878, Aug. 2020.
[66] S. Wang, L. L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4802–4821, Oct. 2018.
[67] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.
[68] N. Settouti, M. E. A. Bechar, and M. A. Chikh, "Statistical comparisons of the top 10 algorithms in data mining for classification task," *Int. J. Interact. Multimedia Artif. Intell.*, vol. 4, no. 1, pp. 46–51, Jan. 2016.

**YANGE SUN** received the M.S. degree from Central China Normal University, in 2007, and the Ph.D. degree in computer science and technology from Beijing Jiaotong University, Beijing, China, in 2019, all in computer science. She is currently a Lecturer with Xinyang Normal University, Xinyang, China. Her research interests include data mining and machine learning.

**YI SUN** received the Ph.D. degree in computer science and technology from Beijing Jiaotong University, Beijing, China, in 2015. She is currently an Associate Professor with the Zhengzhou Information Science and Technology Institute, Zhengzhou, China. Her research interests include cloud computing, big data security, and stream secure exchange.

**HONGHUA DAI** received the M.Sc. degree from the Graduate School, Chinese Academy of Sciences, in 1986, and the Ph.D. degree from the Department of Computer Science, RMIT University, in 1994. After he received his Ph.D. degree, he worked in several universities, including Monash University, University of New England, Deakin University, Zhejiang University, Carnegie Mellon University, and Zhengzhou University. His major research interests include minimum message length principle-based causal discovery, reliable knowledge discovery, machine learning for weather forecasting, big data intelligence and digital health, and causality intelligence.