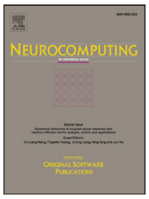
Neurocomputing 000 (2018) 1–17

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



The Gradual Resampling Ensemble for mining imbalanced data streams with concept drift

Siqi Ren[a], Bo Liao[a,*], Wen Zhu[a], Zeng Li[b], Wei Liu[a], Keqin Li[a,c]

[a] College of Information Science and Engineering, Hunan University, Changsha, Hunan 410082, China
[b] School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China
[c] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

A R T I C L E I N F O

Article history:
Received 9 March 2017
Revised 10 November 2017
Accepted 24 January 2018
Available online xxx

Communicated by Dr Xiaofeng Zhu

Keywords:
Concept drift
Data stream mining
Ensemble classifier
Class imbalance

A B S T R A C T

Knowledge extraction from data streams has received increasing interest in recent years. However, most of the existing studies assume that the class distribution of data streams is relatively balanced. The reaction of concept drifts is more difficult if a data stream is class imbalanced. Current oversampling methods generally selectively absorb the previously received minority examples into the current minority set by evaluating similarities of past minority examples and the current minority set. However, the similarity evaluation is easily affected by data difficulty factors. Meanwhile, these oversampling techniques have ignored the majority class distribution, thus risking class overlapping.

To overcome these issues, we propose an ensemble classifier called Gradual Resampling Ensemble (GRE). GRE could handle data streams which exhibit concept drifts and class imbalance. On the one hand, a selectively resampling method, where drifting data can be avoidable, is applied to select a part of previous minority examples for amplifying the current minority set. The disjuncts can be discovered by the DBSCAN clustering, and thus the influences of small disjuncts and outliers on the similarity evaluation can be avoidable. Only those minority examples with low probability of overlapping with the current majority set can be selected for resampling the current minority set. On the other hand, previous component classifiers are updated using latest instances. Thus, the ensemble could quickly adapt to a new condition, regardless types of concept drifts. Through the gradual oversampling of previous chunks using the current minority events, the class distribution of past chunks can be balanced. Favorable results in comparison to other algorithms suggest that GRE can maintain good performance on minority class, without sacrificing majority class performance.

1. Introduction

1.1. Motivation

With the explosive growth of information, traditional methods of extracting knowledge have been far from meeting the needs of actual researches, thus contributing to the emergence of data streams. Streaming data have been widely adopted in many applications, such as wireless sensor network, web click streams, and scientific data. Traditional data mining techniques extract useful information from limited data. They require to scan the training data several times, which consumes a significant amount of time and memory [1]. In a streaming condition, the supplement of data items is unbounded, and only a small summary can be loaded into memory. Therefore, only approximate results can be obtained. The speed of data generation is high, and thus streaming models focus on applying data mining techniques with linear/sublinear time complexity by incrementally handling data items [2]. Concept drift is a crucial characteristic of data streams, which describes the dynamic property of data items [3]. Consequently, models should be adjusted or even rebuilt to adapt to a new concept.

Class imbalance exists in many real-world applications, such as network intrusion detection and credit card transactions. Learning from skewed data refers to the situation where certain types of observations are seriously underrepresented compared with others. The decision boundary of a traditional classifier is likely to incline to minority samples, which inevitably leads to a poor predictive accuracy on the underrepresented data. Data difficulty factors, including small sub-concepts [4], class overlapping [5], and outliers [6], have been treated as more influential elements than the imbalance ratio.

* Corresponding author.
E-mail addresses: siqirenzl@163.com (S. Ren), dragonbw@163.com (B. Liao), syzhuwen@163.com (W. Zhu), lizeng@mail.ustc.edu.cn (Z. Li), lw2001184@163.com (W. Liu), lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.neucom.2018.01.063
0925-2312/© 2018 Elsevier B.V. All rights reserved.

The issues of concept drifts and class imbalance have been studied separately. However, the joint problem has been mostly underexplored. The following issues need to be resolved in this area:

1. Modifying the model timely: removing obsolete knowledge and capturing new data to catch up with the changing concepts.
2. Preserving valuable information: storing minority instances for oversampling the current underrepresented data.
3. Balancing the predictive accuracies of models on minority and majority instances: intensifying the underrepresented class concept without sacrificing the predictive accuracy on majority samples.
4. Considering data difficulty factors in the resampling procedure: excluding the influences of outliers and small disjuncts on the similarity evaluation and avoiding the class overlapping issue.

1.2. Contribution

This paper aims to create a framework for learning concept drifts from imbalanced data. In summary, the key contributions are as follows:

1. The Gradual Resampling Ensemble (GRE) algorithm is a new hybrid ensemble that integrates the operations of chunk-based ensembles with those of online ensembles. GRE gradually oversamples minority samples of past chunks using the current minority set, thus improving the predictive powers of past component classifiers on minority instances. In general, the latest chunk is the best role to describe the current and near-future data distributions. Therefore, GRE can quickly adapt to a new condition, regardless of types of drifts. Meanwhile, GRE periodically amplifies the data sets of previous blocks using the observations in the latest block. Thus, GRE is robust against different predefined chunk sizes.
2. For the latest block, a selectively resampling technique is employed to improve the recognition ratio of the ensemble on minority events. Most of the existing resampling techniques ignore the influences of data difficulty factors on the similarity evaluation among minority samples. In GRE, the small disjuncts are discovered by clustering, and thus the similarity evaluation cannot be affected by outliers and disjoints. Meanwhile, those past minority samples that have small distances from the current minority set have the priority to be absorbed into the training sets of candidate hypotheses, which can avoid introducing drifting data. Different from previous studies, GRE also considers the similarities between past minority sets and the current majority set to reduce the probability of class overlapping.
3. The final decision of a testing instance is supported by a combination result obtained from all the ensemble members, in which each past member is weighted by its prediction accuracy on the observations in the candidate block. GRE maintains a part of previous component classifiers in the ensemble group, thereby preventing forgetting catastrophically. Where a cross-validation procedure is avoidable, the weights of candidate hypotheses are designated as a supreme value, regardless of their performances. Treating the candidate classifiers as best members is especially important in the presence of sudden changes when only the latest chunk can represent the distribution of testing data [7].
4. Through the Massive Online Analysis (MOA) platform [8], characteristics of GRE are analyzed. The statistical results demonstrate that different chunk sizes cannot have a significant effect on the performance of GRE. Using a small number of candidate classifiers can obtain good predictive performance by preserving more knowledge of data streams. The comparative study is conducted to analyze the effectiveness of GRE. The statistical tests show that our method can well learn concept drifts from imbalanced data.

1.3. Paper organization

The rest of the paper is organized as follows. Section 2 reviews related works on classifiers for dynamic data streams. In Section 3, we present the popular techniques for classifying imbalanced data streams. Section 4 provides a detailed introduction of our proposed algorithm. Experimental analysis and results are presented in Section 5. Section 6 concludes the paper and suggests a direction for future researches.

2. Classifiers for data streams with concept drifts

A streaming classifier requires to process incoming data sequentially and make its result as accurate, or nearly as accurate, as a model trained on the whole data. In a nonstationary environment, the underlying data distribution changes over time, and this phenomenon is referred to as concept drift [9]. To capture the evolution of underlying concepts, streaming classifiers should develop a strategy for capturing new data and eliminating obsolete knowledge in the training set.

A wide variety of data stream classifiers, which can be characterized as single classifiers and ensembles, have been developed. Single classifiers are generally equipped with a forgetting mechanism to cope with dynamic data streams. Many forgetting mechanisms are designed to weaken the effect of obsolete data, such as windowing techniques that remove obsolete knowledge [10,11], instance weighting methods that weaken the importance of obsolete data [12], and change detectors that discover concept drifts [13–15]. Very Fast Decision Tree (VFDT) [1] proposed by Domingos and Hulten is a single model for static data streams. VFDT cannot obtain all the data, thus using the Hoeffding bound to determine the number of instances for splitting a node with a certain probability. Furthermore, Concept-adapting Very Fast Decision Tree (CVFDT) [16] extends VFDT with a fixed-size sliding window to handle concept drifts.

When tackling nonstationary concepts, ensemble-based models are advantageous over single classifiers. First, ensembles provide a natural way of coping with concept drifts without the need for any forgetting mechanisms. Second, they are easy to scale and parallelize. Third, they can quickly adapt to changes by pruning badly-behaved ensemble members. Finally, the good generalization ability can be obtained as previous information of data streams can be reused. Nevertheless, ensembles often require several times more processing time than single classifiers, plus the procedures of selecting members referred to as ensemble pruning and importance evaluation of every hypothesis known as component weighting can make the process even longer.

The ensemble-based classifiers can be divided into two categories based on the amount of data to be processed during each training step: chunk-based ensembles and online ensembles. For chunk-based ensembles, fixed-size chunks are divided and hypotheses are trained over a certain amount of data. Thus, they are not purely online approaches. By adjusting the weights of hypotheses, Accuracy Weighted Ensemble (AWE) [17] and Streaming Ensemble Algorithm (SEA) [18] can gradually forget outdated knowledge, thus providing preferable manipulations for gradual drifts. Moreover, the weights of components in Learn++.NSE [19] are dynamically updated with the time-changing errors of classifiers on the current and past concepts to handle various kinds of concept drifts.

Unlike the above approaches, online ensembles update models after receiving a single example, which strictly comply with the requirements of online learning. Online Bagging [20], which is an online version of Bagging [21], presents every incoming instance to a component several times. Online Boosting [20], which can be applied to a large volume of streaming data, is the online version of Boosting [22]. The probability of an instance used for training models is determined by the Poisson distribution in Online Boosting. In Dynamic Weighting Majority (DWM) [23], the weight of every component classifier is set to one initially and adjusted after every observation. Considering the two schemes previously mentioned, a hybrid approach has been developed. Accuracy Updated Ensemble (AUE) [7] is a famous approach for learning data streams with multiple kinds of concept drifts. The ensemble system can quickly accommodate new circumstances by assimilating new knowledge into past component classifiers.

3. Previous work on handling data streams with imbalanced class distribution

The issue of concept drift is further complicated if the class distribution of a data set is imbalanced. The minority instances, which are underrepresented in static circumstances, tend to be seriously neglected in a dynamic learning framework. As a branch of data steam mining, intensifying the underrepresented class concepts when classifying skewed data streams has raised concerns in recent years. Thus, a classifier for dealing with the joint problem of concept drifts and class imbalance is a necessity.

Many existing methods process data items in chunks and balance the class distribution of the latest chunk using various strategies, such as oversampling the current minority set using minority events of the previous chunks [24–27], oversampling the current minority set using minority samples in a sliding window [28], and undersampling the current majority set [29]. Gao et al. presented a representative of block-based algorithms called Uncorrelated Bagging (UB) [24]. In UB, all minority examples of consecutive blocks are preserved and used to supplement the current minority set. Therefore, drifting data are likely to be absorbed into the latest block. An ensemble is generated through dividing the post-balanced chunk into several sub-blocks. The majority observations are randomly propagated into sub-blocks to ensure the diversity. SElectively Recursive Approach (SERA) [25] was proposed to selectively oversample minority examples from previous chunks to balance the class distribution of the candidate chunk. Considering the evolution of concepts, SERA uses the Mahalanobis distance to evaluate the probabilities of past minority data to be selected. Based on SERA, Multiple Selectively Recursive Approach (MuSeRA) [27] and Recursive Ensemble Approach (REA) [30] were developed. Compared with SERA, MuSeRA maintains the hypotheses built over all the chunks, and then the combination result is leveraged to predict an incoming observation. Targeted for handling sub-concepts within the minority set, REA adopts *k*-nearest neighbors to estimate the similarities among minority samples. In Dynamic Feature Group Weighting with Importance Sampling (DFGW-IS) [28], a fixed-size window is maintained to collect the latest minority data for amplifying the current minority set, which assumes that the latest data can best represent current and near-future concepts.

Learn++ for Nonstationary and Imbalanced Environments (Learn++.NIE) and Learn++ for Concept Drift with SMOTE (Learn++.CDS) do not need to access previous data [31]. Learn++.CDS is a natural combination of Learn++.NSE [19] and Synthetic Minority class Oversampling TEchnique (SMOTE) [32], in which Learn++.NSE handles concept drifts and SMOTE balances the distribution by generating new data. Furthermore, Learn++.NIE modifies the weighting mechanism with a penalty constraint to balance the importances of different classes. Without creating synthetic data, it replaces SMOTE with bagging-based sub-ensembles. Oversampling-based Online Bagging (OOB) and Undersampling-based Online Bagging (UOB), which are based on resampling and time-decayed metrics, are two online approaches [33]. The sampling rates of them are consistent with the imbalance degree of data. The study shows that UOB is good at handling static data streams, whereas OOB is more robust against dynamic conditions.

4. The Gradual Resampling Ensemble (GRE) algorithm

In this section, we first present the details of the GRE algorithm. Then, the inherent handling mechanisms of GRE, which cover the selectively resampling mechanism, ensemble update mechanism, weighting mechanism, and final decision, are respectively described. The frequently used symbols and their descriptions in this paper are summarized in Table 1.

Algorithm 1 Gradual Resampling Ensemble (GRE).

```
Input: $S$: data stream
       $m$: the current time stamp
       $S_m$: the current training set with $a$ data items $\{(x_1,y_1),\cdots,(x_a,y_a)\}$
       $T_m$: the current testing set with $b$ data items $\{x_1',x_2'\cdots,x_b'\}$
       $k$: the predefined ensemble size.
       $p$: the number of new hypotheses built over the latest block
Output: $\hat{y}_i$: predictive label for testing set $x_i'$
1: for all data chunks $B_m \in S$ do
2:     split $S_m$ into $P_m$ and $N_m$
3:     $r_m \leftarrow \frac{|P_m|}{|N_m|}$
4:     if $r_m < f$ then
5:         call the resampling procedure in Algorithm 2 and obtain $P_m'$
6:     end if
7:     build new hypotheses $K_l'$ based on $S_m' = \{P_m', N_m\}$
8:     compute $w_{K_l'}$ using Eq. (3)
9:     for all the past ensemble members $K_{t,j}$ do
10:        compute $w_{t,j}^m$ using Eq. (5)
11:    end for
12:    achieve the $p$ poorest performing hypotheses
13:    substitute the $p$ poorest hypotheses using $K_l'$
14:    for all the past ensemble members $K_{t,j}$ do
15:        select a set $N$ from $N_m$ randomly, where $|N|$ is $|P_m| + |P_{t,j}| - |N_{t,j}|$
16:        update $K_{t,j}$ using $P_m \cup N$
17:    end for
18: end for
19: for $i = 1,2,\cdots,b$ do
20:    assign $\hat{y}_i$ using Eq. (7)
21: end for
```

4.1. The proposed learning framework

The complete procedure of GRE is provided in Algorithm 1. The GRE algorithm involves four key phases. First, a selectively resampling technique is applied to the current block, which is described in Section 4.2 (lines 4–6 of Algorithm 1). The pseudo-code of the selectively resampling mechanism is presented in Algorithm 2. To limit the memory usage, only minority samples in the previous *w* blocks are preserved in *M* to resample the minority set of the latest block. This is based on the assumption that the latest examples can well represent the current and near-future concepts. If the number of examples in *M* cannot balance the current class distribution, the entire samples in *M* are added into the current training chunk S_m $(m > 1)$ (lines 1 and 2 of Algorithm 2). On the contrary, a clustering procedure is applied to the current minority set P_m to obtain a series of clusters (line 5 of Algorithm 2). Then, the similarity between each of past minority samples $X_u \in M$ and P_m is based on the Mahalanobis distances of X_u from clusters of P_m (line 6 of Algorithm 2). In this way, the influences of small disjuncts and outliers on the similarity evaluation can be avoidable.

Table 1
Frequently used symbols and their descriptions.

Symbol	Description	Symbol	Description
m	The current time stamp	B_m	The current data block
S_m and T_m	The current training and testing sets, respectively	P_m and N_m	The current minority and majority sets, respectively
M and $\|M\|$	The set containing minority examples of previous w blocks and its cardinality	N and $\|N\|$	The majority set for updating past hypotheses and its cardinality
$S_m^{'}$	The current amplified training set	$K_{t,j}$ and $w_{t,j}^{m}$	The jth hypothesis in the tth block and its weight
$P_m^{'}$	The current amplified minority set	$K_l^{'}$ and $w_{K_l^{'}}$	The lth candidate hypothesis and its weight
d	Chunk size	k	Ensemble size
p	The number of candidate hypotheses	X_u	An example in M
f	The post-balance ratio	r_m	The imbalance ratio of S_m
a and b	The numbers of data items in S_m and T_m, respectively	x_i and y_i	A training observation and its label
$x_i^{'}$ and $y_i^{'}$	A testing observation and its label	$\hat{y}_i$	The predictive label of $x_i^{'}$
d_{X_u}	The dissimilarity between X_u and P_m	$d_{X_u}^{'}$	The dissimilarity between X_u and N_m
$\|P_m\|$ and $\|N_m\|$	The numbers of instances in P_m and N_m, respectively	C_i	The misclassification cost of x_i
$K_{t,j}(x_i^{'})$	The predictive label of $x_i^{'}$ using $K_{t,j}$	$K_l^{'}(x_i^{'})$	The predictive label of $x_i^{'}$ using and $K_l^{'}$
$f(\cdot,\cdot)$	Indicator function	$f_{t,j}^{m}(x_i)$	The probability that x_i is classified as y_i by $K_{t,j}$
$\mathrm{MSE}_{t,j}^{m}$	The mean square error of $K_{t,j}$ on examples in S_m	MSE_r	The mean square error of a random prediction

Algorithm 2 Selectively Resampling Procedure.

```
Input: $B_m$: the latest data chunk
    $S_m$: the latest training data chunk
    $r_m$: the imbalanced ratio specifying the proportions between the minority samples and majority samples in the latest training chunk
    $f$: the post-balance ratio specifying the imbalance ratio after resampling the minority set of the latest training chunk
    $a$: the number of training data in the latest training data chunk
    $M$ and $|M|$: the set containing minority data in the previous $w$ blocks and its cardinality, obviously $M = \varnothing$ when $m$=1.
    $P_m$: a dataset containing minority data of the latest training chunk
Output: $P_m^{'}$: the updated minority set in the latest training chunk
 1: if $|M| < (f - r_m) \times a$ then
 2:     $M_m \leftarrow M$
 3: else
 4:     for $u = 1, 2 \ldots$ do
 5:         cluster $P_m$ into several clusters
 6:         obtain distances $d_{X_u,c}$ of an instance $X_u \in M$ from clusters of $P_m$
 7:         $d_{X_u} \leftarrow \min d_{X_u,c}$
 8:         sort $\{d_{X_u}\}$ in ascending order and obtain the order $q_{1,X_u}$ for $X_u$
 9:         cluster $N_m$ into several clusters
10:         obtain distances $d_{X_u,c}^{'}$ of an instance $X_u \in M$ from clusters of $N_m$
11:         $d_{X_u}^{'} \leftarrow \min d_{X_u,c}^{'}$
12:         sort $\{d_{X_u}^{'}\}$ in descending order and obtain the order $q_{2,X_u}$ for $X_u$
13:     end for
14:     determine the order of $X_u$ of being selected, which is corresponding to the sum $q_{X_u} \leftarrow q_{1,X_u} + q_{2,X_u}$
15:     select a minority set $M_m$ with respect to the first $(f - r_m) \times a$ terms from $M$
16: end if
17: $P_m^{'} \leftarrow P_m \cup M_m$
```

Consequently, d_{X_u} is the distance of X_u from its nearest cluster center, which can be used to judge the rank of X_u to be selected for oversampling P_m (lines 7 and 8 of Algorithm 2). Meanwhile, the dissimilarity between a past minority event $X_u \in M$ and the current majority set N_m should be estimated to reduce the risk of overlapping between different classes. Similarly, through clustering samples in N_m (line 9 of Algorithm 2), the dissimilarity between X_u and N_m can be represented as $d_{X_u}^{'}$ (lines 10 and 11 of Algorithm 2). The orders of distances d_{X_u} and $d_{X_u}^{'}$ are sorted in ascending and descending manners, respectively (lines 8 and 12 of Algorithm 2). Consequently, the order of a past minority sample X_u to be selected for oversampling the latest minority set P_m depends on the sum of two orders of distances (line 14 of Algorithm 2). Only those data points, which are close to the current minority set P_m and relatively deviate from the current majority set N_m, are selected and absorbed into P_m (lines 15 and 17 Algorithm 2). After resampling the training minority set of the latest block, the training set in the amplified block, denoted by $S_m^{'}$, is divided into p sub-blocks. All the minority instances and a certain number of majority instances are involved in each sub-block. Then, p candidate hypotheses $K_l^{'}$ $(l = 1, 2, \ldots, p)$ are built (line 7 of Algorithm 1).

Second, the weights of new hypotheses and previous hypotheses are respectively evaluated, which are described in Section 4.4 (lines $8 - 11$ of Algorithm 1). A performance-based pruning technique is adopted to limit the memory and time us-

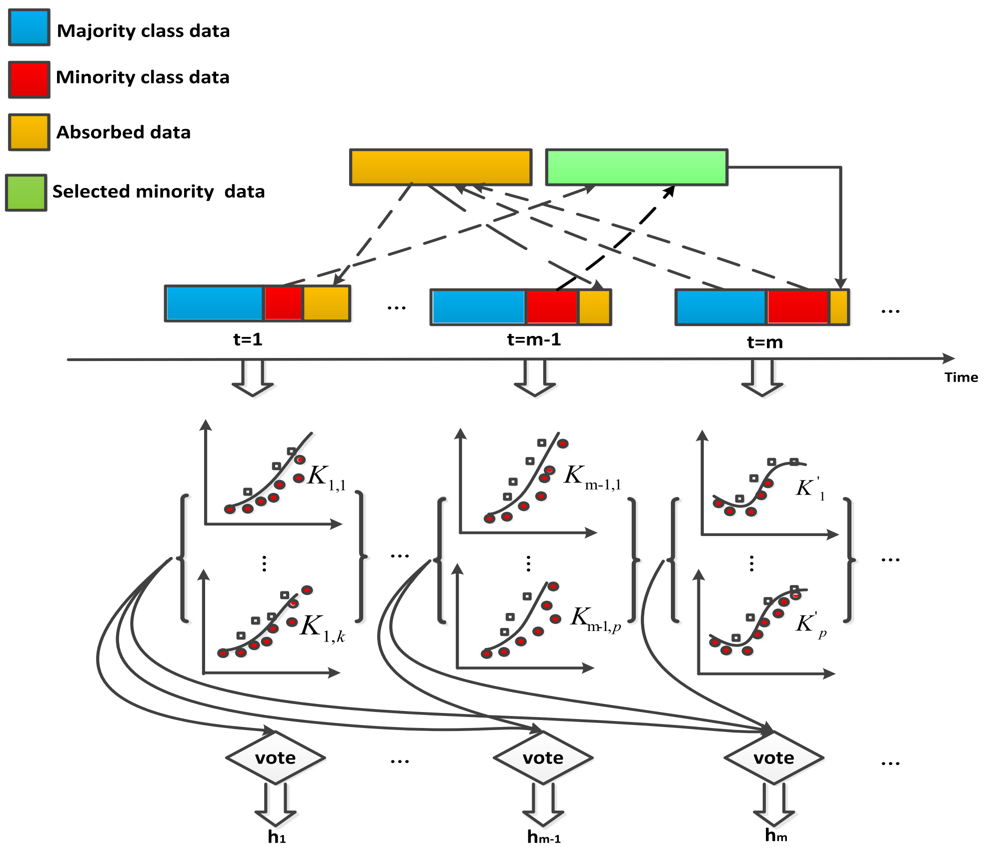


Fig. 1. The learning flow of Gradual Resampling Ensemble algorithm.

age in the ensemble framework. The newly built hypotheses should replace the p poorest performing ones in the ensemble group (lines 12 and 13 of Algorithm 1). Third, a majority set N is randomly selected from the current majority set N_m to update past hypotheses, which is described in Section 4.3 (lines 14 – 17 of Algorithm 1). The number of examples in N, denoted by $|N|$, is equal to $|P_m| + |P_{t,j}| - |N_{t,j}|$ (line 15 of Algorithm 1). $|P_m|$ denotes the number of training minority examples in the current block. Meanwhile, $|P_{t,j}|$ and $|N_{t,j}|$, respectively, denote the numbers of minority and majority examples of the training set of the jth hypothesis in the tth block. Then, previous ensemble members are updated using the latest events, which can further balance the class distributions of training data of past members and makes the ensemble quickly react to different kinds of concept drifts (line 16 of Algorithm 1). Finally, the final decision for a testing event x'_i is derived from all the prediction results of hypotheses, which is described in Section 4.4 (lines 19 – 21 of Algorithm 1).

Fig. 1 provides the system level framework of the proposed algorithm. First, a data stream is divided into equal sized blocks, where B_m is the block at time $t = m$. The data set M conserves the minority samples in the previous w blocks. Second, at time $t = 1$, k component classifiers are built over the first block, where k is the predefined ensemble size. At time $t = m$, a certain number $((f - r_m) \times a)$ of minority examples are selected from M based on the selectively resampling mechanism, where f is the post-balance ratio specifying the imbalance ratio after resampling the minority set of the latest training chunk, r_m is the imbalance ratio specifying the proportions between the minority samples and majority samples in the latest training chunk, and a is the number of training examples of candidate hypotheses. These minority examples are added into B_m so that the imbalance ratio of the amplified candidate training block is equal to f. Then, p hypotheses are built over the amplified training block S'_m and then added into the ensemble. Previous p hypotheses are replaced with new hypotheses to limit the time and memory consumption based on the performance-based pruning technique. Third, past hypotheses are updated using instances in the recent chunk, which makes the ensemble quickly adapt to new conditions. Finally, the ensemble members are weighted to predict the label of a testing instance.

4.2. Selectively resampling mechanism

In the chunk-based framework, several existing techniques are proposed to resample the minority set of the latest chunk, including reusing past minority data to amplify the latest block, generating novel data according to the data distribution, and oversampling minority data randomly. Reusing previous data and generating synthetic events can really absorb novel knowledge into the candidate chunk compared to that in the random oversampling technique. Besides, the minority data of past chunks are more suitable for creating the current concept than the synthetic data generated by some strategies [25].

Data difficult factors and concept drift impose difficulties on the selection of an appropriate subset of past minority data. First, GRE can avoid absorbing drifting data into the current minority set by measuring the similarities between past minority data and the current minority set. Those minority examples that are close to the latest minority set have the priority to be selected. The Mahalanobis distance is treated as the evaluation standard to measure such similarities. Considering the data correlation, Mahalanobis distance is suitable for evaluating similarities compared to Euclidean distance [27].

Second, to avoid the influences of outliers and small disjuncts on the similarity evaluation, GRE divides the current minority set into several clusters based on the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [34]. DBSCAN is a density-based clustering algorithm, which can find arbitrarily shaped clusters and is robust to outliers. Instead of regarding the recent minority set as a whole, each cluster is conceived as a unit for assessing similarities between itself and past minority events. The effect arising from sub-concepts on the similarity evaluation could be avoidable by calculating Mahalanobis distances of past minority samples from cluster centers. For every past minority event, the distance derived from its nearest cluster is employed to describe its probability of being selected for balancing the current class distribution.

Third, aiming at refraining the class boundaries of minority events from spreading further into the majority class area, a strict selection strategy should be applied to past minority samples that are close to the latest minority set. The dissimilarities between past minority samples and the current majority set should be evaluated. The clustering procedure based on DBSCAN [34] is used to discover sub-concepts of the current majority set. For each past minority event, the dissimilarity is transformed into the Mahalanobis distance from the nearest cluster center of the current majority set. Consequently, a past minority instance with a small distance from the current minority set but a large distance from the current majority set has a high probability of being recognized as a valuable data point for resampling the latest minority samples set.

It should be noted that sub-concepts of minority and majority samples of the latest chunk are discovered by the DBSCAN clustering. In the chunk-based framework, the clustering method operates the training events of consecutive blocks. The indexing structure based on k-d trees [35] is used to improve the efficiency of the DBSCAN clustering. The time consumption of clustering processes in the current block is $O(|P_m|\log|P_m| + |N_m|\log|N_m|)$. Thus, the upper bound of the time complexity of clustering phases in a block is $O(a\log a)$, where a is the size of training block. The processing time of DBSCAN in a block is limited since the value of a is small. Then, the overall runtime complexity of $O(ac\log a)$ is needed, where c is the number of chunks in a data stream. If the clustering method processes the entire data items at once, then $O(|S|\log|S|)$ time is required, where $|S|$ is the number of instances in a data stream. When handling high-volume data streams, GRE can significantly reduce the time consumption of clustering phases in a chunk-by-chunk manner ($O(ac\log a) \ll O(|S|\log|S|)$), which could satisfy the time requirements of data streams. Accordingly, we can use the k-d tree structure and operate instances in blocks to improve the efficiency of the DBSCAN clustering. In order to automatically detect two parameters (MinPts and Eps) required by DBSCAN, we use the k-distance graph [36].

The learning process of the selectively resampling mechanism in GRE can be visualized to gain a deeper understanding of its behavior, as shown in Fig. 2. Among all the data points, squares represent the minority instances collected from the previous w chunks, circles denote the instances in the current minority set, triangles are the samples in the current majority set, and stars are the centers of clusters. Fig. 2(a) shows that X_0 deviates from the current minority set and deeply locates in the majority class region, thus being regarded as an outlier. The similarity evaluation of common methods is seriously affected by outliers and sub-concepts as it treats a specific class of samples as a single cluster. In Fig. 2(b), DBSCAN is utilized to divide a specific class of instances into several sub-concepts. The data points that are closely packed together are grouped into a cluster, which makes outliers fall alone in the low-density regions and to be easily detected. Then, two clusters of the minority class and three clusters of the majority class are obtained. O_i $(1 \le i \le 5)$, which is marked using a star, is the center of the ith cluster. In Fig. 2(c), the similarity between a square and the latest minority set is based on the distance from its nearest minority class cluster. Meanwhile, the oversampling procedure considers dissimilarities between the collected minority data and the current majority set to overcome the class overlapping issue. For data points X_1 and X_2, the first cluster is their nearest cluster of the minority class, and thus d_{X_1} and d_{X_2} can be obtained. If d_{X_1} is equal to d_{X_2}, then X_1 and X_2 have the same probability of being selected, regardless of the underlying issue of class overlapping. Moreover, the fourth cluster is the closest one of the current majority set for X_1 and X_2 compared with other clusters. Consequently, X_2 has the priority to be selected if d'_{X_2} is larger than d'_{X_1}.

In UB [24], all the past minority samples are added into the current minority set, and thus the accumulated minority data may become a majority class. However, GRE applies the post-balance ratio f to proportionally select a certain number of past minority data. Therefore, the size of minority class can never surpass that of majority class in the amplified candidate block. After resampling the current minority set, the amplified data chunk is randomly divided into p sub-blocks, where each sub-block includes all the minority examples and a certain number of majority examples. The number of majority examples in each sub-block is designated according to the post-balance ratio f. Then, a candidate component classifier is built over a sub-block. Only limited minority events of past blocks are reused, which may fail to obtain a block with balanced class distribution. Thus, GRE needs to readjust the imbalance ratios of past chunks by updating previous hypotheses, which is described in Section 4.3.

4.3. Ensemble update mechanism

Oversampling techniques normally either collect all the previous minority samples or select only a portion of previous minority samples to obtain an amplified block with relatively balanced distribution. The former method may obtain enough minority samples of previous chunks, whereas drifting data are likely to be involved in the candidate block. In fact, only those minority data that are consistent with the target concept can actually be treated as valuable information for enhancing the predictive ability of candidate hypotheses. On the contrary, the latter strategy may not achieve a balanced chunk when the class distribution is extremely skewed as only limited minority events are absorbed into the current training data set.

Conservatively, the selectively resampling mechanism of GRE, which simultaneously considers concept drifts and data difficulty factors, only selects a certain number of minority events that are close to the current minority set and far from the current majority set to amplify the training sets of candidate hypotheses. Thus, the class distribution of the amplified block is likely to experience an imbalanced condition. To overcome this issue, resampling training minority sets of previous ensemble members using the current minority examples is an effective strategy. When updating previous hypotheses, all the minority instances and a majority class set are extracted from the most recent block. The number of selected majority examples should make class distributions of past amplified sub-blocks balanced.

Once balanced training sets of previous component classifiers have been achieved, the resampling process is no longer required. However, GRE also updates the corresponding hypotheses using the entire minority samples and an equal number of majority samples in the latest block. It is often assumed that the most recent block is the best representative of the current and near-future concepts. Therefore, updating previous hypotheses using the current data set can make the ensemble quickly adapt to new conditions, regardless of the types of drifts. In general, the predictive

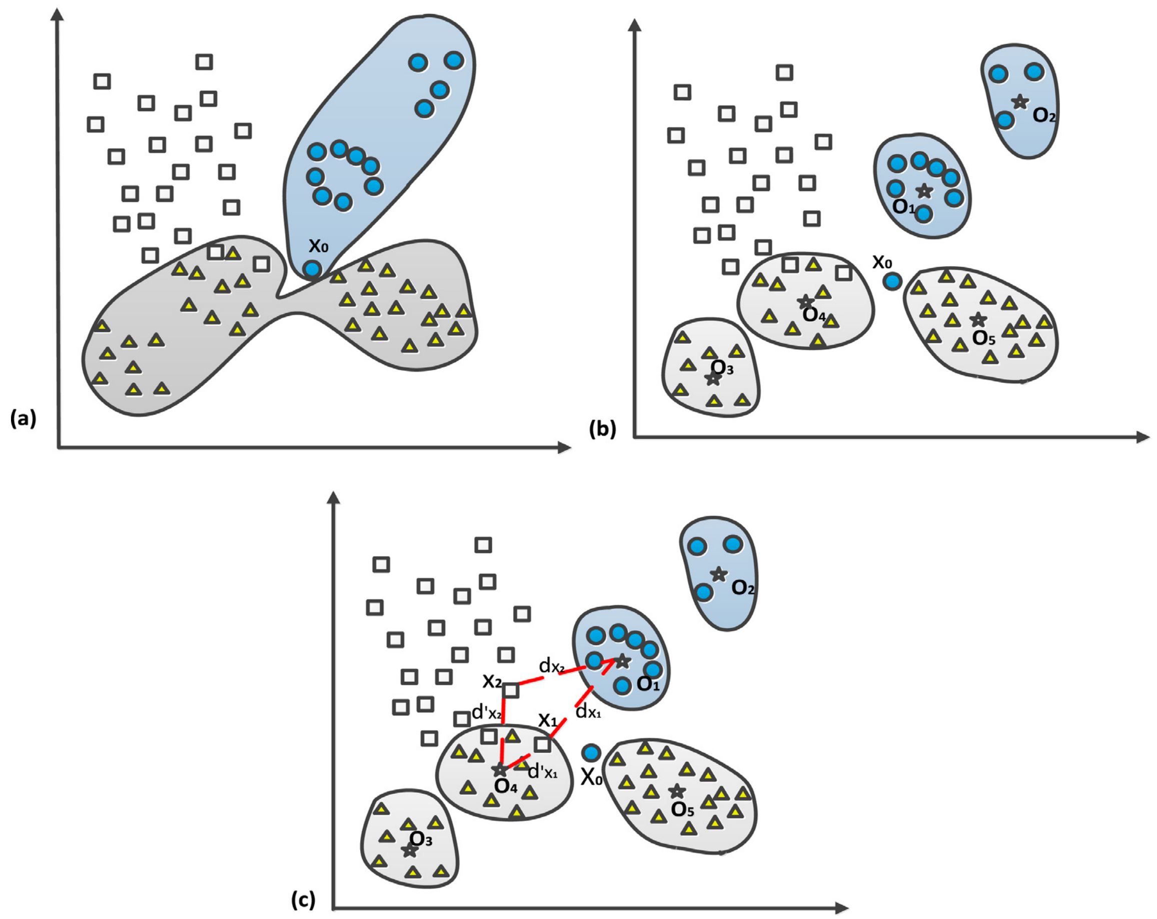


Fig. 2. The selectively resampling procedure. (a) Similarity evaluation of the common methods, (b) achieving clusters of the current majority and minority sets, and (c) similarity evaluation of GRE.

performances of chunk-based ensembles, such as MuSeRA [27] and AWE [17], largely rely on the predefined size of data chunks. On the contrary, the sizes of previous blocks are constantly enlarged in GRE. Therefore, the predefined chunk size could not have a considerable effect on the final decision (discussed in more detail in Section 5.4). As GRE is required to periodically update past members, the base learner should be an online classifier to be updated conveniently. VFDT [1] is a famous classifier for incrementally learning massive data streams, which is treated as the base model of GRE.

4.4. Weighting mechanism and final hypothesis

Ensemble-based classifiers can utilized past information of data streams to show better robustness and higher recognition ratios of examples than single classifiers. To adapt to a new condition, obsolete knowledge should be removed even if it might be relevant to the current concept in the future. Unfortunately, single classifiers cannot retrieve those data, thus resulting in the catastrophic forgetting. On the contrary, ensembles can hold the relevant information within ensemble members without storing a large amount of data. GRE maintains a fixed-size ensemble framework using a performance-based pruning technique to limit the time and memory usage. Each block can obtain a certain number of hypotheses. If the number of components is limited to k, then the candidate hypotheses should replace a part of past members. The predictions of component classifiers are aggregated using a weighted voting rule to make final decisions.

In this paper, we only consider the binary classification issue. The set of sample labels is $Y = \{+1, -1\}$, where $+1$ and -1 denote the labels of minority data and majority data, respectively. When evaluating the weights of component classifiers, an existing component that wrongly identifies a minority instance should be penalized more than the one that has mistaken a majority instance. For the candidate block B_m $(m > 1)$, a examples are selected as the training set S_m and the remaining events are used as the testing set T_m. $(x_i, y_i) \in S_m$ is the ith training example, where x_i is the vector of attribute values and y_i is its class label. The misclassification cost of x_i, denoted by C_i, is defined as follows:

$$C_i = \begin{cases} 1, & \text{if } y_i = +1 \\ r_m, & \text{otherwise} \end{cases} \tag{1}$$

where r_m is the imbalance ratio of the latest training block S_m. It describes the proportions between minority instances and majority instances, which is defined as

$$r_m = \frac{|P_m|}{|N_m|} \tag{2}$$

where $|P_m|$ and $|N_m|$ are the numbers of minority and majority instances in the current training block, respectively.

The candidate classifiers, denoted by K'_l $(l = 1, 2, \ldots, p)$, are trained over the most recent training data chunk and assigned weights as follows:

$$w_{K'_l} = \exp(\mathrm{MSE}_r) \tag{3}$$

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

The weight of candidate classifiers, denoted by $w_{K_l'}$, is proportional to MSE_r. MSE_r denotes the mean square error of a randomly predicted classifier, which is based on the class distribution of examples within the latest training block. Thus, the importances of candidate hypotheses only consider the factor of class distribution. A candidate component classifier with a relatively balanced training set can obtain a higher weight than others. At $t = 1$, k candidate components are created from examples within the first training block, where weights of all the ensemble members can be computed in Eq. (3).

For each incoming chunk B_m, $K_{t,j}$ denotes the jth component of the tth block. The weight of $K_{t,j}$, denoted by $w^m_{t,j}$, is determined by its predictive power on the training set in the most recent block B_m, which can be expressed by

$$w^m_{t,j} = \exp(\mathrm{MSE}_r - \mathrm{MSE}^m_{t,j}) \tag{5}$$

$$\mathrm{MSE}^m_{t,j} = \frac{1}{a}\sum_{i=1}^{a} C_i(1 - f^m_{t,j}(x_i))^2 \tag{6}$$

$\mathrm{MSE}^m_{t,j}$ is the mean square error of the jth hypothesis in the tth data block, which is evaluated over the examples in S_m. MSE_r, which is a preference value, is obtained in Eq. (4). For an example $(x_i, y_i) \in S_m$, $f^m_{t,j}(x_i)$ describes the probability of x_i being classified as category y_i by the jth hypothesis in the tth data block. C_i presented in Eq. (1) is the misclassification cost of x_i.

After obtaining all the weights of component classifiers, the final decision for a testing observation x_i' is determined as

$$\hat{y}_i = \arg\max_{x_i' \in T_m, y_i' \in Y}\left(\sum_{t=1}^{m-1}\sum_{j} w^m_{t,j} \times f(K_{t,j}(x_i'), y_i') + \sum_{l=1}^{p} w_{K_l'} \times f(K_l'(x_i'), y_i')\right) \tag{7}$$

in which $K_{t,j}(x_i')$ and $K_l'(x_i')$ are the predictive labels of x_i' using $K_{t,j}$ and K_l', respectively. y_i' and $\hat{y}_i$ are the true label and the predictive label of x_i', respectively. $f(K_{t,j}(x_i'), y_i')$ and $f(K_l'(x_i'), y_i')$ are the indicator functions and can be defined as follows:

$$f(K_{t,j}(x_i'), y_i') = \begin{cases} 1, & \text{if } K_{t,j}(x_i') = y_i' \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

$$f(K_l'(x_i'), y_i') = \begin{cases} 1, & \text{if } K_l'(x_i') = y_i' \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

It is worth pointing out that the weighting mechanism of candidate hypotheses as shown in Eq. (3) is different from that of the existing components as shown in Eq. (5). The weights of existing components should be continuously adjusted according to their predictive powers on the training examples in the latest block. However, if it is also applied to candidate classifiers, the cross-validation procedures are required, which is not suitable for handling high-speed data streams. The highest possible weight is directly given to the candidate classifiers, which especially makes GRE obtain good performances in the presence of sudden drifts. This setting is based on the assumption that the latest block provides the best representation of the current and near-future data distributions. The weighting formula presented in Eq. (3) does not consider a classifier's performance, and thus no validation sets are required.

5. Experiments

In this section, we empirically demonstrate the effectiveness of the proposed algorithm. GRE is compared with other six approaches proposed for learning concept drifts from imbalanced data on six synthetic datasets and one real-world dataset using multiple evaluation metrics.

5.1. Datasets

Table 2 presents the details of synthetic and real-world datasets used in our experiments. In the following section, we describe the procedures of each streaming data preparation.

5.1.1. Synthetic dataset

There is a shortage of suitable and public real-world datasets for evaluating data stream classification methods. For this reason, the synthetic datasets, in which concept drifts and imbalanced class distribution are involved, are particularly useful to verify whether the proposed algorithm can successfully address the joint issue. Compared with real-world datasets, the details of concept drifts on the synthetic datasets can be acquired in advance.

Two families of synthetic datasets are designed. On the one hand, one artificial two-class dataset called Square, which contains small disjuncts and sudden drifts, are generated according to [37]. Fig. 3 provides the structure of the Square dataset. First, we equally divide the unit square in a two-dimensional feature plane into 3×3 square grids. The input of each dimension is represented as three equal sized intervals and the contiguous intervals have different labels, but the middle one is kept empty. Second, three concepts, denoted by D_a, D_b, and D_c are designed. The inputs of D_a and D_b are in the interval [0, 1]. However, the input of D_c is in the interval [0, 3]. Third, we design a sudden drift by changing the sample labels within grids in addition to that of the middle one. Then, D_b replaces the concept D_a. Meanwhile, a concept drift can be generated by changing the range of samples attributes. Thus, a concept drift occurs if D_c replaces D_a. Concept drifts occur each 250,000 instances, then 4 concepts and 3 changes are on the Square dataset. The generating probability of majority examples is 19 times that of minority examples, and thus the imbalance ratio of Square is 1: 19.

On the other hand, synthetic datasets generated by data stream generators are leveraged to validate the effectiveness of the proposed method on a broad spectrum of concept drift scenarios. A brief description of each dataset created by the data stream generators is presented below.

SEA: the SEA generator [18] is leveraged to create three datasets, each containing 10% of noise. Three features are randomized in the interval [0,10], in which only two features are relevant. The sample labels are determined by comparing the sum of two relevant feature values with a predefined threshold. Then, concept drifts can be simulated by modifying the threshold. First, SEA_S contains three sudden drifts. Second, SEA_{SR} is designed to contain four sudden recurrent drifts. The fifth concept is the recurrent concept of the first one. Third, nine gradual drifts are introduced on the SEA_G dataset. We then induce class imbalance by undersampling one of the classes every 1000 observations, which ensures the minority data cardinality 5% of the total data size.

RanRBF: the random Radial Basis Function (RBF) generator creates new examples by selecting a center randomly, in which each center has a weight. The center with a high weight has a high probability of being selected. We use this generator to create the RanRBF_{GR} dataset of 791,000 instances described by 20 attributes and two classes. We simulate four gradual recurrent drifts by moving the centroids with constant speed. This dataset is particularly

Table 2
Datasets description.

Dataset	#Inst	#Attrs	#Classes	Noise	Imbalance ratio	#Drifts	Drift type
SEA_S	646k	3	2	10%	1:19	3	Sudden
SEA_G	541k	3	2	10%	1:19	9	Gradual
Hyper	100k	10	2	5%	1:19	1	Incremental
$RanRBF_{GR}$	791k	20	2	0%	1:19	4	Gradual recurrent
SEA_{SR}	664k	3	2	10%	1:19	4	Sudden recurrent
Square	1M	2	2	0%	1:19	3	Sudden
Elec	26k	8	2	–	1:19	–	–

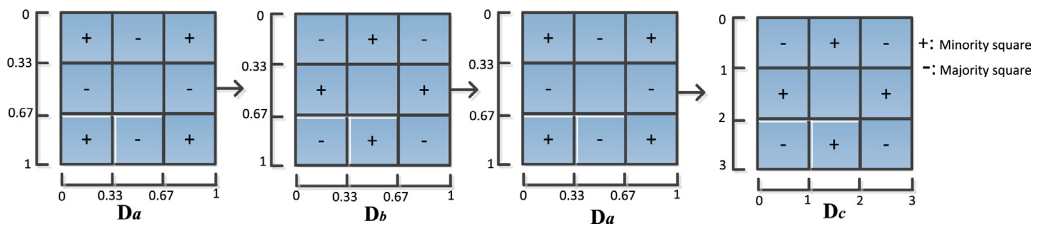


Fig. 3. Figure describing the Square dataset.

useful for analyzing an algorithm's ability to learn data in recurring environments. Then, we undersample one of the classes every 1000 observations. The cardinality of the minority class is set to 5% of total data.

Hyper: the Hyperplane generator [16] could simulate the incremental concept drifts by adjusting the orientation and the position of the rotating hyperplane smoothly. We use this generator to create the Hyper dataset that contains 100,000 observations describing by 10 attributes and two classes. An incremental drift is simulated through the modification weight that changes by 0.1 with every instance, and then 5% of noise is added to the data. One of the classes is undersampled every 1000 instances to create an imbalance ratio of 1: 19.

5.1.2. Real-world dataset

The Electricity Pricing dataset (Elec) was collected from the electricity market in New South Wales, Australian. The electricity prices are affected by the demand and the supply of electricity. Therefore, the natural occurrence of concept drifts has resulted in related problems [15,38]. The Elec dataset can be used to predict the fluctuating situation of electricity prices. To verify the ability of the proposed algorithm for learning imbalanced data, observations describing rising prices are undersampled to create an imbalance ratio of 1: 19 (minority data are 5% of the total data size).

5.2. Evaluation metrics

Several figures of merit can be used to evaluate the classification performances. Accuracy describes the total recognition performances of algorithms on testing observations, which is commonly used in traditional classification. Accuracy is primarily determined by the majority class, and thus it is not an adequate metric for imbalanced datasets.

Several metrics for evaluating classification performances in skewed mining problems have been put forward. First, the recall performance of the minority class is used for measuring the scaling of correctly classified minority events compared to the total size of minority data, which is often utilized to evaluate the recognition ability of algorithms on minority data. Second, the *G*-mean value is an important and commonly used metric for analyzing the imbalanced datasets, which integrates the majority class recall with the minority class recall. A high *G*-mean value indicates that the classifier performs equally well on examples of both classes. Third, *F*-measure combines precision and recall for measuring the performance of a classifier on the minority samples. Finally, AUC describes the area under the ROC curve. By adjusting the classification threshold, AUC provides a single average value of a classifier's performance. We exploit accuracy, recall, *F*-measure, *G*-mean, and AUC for analyzing the performances of tested algorithms from the different aspects in our simulation.

5.3. Experimental setup

We use several state-of-the-art algorithms to compare their ability to learn concept drifts from imbalanced data. All the tested algorithms were implemented in Java as part of the MOA framework [8]. The tested algorithms are listed as follows:

1. AWE [17]. AWE is strictly designed for concept drifts and included as a benchmark algorithm. In the chunk-by-chunk framework, each component classifier is trained over one chunk. Then, $k = 10$ ensemble members are preserved in the ensemble group.
2. SERA [25]. SERA only selects those minority data that are similar to the current minority class set to balance the current class distribution. The Mahalanobis distance is employed to measure such similarities. The post-balance ratio f is equal to 0.5. $k = 10$ ensemble members are built over the amplified training set in the latest block.
3. MuSeRA [27]. MuSeRA is proposed for learning imbalanced datasets in the concept drift scenario. Compared with SERA, the ensemble members of MuSeRA are derived from all the blocks rather than just the latest one. Thus, MuSeRA can utilize past information of data streams, which avoids forgetting knowledge excessively. All the ensemble members based on consecutive blocks are preserved, which is same as the setting of the paper's authors. Similarly, the similarity between each of past minority events and the current minority set is measured by the Mahalanobis distance. The sample size parameter, f, is set to 0.5.
4. SMOTE [32]. In the chunk-by-chunk framework, SMOTE is first used to balance the class distribution of the current block by creating the novel data, disregarding the distribution of majority examples. Then, a single classifier based on VFDT [1] is trained over a data chunk. The chunk size is equal to 1000 and the post-balanced ratio f is set to 0.5.
5. UB [24]. UB is employed to blindly propagate all the past minority examples into the latest block, regardless of their similarities of the current minority set. Thus, drifting data are likely

to be absorbed into the candidate block. $k = 10$ ensemble members are built over the candidate block.

6. OOB [33]. In OOB, the oversampling technique and time-decayed metrics are leveraged to overcome class imbalance and concepts drifts. The ensemble size is set to 10. The time-decayed class size can determine the resampling rate and the decay factor is set to 0.9 (same as suggested by the paper's author).

To make the comparisons more meaningful, we use VFDT [1] as the base classifiers for all the ensembles. The parameters of VFDT are adopted as default values of the MOA framework. The block sizes of all the chunk ensembles (e.g., AWE, SERA, MuSeRA, and UB) are 1000 for all the datasets. The minority examples of the previous $w = 50$ blocks are preserved in the set M. The post-balanced ratio f in our method is set to 0.5. For fair comparison, the GRE algorithm creates a new block every 1000 observations, and then $p = 1$ new hypothesis is trained over the amplified training set in the latest chunk. The number of component classifiers is set to $k = 10$. The effects of different chunk sizes and the number of candidate component classifiers on the performance of GRE are further discussed in Section 5.4.

When testing the algorithms, the holdout evaluation is leveraged. First, we divide the dataset into multiple equal sized chunks. Second, a total of 50% of the observations of each chunk are treated as the training set for training candidate component classifiers and evaluating weights of past ensemble members, and the remaining events of each chunk are regarded as testing data. Third, we periodically apply the current decision models to the testing data of the most recent block. Once all the instances in a testing chunk are tested, we computed the selected evaluation metrics over the periodical holdout sets, i.e., consecutive chunks of testing examples.

In the comparative experiment, the Friedman test [39] is applied to make a formal statistical analysis of the tested classifiers over multiple data sets based on each evaluation metric. Then, the Wilcoxon signed rank test [39] is used to compare GRE with other comparative algorithms in a pairwise manner. Through ranking all the results of comparative algorithms, we can derive the average rank of a tested algorithm in terms of an evaluation measure across all datasets. Many studies have shown that capturing the dynamic characteristics of streaming data is more meaningful than obtaining a final result in data stream mining [40]. We generate graphical plots for each dataset based on all evaluation metrics. Meanwhile, the average performances of comparative algorithms that cover all the time steps are included in the tables.

5.4. Study on the components of the GRE algorithm

In this section, we analyze the effects of different block sizes and the number of candidate classifiers on the performance of our proposed algorithm. In general, the performances of chunk-based ensembles, such as MuSeRA [27] and AWE [17], rely on the predefined size of data chunks. Using big size blocks is likely to contain concept drifts within data chunks, thus leading to the poor performances of component classifiers. On the contrary, using small size chunks may damage the performances of ensemble members in the stationary condition as the number of training examples of each component classifier is limited. This situation will become more serious when the minority examples are underrepresented in the class-imbalance scenario. In GRE, the ensemble can constantly amplify the previous training data chunks by updating previous hypotheses using examples of the most recent block. Consequently, the predefined size of data chunks could not have a significant effect on the predictive ability of GRE.

Table 3 provides the AUC performances and average ranks of GRE on all the selected datasets when using $d \in \{500, 800, 1000, 1250, 2000\}$. All the results are the averages of 30 independent runs and the best result is highlighted with bold-face type. Then, we can observe that differences in each row are small and no global dependency on d can be seen. These observations are further confirmed by the statistical test. Performing a Friedman test [39] on the calculated deviations for $d \in \{500, 800, 1000, 1250, 2000\}$, thus $F_{F_d} = 0.535$ can be calculated. If the selected significant level is equal to 0.5, then we can conclude that the performance of GRE is robust against different block sizes. In the comparative experiment, the chunk size of GRE is equal to 1000, which is consistent with the settings of other chunk ensembles for fair comparisons.

Table 3
AUC values (%) and average ranks of GRE using different chunk sizes d.

	d=500	d=800	d=1000	d=1250	d=2000
SEA$_S$	**87.28(1)**	87.00(3)	86.97(4)	87.14(2)	86.68(5)
SEA$_G$	88.99(2)	88.87(5)	**89.06(1)**	88.95(4)	88.97(3)
Hyper	99.07(5)	99.28(4)	99.31(2)	**99.40(1)**	99.29(3)
RanRBF$_{GR}$	98.86(4)	**99.01(1)**	98.98(3)	98.99(2)	98.85(5)
SEA$_{SR}$	87.28(2)	**87.38(1)**	87.17(4)	87.15(5)	87.20(3)
Square	98.59(2)	98.54(4)	**98.64(1)**	98.57(3)	98.53(5)
Elec	73.57(3)	71.71(5)	74.87(2)	71.79(4)	**76.06(1)**
Average rank	2.71	3.29	2.43	3.00	3.57

The best result is in boldface.

Table 4
AUC values (%) and average ranks of GRE using different values of p.

	p=1	p=2	p=4	p=6	p=8
SEA$_S$	**86.97(1)**	86.86(2)	85.51(3)	84.22(4)	82.55(5)
SEA$_G$	**89.06(1)**	88.61(2)	87.49(3)	86.24(4)	84.58(5)
Hyper	**99.31(1)**	99.18(2)	98.80(3)	98.29(4)	96.31(5)
RanRBF$_{GR}$	**98.98(1)**	98.82(2)	98.09(3)	96.64(4)	94.37(5)
SEA$_{SR}$	**87.17(1)**	86.76(2)	84.99(3)	83.45(4)	82.17(5)
Square	98.64(2)	**98.70(1)**	98.60(3)	97.57(4)	95.78(5)
Elec	**74.87(1)**	74.22(4)	74.34(3)	74.70(2)	73.62(5)
Average rank	1.14	2.14	3.00	3.71	5.00

The best result is in boldface.

Apart from studying the effect of d, we also analyze the impact of the number of candidate component classifiers on the predictive performance of the GRE algorithm. The p candidate ensemble members are built over the latest training data block. Then, k ensemble members are preserved in the ensemble group. Using a large value of p will remove a large number of previous ensemble members. Thus, GRE only have limited information of a data stream to train an ensemble model. On the contrary, using a small value of p can help in preserving enough information of past data chunks. Therefore, GRE using a small value of p can make full use of relevant knowledge of a data stream and could have good generalization performance.

Table 4 provides the AUC performances and average ranks of GRE using different values of p. Each result is the average of 30 independent runs and the best result is highlighted with bold-face type. We perform a Friedman test [39] on the calculated deviations for $p \in \{1, 2, 4, 6, 8\}$. Then, $F_{F_p} = 38.23$ is obtained. Thus, we can conclude that significant differences exist among the AUC performances of GRE using different settings of p under the significant level $\alpha = 0.5$. Then, we perform the Bonferroni–Dunn post-hoc test [39] to compare GRE in terms of $p = 1$ with others. The critical difference (CD) for $\alpha = 0.5$ is equal to 0.302. Therefore, the AUC performance of GRE using $p = 1$ is significantly better than that of $p = 2$, $p = 4$, $p = 6$, and $p = 8$. In the comparison experiment, $p = 1$ is selected as the default value.

Table 5
Performance comparison of different algorithms on all datasets.

Data sets	Methods	Accuracy	*F*-measure	*G*-mean	Recall	AUC
SEA$_S$	AWE	**94.97(1)**	7.95(7)	16.81(7)	4.57(7)	**87.09(1)**
	SERA	90.97(4)	27.30(5)	49.54(5)	29.38(6)	70.13(7)
	MuSeRA	93.64(2)	31.76(4)	53.59(4)	31.98(5)	85.37(4)
	SMOTE	90.01(5)	41.37(3)	79.22(3)	69.48(4)	83.96(6)
	UB	16.80(7)	12.30(6)	23.10(6)	**97.39(1)**	85.21(5)
	OOB	88.62(6)	42.75(2)	80.47(2)	72.52(3)	85.86(3)
	GRE	91.13(3)	**46.04(1)**	**81.72(1)**	72.95(2)	86.97(2)
SEA$_G$	AWE	**94.96(1)**	3.52(7)	8.10(7)	2.01(7)	88.48(3)
	SERA	92.56(3)	26.00(5)	49.66(5)	29.14(6)	71.23(7)
	MuSeRA	92.87(2)	27.50(4)	51.13(4)	29.21(5)	87.15(5)
	SMOTE	87.36(5)	37.46(3)	80.59(3)	75.03(4)	85.64(6)
	UB	27.47(7)	14.00(6)	35.81(6)	**96.64(1)**	87.44(4)
	OOB	85.23(6)	37.67(2)	82.01(2)	82.38(2)	88.60(2)
	GRE	88.64(4)	**41.21(1)**	**82.75(1)**	77.52(3)	**89.06(1)**
Hyper	AWE	95.56(2)	20.01(7)	31.57(7)	11.99(7)	90.71(4)
	SERA	92.66(4)	33.21(6)	56.61(6)	36.31(6)	72.46(7)
	MuSeRA	93.91(3)	39.53(5)	61.85(5)	41.48(5)	84.34(6)
	SMOTE	90.75(6)	42.61(4)	75.57(4)	63.02(4)	84.57(5)
	UB	87.28(7)	47.45(3)	89.57(3)	**92.76(1)**	94.26(3)
	OOB	91.64(5)	59.84(2)	89.83(2)	88.18(3)	96.40(2)
	GRE	**97.81(1)**	**87.30(1)**	**93.45(1)**	88.97(2)	**99.31(1)**
RanRBF$_{GR}$	AWE	97.03(3)	53.98(4)	59.64(5)	41.98(5)	91.05(3)
	SERA	95.41(4)	38.28(5)	50.70(6)	30.00(6)	70.16(7)
	MuSeRA	92.69(6)	10.26(7)	21.51(7)	8.51(7)	79.23(6)
	SMOTE	92.86(5)	54.05(3)	79.17(4)	67.72(4)	85.99(5)
	UB	70.81(7)	28.43(6)	81.76(3)	**92.50(1)**	90.69(4)
	OOB	97.49(2)	83.08(2)	90.31(2)	88.31(2)	98.09(2)
	GRE	**99.03(1)**	**90.30(1)**	**91.86(1)**	85.20(3)	**98.98(1)**
SEA$_{SR}$	AWE	**94.95(1)**	9.79(7)	19.42(7)	5.80(7)	**87.58(1)**
	SERA	92.92(3)	26.68(5)	48.91(5)	28.32(6)	70.10(7)
	MuSeRA	93.72(2)	31.88(4)	53.48(4)	31.54(5)	85.14(5)
	SMOTE	89.95(5)	41.29(3)	78.87(3)	68.87(4)	83.67(6)
	UB	17.90(7)	12.61(6)	23.34(6)	**97.22(1)**	85.46(4)
	OOB	89.37(6)	44.08(2)	82.65(2)	80.00(2)	86.72(3)
	GRE	91.85(4)	**48.26(1)**	**83.25(1)**	75.09(3)	87.17(2)
Square	AWE	95.03(2)	0(7)	0(7)	0(7)	80.95(4)
	SERA	88.73(5)	16.76(5)	37.21(5)	19.81(5)	60.14(6)
	MuSeRA	92.89(3)	1.54(6)	6.25(6)	1.81(6)	51.53(7)
	SMOTE	90.48(4)	31.22(3)	59.53(3)	40.77(4)	76.59(5)
	UB	33.88(7)	18.19(4)	39.28(4)	**98.79(1)**	88.76(3)
	OOB	82.13(6)	52.98(2)	81.88(2)	85.46(3)	91.45(2)
	GRE	**95.37(1)**	**76.00(1)**	**94.33(1)**	94.49(2)	**98.64(1)**
Elec	AWE	87.91(3)	26.18(5)	48.74(5)	37.60(5)	68.76(4)
	SERA	81.93(6)	18.12(6)	34.67(6)	36.80(6)	62.57(7)
	MuSeRA	**95.00(1)**	0(7)	0(7)	0(7)	66.51(5)
	SMOTE	82.72(5)	28.42(2)	62.69(2)	53.07(2)	73.42(3)
	UB	84.99(4)	27.85(3)	53.91(4)	40.80(3)	65.89(6)
	OOB	76.85(7)	27.24(4)	**62.87(1)**	**63.26(1)**	74.52(2)
	GRE	89.92(2)	**31.78(1)**	60.17(3)	40.48(4)	**74.87(1)**

The best result for each dataset and criteria is highlighted in bold.

5.5. Comparative with other algorithms

This section compares GRE with six state-of-the-art methods on six synthetic datasets and one real-world dataset. We generated graphical plots for each dataset describing algorithms's performances in terms of accuracy, *F*-measure, *G*-mean, recall, and AUC across all the time steps. Due to the number of comparisons (seven algorithms on five evaluation metrics), we split the results into two groups. On the one hand, we compare GRE with the chunk-based ensembles proposed for learning imbalanced data streams (SERA, UB, and MuSeRA). On the other hand, GRE is compared with other tested algorithms (AWE, SMOTE, and OOB). Due to space limitations, we only provide the performance curves of tested algorithms on two representative datasets (SEA$_S$ and RandRBF$_{GR}$). Meanwhile, other results are included in Table 5.

Before we present the detailed analysis of the experimental results, we first provide the major observations from the algorithms's performances on all data sets. Table 6 provides the a summary of mean ranks of algorithms on all the datasets considering each evaluation metric. First, AWE obtains the best mean rank in terms of accuracy attribute to maintaining good performance on majority data (followed by GRE). However, the boost in accuracy for AWE causes a large drop in minority class recall, *G*-mean, and *F*-measure. This is because AWE lacks a mechanism to accommodate imbalance data. In particular, AWE misclassifies all the minority examples as majority examples when the complex data

Table 6
Mean ranks of seven comparative methods on all data sets.

Rank	AWE	SERA	MuSeRA	SMOTE	UB	OOB	GRE
Accuracy	1.86	4.14	2.71	5.00	6.57	5.43	2.29
F-measure	6.29	5.29	5.29	3.00	4.89	2.29	1.00
G-mean	6.43	5.57	5.29	3.14	4.57	1.86	1.29
Recall	6.43	5.71	5.86	3.71	1.29	2.29	2.71
AUC	2.86	6.86	5.43	5.14	4.14	2.29	1.29

distribution are involved on the Square dataset. Second, UB has the best mean rank for recall, especially on the datasets of big data sizes (e.g., all synthetic datasets). This is not surprising, because UB oversamples the current minority set by accumulating all the preserved minority events of past blocks. The accumulated minority data may become a majority class when the data size is large enough. Therefore, the majority class recall is calculated for UB when the size of minority class surpasses that of majority class. However, the boost in recall for UB comes at the cost of accuracy, *F*-measure, *G*-mean, and AUC. It is should be noted that GRE applies the post-balance ratio f to proportionally select a certain number of minority examples in the previous blocks. Thus, the number of minority samples can never surpass that of majority samples in the candidate block for GRE. Third, MuSeRA can maintain all the knowledge of data streams by preserving all the ensemble members built over consecutive blocks. However, all the component classifiers of SERA are trained over the latest chunk. Thus, MuSeRA outperforms SERA in terms of accuracy (primarily determined by majority class accuracy). Fourth, SMOTE maintains a better mean rank than that of AWE in terms of recall by generating novel minority data. OOB typically provides a good mean rank for *F*-measure, AUC, recall, and *G*-mean by resampling and time-decayed metrics. However, the boost in performance on minority events for OOB comes at the cost of accuracy. Finally, we can observe that GRE ranks the best in terms of *F*-measure, *G*-mean, and AUC. Meanwhile, it maintains the second and the third mean rank in terms of accuracy and recall, respectively. The selectively resampling technique in GRE, which simultaneously considers the data difficulty factors and concept drifts, is applied to improve the recognition ratio of models on minority examples. Through updating previous ensemble members using the current observations, GRE quickly reacts to different kinds of concept drifts and maintains good performance on majority examples. Therefore, we can conclude that GRE can provide good performance on minority examples, without sacrificing the performance on majority examples.

Figs. 4 and 5 present the results on the SEA$_S$ dataset. The average values of all figures of merit used in the evaluation are included in Table 5. There are several observations we can make from these results. First, we can observe that most algorithms experience a performance drop when sudden drifts occur. Second, SERA and SMOTE maintain stable levels of the accuracy performances for nearly all time steps. This is because SERA does not need to use component classifiers built over previous blocks. Meanwhile, SMOTE builds a single classifier over the latest data chunk. Thus, there is no prior knowledge of major class in the training sets of SERA and SMOTE. Third, the accuracy performance of AWE always maintains a high level compared with other tested algorithms (closely followed by MuSeRA and GRE), but primarily due to its performance on the majority class. The boost in accuracy for AWE comes at the cost of *F*-measure, *G*-mean, and recall. This is because AWE lacks a mechanism for handling imbalanced data. Meanwhile, MuSeRA does well on accuracy compared with SERA by maintaining all the component classifiers built over consecutive blocks. Fourth, while UB has the best rank for recall (closely followed by GRE); it performs rather poorly on other figures of merit. This is a common trend for UB on all the synthetic datasets. Finally, GRE is competitive with other algorithms in terms of *F*-measure and *G*-mean. The *F*-measure and *G*-mean performances of GRE can quickly recover from a sudden change and maintain high levels during the steady state periods. Meanwhile, GRE maintains the second rank in terms of recall and AUC.

On the dataset with gradual drifts (SEA$_G$), we observe several trends from Table 5, which appear to be same with those observed on the SEA$_S$ dataset. First, AWE has the best accuracy performance, but primarily due to its good predictive ability on majority class. However, AWE performs rather poorly with the worst rank in terms of *F*-measure, *G*-mean, and recall. Second, UB outperforms other algorithms in terms of recall, which comes at the cost of accuracy, *F*-measure, and *G*-mean. Third, GRE has the best rank for *F*-measure and *G*-mean. On the SEA$_G$ dataset, we can observe that GRE is the top ranking algorithm in terms of the AUC performance. Through a resampling technique and time-decayed metrics, OOB provides good performance in terms of *F*-measure, *G*-mean, recall, and AUC, but performs rather poorly on accuracy. On the contrary, GRE can provide a good overall balance in accuracy, *F*-measure, *G*-mean, recall, and AUC.

For the dataset with an incremental drift (Hyper), GRE performs consistently well on all the evaluation metrics. Table 5 provides the mean values of all figures of merit used in the evaluation on the Hyper dataset. On the one hand, GRE ranks the best in terms of accuracy, *F*-measure, *G*-mean, and AUC. On the other hand, UB is the top-ranking algorithm in terms of recall, closely followed by GRE. As in the previous experiments, UB ranks the best in terms of recall but ranks the lowest in terms of accuracy. Meanwhile, AWE has the worst rank in terms of *F*-measure, *G*-mean, and recall.

Figs. 6 and 7 present the results on the RanRBF$_{GR}$ dataset. Recurring concepts were introduced to validate whether the tested algorithms can utilize prior knowledge of data streams to improve the performance of algorithms at the current time. Several observations should be noticed from Table 5. First, SMOTE, UB, and SERA build models over the latest chunk, thus cannot reactivate past hypotheses. We can observe that the performances of them fail to enhanced when the environment reoccurs. Second, we observe, again, the boost in recall for UB comes at the cost of the overall accuracy. UB maintains the best performance in terms of recall, followed by OOB and GRE. Third, GRE maintains a series of past ensemble members to make a decision for each testing event, which obtains the best rank for accuracy, *F*-measure, *G*-mean, and AUC.

On the dataset with sudden recurrent drifts (SEA$_{SR}$), the ordered sequence consists of five concepts, in which the fifth concept is the recurrent concept of the first one. There are several observations, which are consistent with those observed in the SEA$_S$ and SEA$_G$ experiments, as shown in Table 5. First, lacking a mechanism to accommodate imbalanced data, AWE has the best accuracy performance but ranks the lowest in terms of *F*-measure, *G*-mean, and recall. Second, UB outperforms other algorithms in terms of recall, which causes a large drop in *F*-measure, *G*-mean, and accuracy. Third, GRE provides a significant improvement in *F*-measure and *G*-mean compared to other algorithms.

Apart from multiple kinds of concept drifts, we also analyze the effect of small disjuncts on the performances of tested algorithms. Sudden drifts and small disjuncts are involved on the Square dataset. We do observe from Table 5 that AWE misclassifies all the minority examples as majority class. Thus, AWE performs rather poorly on *F*-measure, recall, and *G*-mean. As in previous experiments, UB'ss strong recall performance comes at the cost of accuracy. OOB maintains good performance in terms of *F*-measure, *G*-mean, and AUC. However, it drops to rank 6 for accuracy. It should be noted that the goal of learning imbalanced data is to improve the recognition ratio on minority examples, without sacrificing the performance on majority examples. The *F*-measure, *G*-mean, AUC, and accuracy performances of GRE all rank the best on the Square dataset. Meanwhile, GRE maintains the second rank for minority class recall.

On the Elec dataset, the average values of selected evaluation metrics for all comparative algorithms are included in Table 5. We have converted it to an imbalanced learning problem. A few observations: first, MuSeRA ranks the best in terms of accuracy, followed by GRE. However, MuSeRA fails to identify any minority samples, which performs particularly poorly with the worst rank in terms of *F*-measure, *G*-mean, and recall. Second, while OOB outperforms other algorithms in terms of minority class recall and

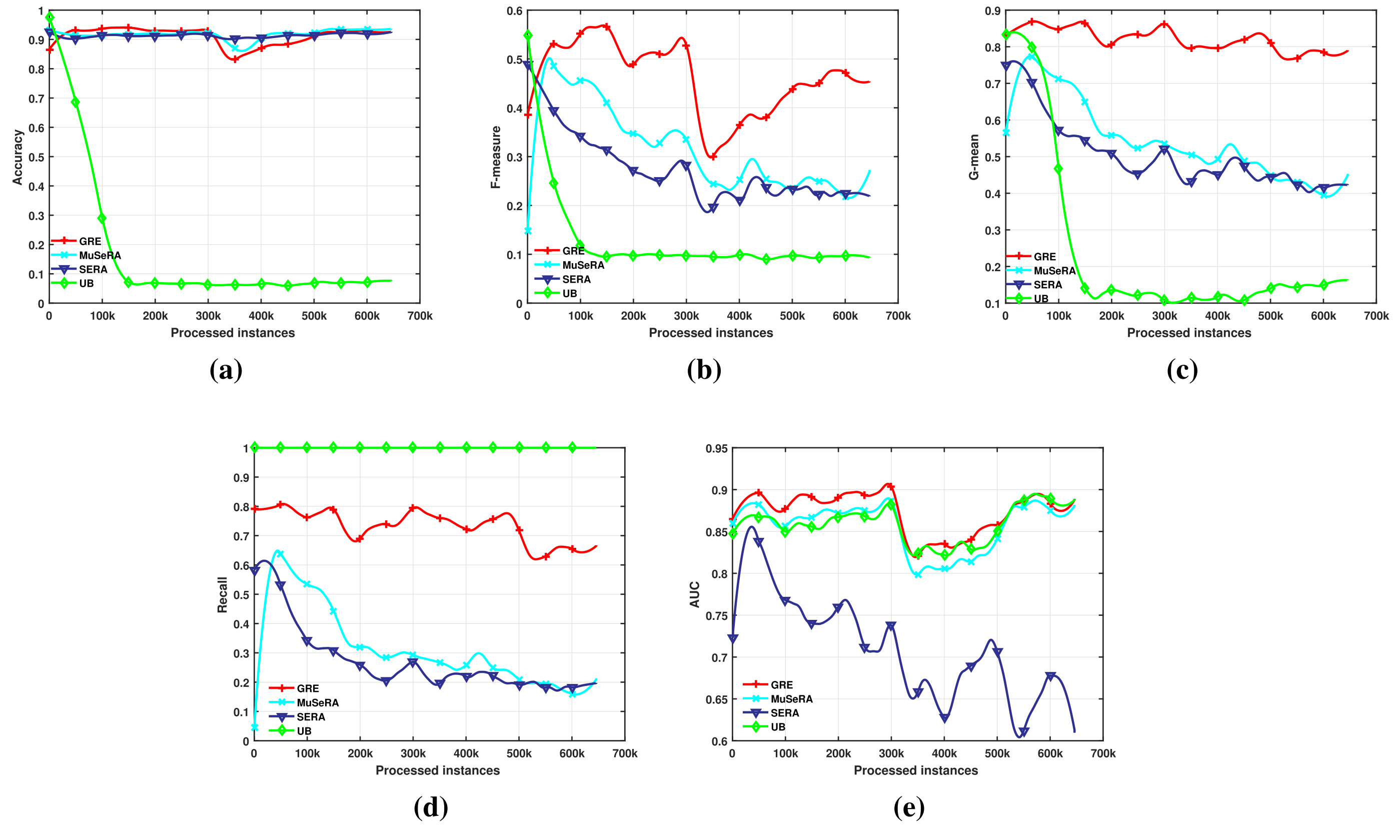


Fig. 4. The first group algorithms comparison on SEA$_S$. (a) Accuracy, (b) *F*-measure, (c) *G*-mean, (d) Recall, and (e) AUC.

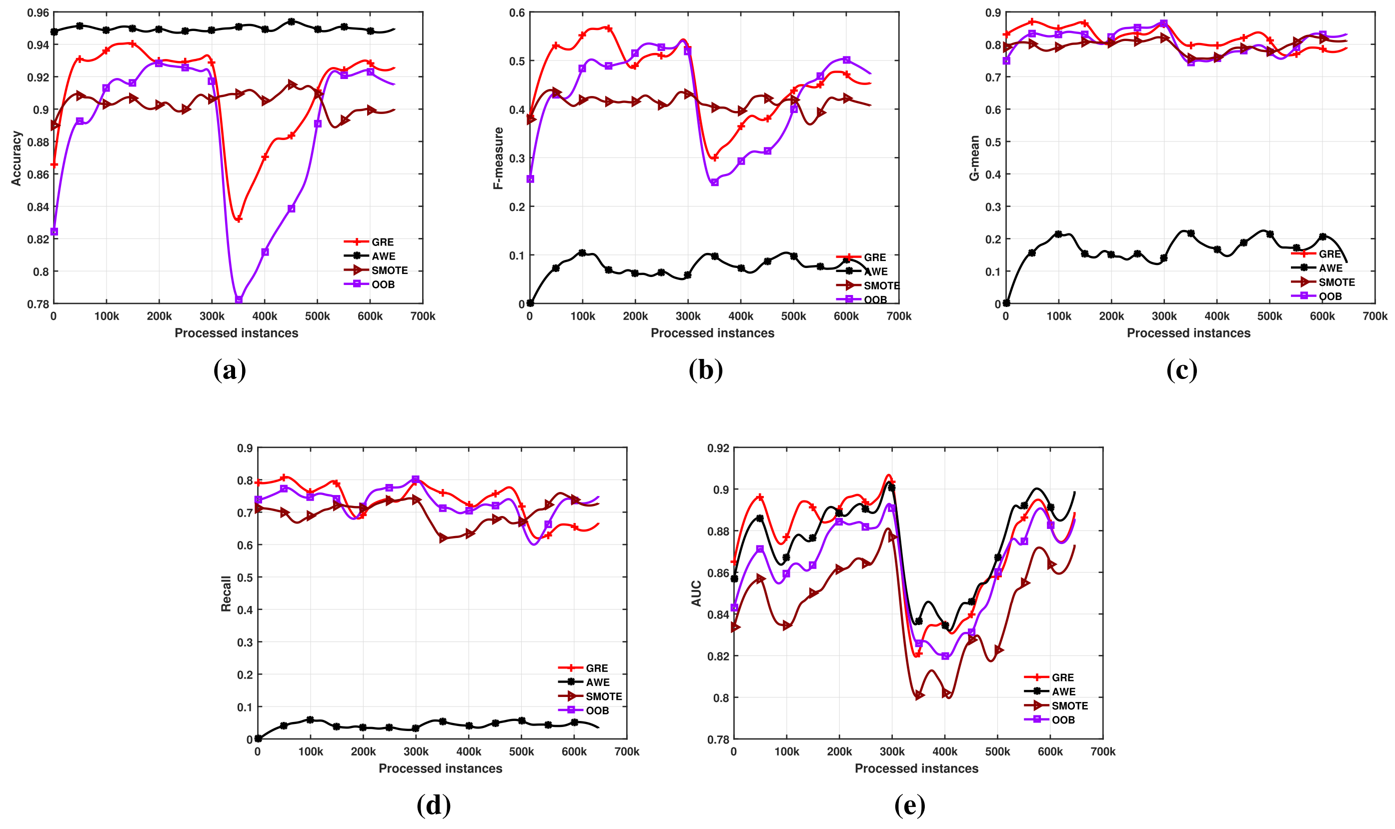


Fig. 5. The second group algorithms comparison on SEA$_S$. (a) Accuracy, (b) *F*-measure, (c) *G*-mean, (d) Recall, and (e) AUC.

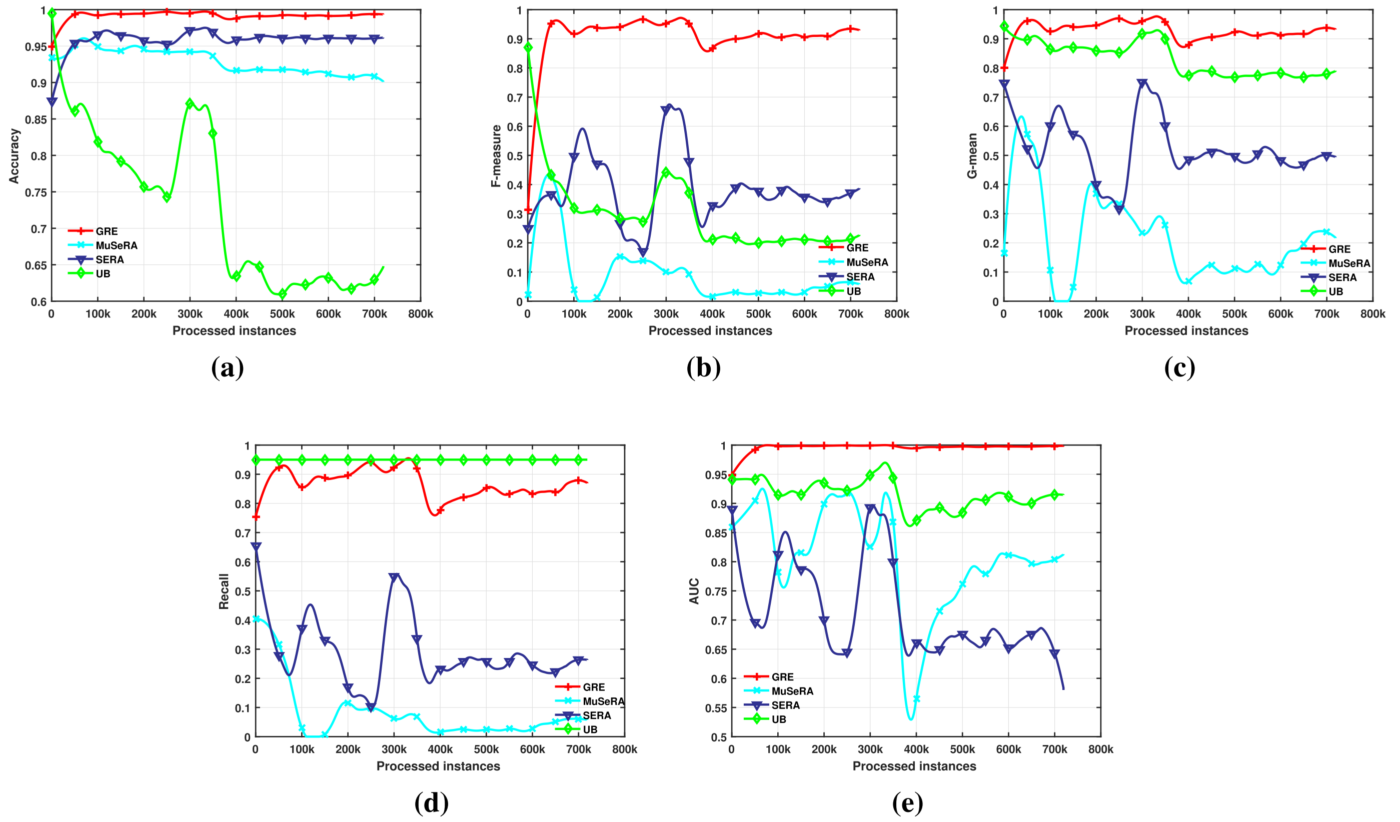


Fig. 6. The first group algorithms comparison on RanRBF$_{GR}$. (a) Accuracy, (b) *F*-measure, (c) *G*-mean, (d) Recall, and (e) AUC.

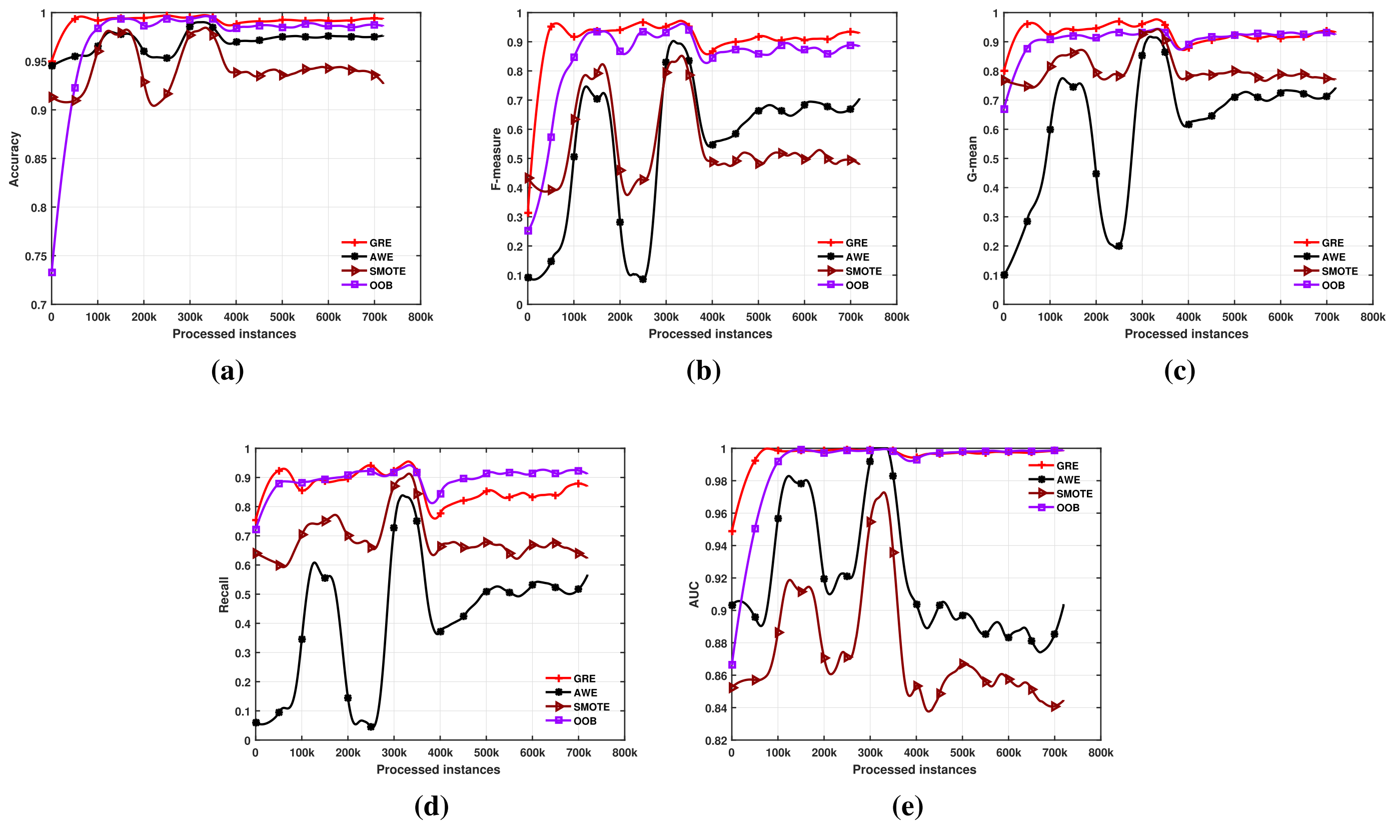


Fig. 7. The second group algorithms comparison on RanRBF$_{GR}$. (a) Accuracy, (b) *F*-measure, (c) *G*-mean, (d) Recall, and (e) AUC.

Table 7
Friedman test with the corresponding post-hoc test, Bonferroni–Dunn for seven comparative methods on all data sets.

	Friedman	AWE	SERA	MuSeRA	SMOTE	UB	OOB
F-measure	Reject	**5.29**	**4.29**	**4.29**	2.00	**3.89**	1.29
G-mean	Reject	**5.14**	**4.28**	**4.00**	1.85	**3.28**	0.57
AUC	Reject	1.57	**5.57**	**4.14**	**3.85**	2.85	1.00

A value greater than the CD (CD = 3.046) indicates statistically significant differences between the methods, which are highlighted in boldface.

Table 9
Runtime of the compared approaches (s).

	AWE	SERA	MuSeRA	SMOTE	UB	OOB	GRE
SEA$_S$	15.08	9.23	132.00	8.14	277.00	10.58	14.84
SEA$_G$	13.55	7.31	94.00	6.64	149.00	8.39	10.12
Hyper	8.92	3.00	24.16	3.34	6.20	7.89	7.08
RanRBF$_{GR}$	66.00	25.30	1328.00	22.83	585.00	35.75	39.00
SEA$_{SR}$	14.39	9.27	130.00	7.42	292.00	10.11	10.89
Square	18.59	13.89	244.00	11.03	966.00	15.03	19.53
Elec	0.52	0.22	0.25	1.00	0.44	1.05	0.80

G-mean; it drops to rank 7 for accuracy. Third, the size of minority class is not easy to exceed that of majority class because the data size of the Elec dataset is much smaller than that of other datasets. Thus, UB performs well in terms of accuracy but obtains relatively poor recall performance compared with its performances on other datasets. Finally, GRE has the best performance in terms of *F*-measure and AUC. Meanwhile, it provides the second rank for the accuracy performance.

5.6. Statistical analysis of results

To extend the analysis provided above, we conduct statistical tests for validating the effectiveness of GRE in terms of *F*-measure, *G*-mean, as well as AUC. First, when different algorithms provide varying performances on different data sets, the Friedman test [39] is leveraged to verify whether there is a significant difference among the mean ranks of different alternatives. The Friedman test is a nonparametric statistical method, where the null hypothesis assumes no significant differences among the mean ranks of different algorithms. Table 6 provides a summary of mean ranks of comparative algorithms on all datasets based on each selected figure of merit. Table 7 presents the results of Friedman tests and post-hoc tests on *F*-measure, *G*-mean, and AUC for comparative algorithms over all the datasets. If the significant level is selected as 0.05, then the null hypotheses in terms of all three metrics can be rejected. Each value in Table 7 is the difference of the mean ranks between two algorithms. To verify whether GRE performs better than other algorithms in terms of *F*-measure, *G*-mean, and AUC, we compute the critical difference (CD) chosen by the Bonferroni–Dunn post-hoc test. If the difference between the mean ranks of two algorithms in terms of an evaluation metric is greater or equal to CD, then we can state that there is a statistical difference between the two algorithms. As CD=3.046, the *F*-measure performance of GRE is significantly better than that of AWE, SERA, MuSeRA, and UB. The *G*-mean performance of GRE is significantly better than that of AWE, SERA, MuSeRA, and UB. Meanwhile, GRE performs significantly better than SERA, MuSeRA, and SMOTE in terms of AUC.

In addition, we perform the Wilcoxon singed rank test [39] to compare GRE with the remaining algorithms in terms of *F*-measure, *G*-mean, and AUC. Table 8 provides Wilcoxon's test results. First, the *p*-values derived from this test in terms of *F*-measure are: p_{SMOTE}=0.0078 and p_{OOB}=0.0078. Second, the *p*-values resulting from this test in terms of *G*-mean are: p_{SMOTE}=0.0391 and p_{OOB}=0.0781. Third, we can state that GRE significantly outperforms the remaining algorithms in terms of the AUC performance ($p_{\mathrm{AWE}} = 0.0391$, $p_{\mathrm{UB}} = 0.0078$, and $p_{\mathrm{OOB}} = 0.0078$). These results show that GRE significantly outperforms other algorithms in terms of *F*-measure, *G*-mean, and AUC. Meanwhile, we do observe from Table 6 that GRE maintains the second and the third mean rank for accuracy and recall, respectively.

In an imbalanced data conditions, we aims to design a classifier that has the best balance in accuracy, *F*-measure, *G*-mean, recall, and AUC. In general, a classifier has difficulty in performing best on all the evaluation metrics. From the above analysis, we can state that GRE obtains a good tradeoff between the majority class and minority class performances when tested on data streams with multiple kinds of concept drifts.

5.7. Running time efficiency

In this section, we discuss the running time efficiency of the comparative algorithms. Table 9 displays the time consumption of comparative approaches on all the datasets. Each result is the average of 30 independent runs. The hardware configuration used for simulation is an Intel Core i7 Processor with 8 GB RAM.

There are several observations we can make from these results. First, SMOTE generally needs relatively small amount of time. This is because only one classifier is generated to predict labels of testing observations. Second, UB consumes much time than SERA over all the datasets. After resampling the current minority set, SERA and UB build all the ensemble members over the amplified training chunk. However, UB selects all previously minority observations and SERA limits the number of accepted minority observations proportional to the size of the current majority set. Thus, UB increases the time cost of the algorithm since over-time more examples are involved in its training, especially on the datasets with large data sizes (e.g., Square and RanRBF$_{\mathrm{GR}}$). Third, MuSeRA maintains all the ensemble members trained over consecutive data chunks without the pruning procedure. Because a large number of component classifiers are involved in the ensemble group, MuSeRA performs rather poorly in terms of time consumption. Finally, we can observe that GRE does not consume too much time compared with other tested algorithms. Thus, GRE has a satisfactory time efficiency, which is suitable for mining data streams.

6. Conclusion remarks

In this paper, we discuss the classification techniques for handling the combined problem of concept drifts and class imbalance. While each of these two issues has been well researched, the joint issue is still underexplored even though it has received increasing attention. We propose an ensemble-based approach called GRE

Table 8
Wilcoxon's test results for the comparison of GRE versus the remaining methods on all data sets.

F-measure		*G*-mean		AUC	
Methods	*p*-Values	Methods	*p*-Values	Methods	*p*-Values
GRE vs. SMOTE	0.0078	GRE vs. SMOTE	0.0391	GRE vs. AWE	0.0391
GRE vs. OOB	0.0078	GRE vs. OOB	0.0781	GRE vs. UB	0.0078
				GRE vs. OOB	0.0078

for learning different kinds of concept drifts from imbalanced data. Three major contributions have been made.

First, a selectively resampling mechanism, which simultaneously considers concept drifts and data difficulty factors, is applied to balance the current class distribution by reusing preserved minority data of past chunks. Compared with existing resampling techniques proposed for handling imbalanced data streams, GRE can avoid absorbing drifting data by evaluating the similarity between each of preserved minority examples and the current minority set. Meanwhile, DBSCAN is utilized to discover sub-concepts of the majority set and the minority set, respectively. Thus, the similarity evaluation of GRE is not affected by outliers and small disjuncts. Furthermore, the dissimilarities between the preserved minority examples and the current majority set are considered to avoid the class overlapping between different classes.

Second, the ensemble update mechanism can periodically enlarge the minority sets of previous blocks. Through updating previous ensemble members using the examples in the most recent chunk, GRE can quickly adapt to new conditions, regardless the types of concept drifts. Then, a weighted voting of ensemble members is used to make predictions. The weights of past hypotheses are based on the performances on examples in the latest training data chunk. The candidate classifiers are treated as “perfect” ones and provided with the highest weight, which is especially effective for reacting to sudden drifts. Meanwhile, the cross-validation of candidate classifiers can be avoidable.

Third, we present a detailed empirical study on both the synthetic and real-world datasets. The effects of predefined chunk sizes and the number of candidate ensemble members are analyzed. Our proposed algorithm is robust against different predefined chunk sizes by periodically updating previous ensemble members. GRE using a small value of p can preserve enough knowledge of data streams to obtain a good generalization performance. Then, we compare GRE with other state-of-art methods in terms of accuracy, F-measure, G-mean, recall, and AUC. The statistical tests suggest that GRE obtains a perfect tradeoff between majority class and minority class performances. Compared with the common streaming classifiers that specialize in only one type of concept drifts, the experimental results show that GRE is able to accommodate a wide variety of drift scenarios.

In the future, we would like to extend our work to cope with the multi-class classification issue.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant nos. 11171369, 61272395, 61370171, 61300128, 61572178, 61672214).

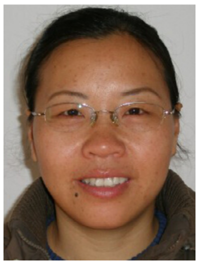
References

[1] P. Domingos, G. Hulten, Mining high-speed data streams, in: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2000, pp. 71–80.
[2] S. Muthukrishnan, Data Streams: Algorithms and Applications, Now Publishers Inc., 2005.
[3] B. Krawczyk, J. Stefanowski, M. Wozniak, Data stream classification and big data analytics, Neurocomputing 150 (PA) (2015) 238–239.
[4] T. Jo, N. Japkowicz, Class imbalances versus small disjuncts, ACM SIGKDD Explor. Newsl. 6 (1) (2004) 40–49.
[5] V. García, J. Sánchez, R. Mollineda, An empirical study of the behavior of classifiers on imbalanced and overlapped data sets, in: Proceedings of the 2007 Iberoamerican Congress on Pattern Recognition, Springer, 2007, pp. 397–406.
[6] J. Stefanowski, Dealing with data difficulty factors while learning from imbalanced data, in: Proceedings of the 2016 Challenges in Computational Statistics and Data Mining, Springer, 2016, pp. 333–363.
[7] D. Brzezinski, J. Stefanowski, Reacting to different types of concept drift: the accuracy updated ensemble algorithm, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2014) 81–94.
[8] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, Moa: massive online analysis, J. Mach. Learn. Res. 11 (2010) 1601–1604.
[9] D. Brzeziński, Mining Data Streams with Concept Drift, Poznan University of Technology, 2010 Ph.D. thesis, Master's thesis.
[10] A. Bifet, R. Gavalda, Learning from time-changing data with adaptive windowing., in: Proceedings of the 2007 SIAM International Conference on Data Mining (SDM), vol. 7, SIAM, 2007, p. 2007.
[11] I. Žliobaitė, Combining time and space similarity for small size learning under concept drift, in: Proceedings of the 2009 Foundations of Intelligent Systems, Springer, 2009, pp. 412–421.
[12] E. Cohen, M. Strauss, Maintaining time-decaying stream aggregates, in: Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ACM, 2003, pp. 223–233.
[13] E. Page, Continuous inspection schemes, Biometrika 41 (1–2) (1954) 100–115.
[14] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Proceedings of the 2004 Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence – SBIA 2004, Springer, 2004, pp. 286–295.
[15] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales-Bueno, Early drift detection method, in: Proceedings of the Fourth International Workshop on Knowledge Discovery from Data Streams, vol. 6, 2006, pp. 77–86.
[16] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2001, pp. 97–106.
[17] H. Wang, W. Fan, P.S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2003, pp. 226–235.
[18] W.N. Street, Y. Kim, A streaming ensemble algorithm (sea) for large-scale classification, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2001, pp. 377–382.
[19] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Trans. Neural Netw. 22 (10) (2011) 1517–1531.
[20] N.C. Oza, S. Russell, Experimental comparisons of online and batch versions of bagging and boosting, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2001, pp. 359–364.
[21] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[22] Y. Freund, R.E. Schapire, et al., Experiments with a new boosting algorithm, in: Proceedings of the Thirteenth International Conference on International Conference on Machine Learning (ICML), vol. 96, 1996, pp. 148–156.
[23] J.Z. Kolter, M.A. Maloof, Dynamic weighted majority: a new ensemble method for tracking concept drift, in: Proceedings of the Third IEEE International Conference on Data Mining, ICDM 2003, IEEE, 2003, pp. 123–130.
[24] J. Gao, W. Fan, J. Han, S.Y. Philip, A general framework for mining concept–drifting data streams with skewed distributions., in: Proceedings of the 2007 SIAM International Conference on Data Mining (SDM), SIAM, 2007, pp. 3–14.
[25] S. Chen, H. He, Sera: selectively recursive approach towards nonstationary imbalanced stream data mining, in: Proceedings of the 2009 International Joint Conference on Neural Networks, pp. 522–529.
[26] R.N. Lichtenwalter, N.V. Chawla, Adaptive methods for classification in arbitrarily imbalanced and drifting data streams, in: Proceedings of the 2009 Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2009, pp. 53–75.
[27] S. Chen, H. He, K. Li, S. Desai, Musera: multiple selectively recursive approach towards imbalanced stream data mining, in: Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN), IEEE, 2010, pp. 1–8.
[28] K. Wu, A. Edwards, W. Fan, J. Gao, K. Zhang, Classifying imbalanced data streams via dynamic feature group weighting with importance sampling, in: Proceedings of the 2014 SIAM International Conference on Data Mining (SDM), SIAM, 2014, pp. 722–730.
[29] Y. Wang, Y. Zhang, Y. Wang, Mining data streams with skewed distribution by static classifier ensemble, in: Opportunities and Challenges for Next-Generation Applied Intelligence, Springer, 2009, pp. 65–71.
[30] S. Chen, H. He, Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach, Evol. Syst. 2 (1) (2011) 35–50.
[31] G. Ditzler, R. Polikar, Incremental learning of concept drift from streaming imbalanced data, IEEE Trans. Knowl. Data Eng. 25 (10) (2013) 2283–2301.
[32] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, Smote: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.
[33] S. Wang, L.L. Minku, X. Yao, Resampling-based ensemble methods for online class imbalance learning, IEEE Trans. Knowl. Data Eng. 27 (5) (2015) 1356–1368.
[34] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise., in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD), vol. 96, 1996, pp. 226–231.
[35] J.L. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (9) (1975) 509–517.
[36] E. Schubert, J. Sander, M. Ester, H.P. Kriegel, X. Xu, DBSCAN revisited, revisited: why and how you should (still) use DBSCAN, ACM Trans. Database Syst. (TODS) 42 (3) (2017) 19.
[37] N. Japkowicz, S. Stephen, The class imbalance problem: a systematic study, Intell. Data Anal. 6 (5) (2002) 429–449.
[38] J.Z. Kolter, M.A. Maloof, Dynamic weighted majority: an ensemble method for drifting concepts, J. Mach. Learn. Res. 8 (2007) 2755–2790.

[39] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[40] A. Shaker, E. Hüllermeier, Recovery analysis for adaptive learning from non-stationary data streams: experimental design and case study, Neurocomputing 150 (2015) 250–264.

Siqi Ren received the B.S. degree in computer science and technology from Sichuan University in 2013. She is currently working toward the Ph.D.degree in computer science and technology at Hunan University. Her current research interests include data stream algorithm, online learning and machine learning.

Bo Liao received the Ph.D. degree in computational mathematics from the Dalian University of Technology, China, in 2004. He is currently at Hunan University as a professor. He was with the Graduate University of Chinese Academy of Sciences as a post doctorate from 2004 to 2006. His research interests include bioinformatics, data mining and machine learning.


Wen Zhu received the M.Sc. degree in computer science and technology from Hunan University, China, in 2010. She is currently Hunan University as a lecturer. Her current research interest includes bioinformatics, data mining and machine learning.

Zeng Li received the B.S. degree in computer science and technology from Sichuan University in 2013. He is currently working toward the Ph.D. degree in computer science from University of Science and Technology of China. His current research interests include data mining, data management and information security.

Wei Liu is currently working toward the Ph.D. degree in the College of Information Science and Engineering, Hunan University, Changsha, China. His research interests include machine learning, data mining and bioinformatics.

Keqin Li is a SUNY Distinguished Professor of computer science. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things and cyber-physical systems. He has published over 410 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently or has served on the editorial boards of IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, IEEE Transactions on Cloud Computing, Journal of Parallel and Distributed Computing. He is an IEEE Fellow.