Knowledge-Based Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning

Siqi Ren [a,c], Wen Zhu [b], Bo Liao [a,b,*], Zeng Li [d], Peng Wang [a], Keqin Li [a,e], Min Chen [a,f], Zejun Li [a,f]

[a] College of Information Science and Engineering, Hunan University, Changsha 410082, Hunan, China
[b] School of Mathematics and Statistics, Hainan Normal University, Haikou, China
[c] School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou 310016, Zhejiang, China
[d] School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, Anhui, China
[e] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA
[f] College of Computer and Information Science, Hunan Institute of Technology, Hengyang 421000, China

HIGHLIGHTS

- A classifier is proposed to handle concept drift and class imbalance.
- The selectively resampling method avoids drifting examples and difficult examples.
- SRE can quickly react to different kinds of concept drift.
- Costly misclassification examples and minority examples are emphasized.
- SRE is robust against chunk size.
- SRE obtains a good overall balance in accuracy, recall, G-mean, F-measure, and AUC.

ARTICLE INFO

Article history:
Received 12 May 2018
Received in revised form 9 August 2018
Accepted 20 September 2018
Available online xxxx

Keywords:
Data stream classification
Concept drift
Class imbalance
Ensemble

ABSTRACT

Although the issues of concept drift and class imbalance have been studied separately, the joint problem is underexplored even though it has received increasing attention. Concept drift is further complicated when the dataset is class imbalanced. Meanwhile, most of the existing techniques have ignored the influence of complex data distribution on learning imbalanced data streams.

To overcome these issues, we propose an ensemble-based model for learning concept drift from imbalanced data streams with complex data distribution, called selection-based resampling ensemble (SRE). SRE combines the operators of resampling and periodical update to handle the joint issue. In the chunk-based framework, a selection-based resampling mechanism, which focuses on drifting and unsafe examples, is first employed to re-balance the class distribution of the latest block. Then, previous ensemble members are periodically updated using the latest examples, where update weights are determined to emphasize costly misclassification examples and minority examples. Meanwhile, SRE can quickly react to new conditions. Empirical studies demonstrate the effectiveness of SRE in learning nonstationary imbalanced data streams.

© 2018 Published by Elsevier B.V.

1. Introduction

1.1. Motivation

In the field of traditional data mining, it is possible to store the entire data. Meanwhile, models can access each instance multiple times. Consequently, a precision model can be obtained in a batch manner. However, data items of data streams are provided sequentially and rapidly over time [1]. Learning classifiers from data streams has been widely used in the area of machine learning, data mining, and pattern recognition [2]. Only limited knowledge of data streams can be utilized at each time step and systems

* Corresponding author.
E-mail addresses: siqirenzl@163.com (S. Ren), syzhuwen@163.com (W. Zhu), dragonbw@163.com (B. Liao), lizeng@mail.ustc.edu.cn (Z. Li), wangpenglw@126.com (P. Wang), lik@newpaltz.edu (K. Li), chenmin@hnit.edu.cn (M. Chen), lzjfox@163.com (Z. Li).

https://doi.org/10.1016/j.knosys.2018.09.032
0950-7051/© 2018 Published by Elsevier B.V.

cannot retrieve previous data. Thus, data stream classifiers only obtain approximate results for testing events. Our main objective of the study is to learn a classification model from a data stream in order to make its result approximate that of the batch processing by making full use of knowledge.

Traditional data mining models aim at extracting knowledge hidden in a finite dataset whose data distribution is stationary. However, data stream models act in dynamic environments, where data items are collected over time. The target concepts of data streams change over time, which is called concept drift [3]. Examples of such applications include credit-card fraud detection and telecommunications [2]. Data stream models should be constantly adjusted or rebuilt to adapt to new conditions. The underlying changes of data distributions can be categorized as sudden, gradual, incremental, and recurrent drifts. Real-world problems are generally combinations of different types of drift. However, most of the existing solutions generally specialize in only one type of drift. The goal of the study is to design a classification model to handle different types of concept drift.

In addition to high volumes and concept drift, class imbalance is a tricky issue in the area of data stream mining. If examples come from two classes, class imbalance learning refers to unequal distribution of examples between two classes, where the class having most of the examples is majority class and the other class is minority class [4,5]. Traditional classifiers tend to be biased toward majority examples and perform poorly on minority examples [6]. A large number of researches in class imbalance problems have been done. These techniques can be categorized into three groups: data level approaches [7,8], algorithm level approaches [9], and cost-sensitive approaches [10]. In [7], the labels of new data are determined by computing their distances from the nearest generalized instance. The selection of the most suitable generalized exemplars is optimized by evolutionary algorithms. García et al. analyzed how different resampling techniques affect the learning procedure [8]. Algorithm level approaches are defined as internal approaches, since their performance always depends on the problems and the classifiers [9]. Cost-sensitive approaches handle the class imbalance issue by assigning higher misclassification costs to minority samples. BEE-Miner generates neighbor solutions and evaluates the quality of the solutions [10]. The class imbalance classification issue occurs in many real-world applications. Based on the synthetic minority over-sampling technique (SMOTE) and the bagging ensemble learning algorithm with differentiated sampling rates (DSR), Sun et al. addressed a decision tree ensemble model for imbalanced enterprise credit evaluation [11]. The multi-class classification models have been used to predict the listing status of companies [12,13]. In addition to the imbalance ratio (IR), the degradation of model performance is related to data difficulty factors, including small disjuncts, outliers, and class overlapping [14,15]. However, most researchers have ignored these factors when learning imbalanced data streams. The minority examples can be categorized as safe, borderline, outliers, and rare examples. The last three types of examples are known as unsafe examples. They are difficult to learn and should be given more focus. The motivation of this study is to learn imbalanced data streams with unsafe examples.

1.2. Contribution

Although the two problems, namely learning concept drift and learning from imbalanced data, have been studied separately, not much work discusses the issues when both class imbalance and concept drift exist. Applications in various domains such as climate monitoring, spam filtering, and anomaly detection are affected by both class imbalance and concept drift [16]. In this paper, we propose a novel data stream classifier, called selection-based resampling ensemble (SRE), whose goal is to learn nonstationary imbalanced data streams. The main contributions are described as follows.

1. SRE oversamples the current minority set using past minority samples. Data difficulty factors and concept drift impose difficulties on the selection of an appropriate subset of past minority instances. SRE avoids absorbing drifting data by measuring the similarity between each of previous minority examples and the current minority set. Then, selection weights are assigned to past minority samples according to data difficulty factors.
2. SRE can address concept drift by dynamically assigning weights to past ensemble members according to their performance on the latest events. The latest component, which is also known as the candidate component, is derived from the most recent data chunk. The candidate component is treated as the best-performing classifier and should be provided the highest weight. This weighting setting is especially important for the adaptation of sudden drift. Meanwhile, cross-validation of the candidate classifier can be avoided to cater to high-speed data streams.
3. Most of the existing methods specialize in only one type of concept drift. Through updating previous hypotheses using latest examples, SRE can handle different types of concept drift, including sudden, gradual, incremental, and recurrent concept drifts. The update weights of examples, which describe the probabilities of being selected to update a specific past classifier, are evaluated according to classifier outputs and sample labels. The misclassification costs denote uneven identification importance between classes. Costly misclassified examples and minority examples are emphasized in the periodical update procedure.
4. SRE, which integrates operators of chunk-based and online ensembles, is a hybrid ensemble [17,18]. On the one hand, SRE maintains the mechanism of online updating of ensemble members known from online ensembles, which improves the models' reactions to different kinds of concept drift. Thus, previous classifiers are periodically updated using the current knowledge. The performance of SRE is robust against chunk size. On the other hand, data chunks are equally divided beforehand. Ensemble members are built over consecutive data blocks. Then, a weighted result of all components is used to make decision. Thus, the knowledge of data streams is made full use of to approximate results of batch processing.

1.3. Paper organization

The rest of this paper is organized as follows. In Section 2, we review the related researches to learn concept drift from imbalanced data. The differences between SRE and other existing algorithms are provided. Section 3 provides a detailed introduction of the proposed algorithm. The time complexity of SRE is discussed in Section 4. Experimental analyses and results are presented in Section 5. Section 6 concludes the paper and suggests a direction for the future research.

2. Existing research

Learning from imbalanced data streams is required to resolve the combined issue of concept drift and class imbalance. The decision boundaries of standard classifiers tend to be biased toward the majority class, and thus minority events are easy to be misclassified. The class imbalance issue has been extensively studied in stationary scenarios. However, learning evolving concepts from imbalanced data has been mostly underexplored.

Please cite this article in press as: S. Ren, et al., Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.09.032.

Uncorrelated bagging (UB) [19] is one of the earliest techniques to learn nonstationary imbalanced data streams. The class distribution of the latest chunk is first re-balanced by all the preserved minority examples. Then, a component classifier is trained over a randomly selected majority sample subset and all minority samples. However, it is assumed that minority data among consecutive blocks are stationary. In selectively recursive approach (SERA) [20], the Mahalanobis distance is used as the similarity measurement to select past minority samples that are most similar to those in the candidate block. Thus, SERA avoids propagating drifting data into the latest chunk. The oversampling mechanism of multiple selectively recursive approach (MuSeRA) [21] is the same as that of SERA. In contrast to SERA, MuSeRA maintains all hypotheses built over consecutive blocks to make predictions. Ditzler and Polikar proposed two batch-based ensembles called learn++ for concept drift with smote (Learn++.CDS) [22] and learn++ for nonstationary imbalanced environments (Learn++.NIE) [22]. They are derived from Learn++.NSE [23] to tackle imbalanced data streams with concept drift. Learn++.CDC integrates the mechanisms of Learn++.NSE and SMOTE [24], where Learn++.NSE overcomes concept drift through a dynamic weighting strategy and SMOTE enriches the feature space of minority data. Learn++.NIE modifies the penalty constraint to place more importance on misclassified minority instances. Meanwhile, bagging-based subensembles are used to create subsets with relatively balanced class distribution without generating novel data or resampling minority data. The modification of heuristic updatable weighted random subspaces (HUWRS) [25] was addressed to handle nonstationary data streams which exhibit class imbalance [26]. In heuristic updatable weighted random subspaces with instance propagation (HUWRS.IP), an instance propagation method is used to resample minority examples. The probabilities of instances to be selected are based on their similarities to the current concept. In dynamic feature group weighting framework with importance sampling (DFGW-IS) [27], concept drift can be tackled by weighting each ensemble member based on discriminative power and stable level. Meanwhile, the training set of each component classifier is extracted from a specific feature group. Wang et al. addressed two online ensembles called oversampling-based online bagging (OOB) and undersampling-based online bagging (UOB) [28]. UOB and OOB can cope with imbalanced data streams which exhibit changing imbalance status through resampling and time-decayed metrics. Then, WEOB1 and WEOB2 [28], which maintain both OOB and UOB with the adaptive weight adjustment, were proposed to obtain a tradeoff between predictive accuracy and robustness.

In general, the ensemble-based classifiers to handle the joint problem of concept drift and class imbalance can be categorized into two groups: chunk-based ensembles [19–22,26,27] and online ensembles [28]. In the chunk-based framework, data arrive over time in batches of examples and models process a batch of examples at each time step. However, online ensembles process streaming data one by one. Most of the chunk-based ensembles, such as UB [19], SERA [20], MuSeRA [21], HUWRS.IP [26], and DFGW-IS [27], resample minority examples from ones in the past chunks to obtain a relatively balanced candidate block. The main differences among them are the techniques to extract minority examples of previous blocks and ideas to build the ensemble model.

UB [19] blindly absorbs all the previous minority examples to promote class balance, which easily brings drifting data. All the ensemble members are built over the latest block. SERA [20], HUWRS.IP [26], and DFGW-IS [27] use similar ideas to UB [19] of building an ensemble of weighted classifiers. However, they leverage "smarter" oversampling techniques to avoid absorbing drifting data into the latest block. MuSeRA [21] uses the similar strategy to select an appropriate subset of past minority examples. However, MuSeRA [21] can maintain all the knowledge of data streams by preserving all the ensemble members built over consecutive blocks. The issue of concept drift can be passively overcome by continuously adjusting the weights of component classifiers according to the evolving concepts.

However, all of the above algorithms have ignored the data difficulty factors on the classification task. On the one hand, SRE can avoid propagating drifting data into the latest block by evaluating the similarities between past preserved minority examples and the current minority class set. On the other hand, the probability of a past minority example to be selected depends on data difficulty factors. Thus, SRE can select an appropriate subset from preserved minority examples. Through absorbing the selected minority examples into the latest chunk, the amplified block is generated. Then, the candidate component is built over the amplified block. Similar to MuSeRA, SRE can leverage past knowledge of data streams by preserving ensemble members built over past blocks.

The performance of chunk-based ensembles generally relies on the predefined size of data chunks. On the one hand, chunk size can determine the number of training data if chunk-based ensembles build all ensemble members over the latest block. On the other hand, using fixed-size blocks is easily to make the models catch in the stability–plasticity dilemma if chunk-based ensembles build one hypothesis over a data chunk [23]. In SRE, previous ensemble members are periodically updated using the latest observations, where the update weights of samples are based on their categories and classifier outputs over previous components. This way, the previous chunks are constantly amplified. Thus, the performance of SRE does not rely on chunk size (discussed in more detail in Section 5.4). Most of the existing techniques for handling imbalanced data streams specialize in only one type of concept drift. SRE can timely react to multiple kinds of concept drift by the periodical update mechanism. This is based on the assumption that the latest instances can well represent the current and near-future concepts.

3. The selection-based resampling ensemble (SRE) algorithm

Motivated by the joint issue of concept drift and class imbalance, we propose a novel streaming classifier called SRE. In this paper, we focus on the binary classification issue. A set of learning examples containing pairs $\{\mathbf{x}, y\}$, where $\mathbf{x}$ is a vector of attribute values and $y \in \{+1, -1\}$ is a class label, arrives continuously in the form of a data stream. Samples with class label $y = -1$ are much more than samples with class label $y = +1$ [29]. The notations used in this paper and their descriptions are summarized in Table 1.

Fig. 1 provides the system level framework of the SRE algorithm. The operators of SRE involve four phases. In the first phase, a data stream is divided into fixed-size blocks. To limit the memory usage, SRE only preserves the minority examples in the latest w blocks. Then, a selectively resampling technique is used to select an appropriate subset of the past preserved minority samples to rebalance the current class distribution and a candidate component classifier is built over the amplified data chunk. h_t is the component classifier built over the tth amplified block. In the second phase, the weights of component classifiers are evaluated. In order to limit the time and memory usage, a performance-based pruning technique is applied to the ensemble when the ensemble size exceeds a predefined threshold. In the third phase, previous ensemble members are periodically updated using the latest instances, and thus the ensemble can react to different kinds of concept drift. The latest instances are weighted to emphasize costly misclassification examples and minority examples. In the fourth phase, the predictions of component classifiers are aggregated using a weighted voting rule to make decision. $h_t(x'_j)$ is the predictive label of a testing instance x'_j at timestamp t. The complete method is presented in Algorithm 1. Lines 1–5 and 6–14 comprise the first and second phases, respectively. Finally, lines 15–18 and 19–21 complete the third and fourth phases, respectively. The main components of SRE are described in Sections 3.1–3.3.

Please cite this article in press as: S. Ren, et al., Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.09.032.

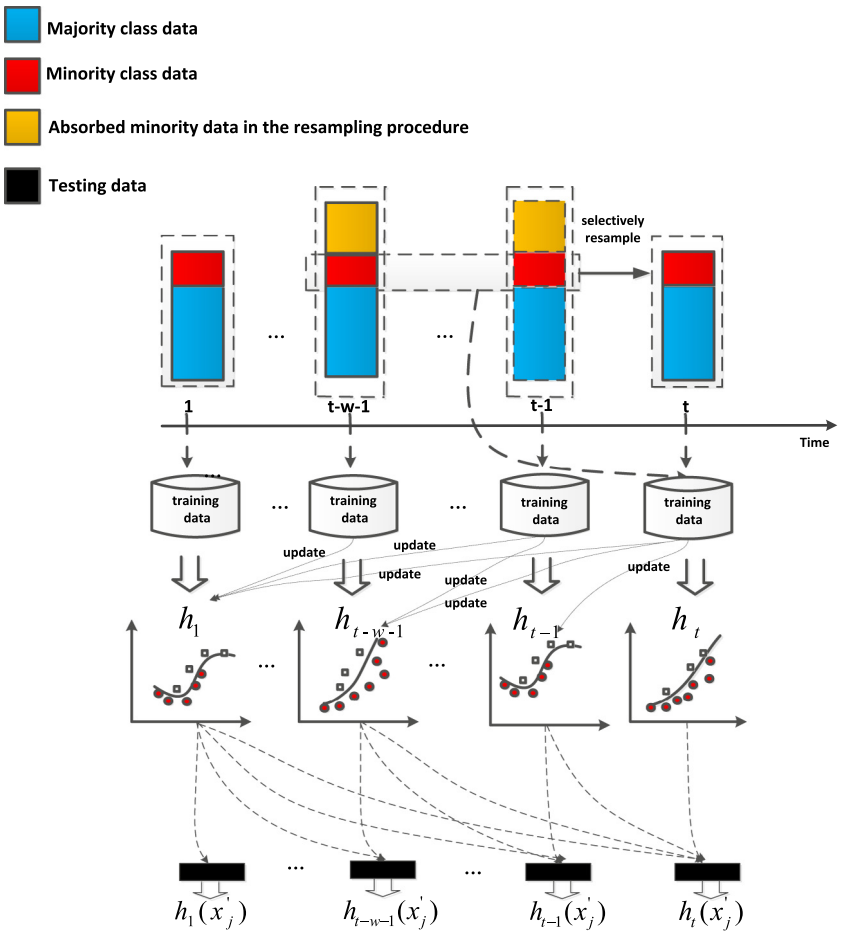

Fig. 1. The learning flow of the SRE algorithm.

Table 1
Notations.

Notation	Description	Notation	Description
t	Current time stamp	c	Ensemble size
d	Chunk size	B_t	Current data block
S_t and T_t	Current training and testing sets, respectively	a and b	Numbers of instances in S_t and T_t, respectively
P_t and N_t	Minority and majority sets in S_t, respectively	x_j and y_j	Instance in S_t and its label
x_j' and y_j'	Instance in T_t and its label	$\hat{y}_j'$	Label of x_j' predicted by the ensemble
f	Post-balance ratio	IR_t	Imbalance ratio of S_t
h_t and w_{h_t}	Candidate classifier and its weight	h_n and w_{h_n}	nth component classifier and its weight
ε and $\|\varepsilon\|$	Ensemble model and the number of members in ε	$w_{t,n}(x_j)$	Update weight of x_j
$w_t^c(x_j)$	Cost weight of x_j	$w_{t,n}^{IR}(x_j)$	Category weight of x_j
M and $\|M\|$	Set containing minority examples of previous w blocks and its cardinality	k'	Number of examples with different categories among an instance's k-nearest neighbors
m_i	Minority example in M	$\|P_t\|$ and $\|N_t\|$	Numbers of instances in P_t and N_t, respectively
M_0	Noise dataset in M	M_1 and $\|M_1\|$	Filtered minority set in M and its cardinality
P_t'	Minority set in current amplified training block	M_2	Selected minority set in M for oversampling S_t
Cl_i	Nearest minority class cluster of m_i	Cl_i'	Nearest majority class cluster of m_i
d_i	Distance of m_i from Cl_i	q_i	Order of m_i according to d_i
d_i'	Distance of m_i from Cl_i'	q_i'	Order of m_i according to $w(m_i)$
Bo_t	Borderline sample set	m_i^b	Borderline example nearest to m_i
$w(m_i)$	Selection weight of m_i	$Bf(m_i)$	Borderline factor of m_i
$Df(m_i)$	Disjunct factor of m_i	$Of(m_i)$	Deviation factor of m_i
$dist(\cdot,\cdot)$	Distance of two data points	den_i	Density of cluster Cl_i
$h_n(x_j)$	Label of x_j predicted by h_n	$\beta(j)$	Cost adjustment function
C_j	Misclassification cost of x_j	$P(y_j)$	y_j's class distribution
$IR_{t,n}$	Imbalance ratio of the union set of training instances of the tth and nth ensemble members	Z_n and Z_w	Normalization factors of the cost weight and the category weight, respectively
MSE_r	Mean square error of a random prediction	MSE_t	Mean square error of h_t on examples in S_t
$h_n(x_j')$ and $h_t(x_j')$	Labels of x_j' predicted by h_n and h_t, respectively	$f_{y_j}^n(x_j)$	Probability of x_j being classified as y_j by h_n
N	Number of blocks in a data stream	$O(\cdot)$	Computational complexity

Please cite this article in press as: S. Ren, et al., Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.09.032.

Algorithm 1 Selection-based Resampling Ensemble Algorithm

Input: t: Timestamp of the current block; c: Ensemble size; d: chunk size; B_t: Current data block; S_t: Current training set containing a instances $(x_1, y_1), \cdots, (x_a, y_a)$; P_t: Set of minority samples in the current training block; N_t: Set of majority samples in the current training block; T_t: Current testing set containing b instances $\{x_1^{'}, x_2^{'} \cdots, x_b^{'}\}$; f: Post-balance ratio specifying the imbalance ratio after resampling the training block S_t.

Output: $\hat{y_j^{'}}$: Predictive label of a testing event $x_j^{'}$

1: $IR_t \leftarrow \frac{|P_t|}{|N_t|}$ //IR_t: Imbalance ratio specifying the proportion between the minority samples and majority samples of S_t
2: **if** $IR_t < f$ **then**
3: Call the Algorithm 2 and obtain $P_t^{'}$
4: **end if**
5: Build a candidate classifier h_t over $N_t \cup P_t^{'}$
6: Compute w_{h_c} using Eq. (16)
7: **for** each previous ensemble member h_n $(n = 1, 2, \cdots, t-1)$ **do**
8: Compute w_{h_n} using Eq. (13)
9: **end for**
10: **if** $|\varepsilon| < c$ **then**
11: $\varepsilon \leftarrow \varepsilon \cup h_t$
12: **else**
13: Substitute the worst-performing classifier in ε with h_t
14: **end if**
15: **for** all previous classifiers $h_n \in \varepsilon$ $(n = 1, 2, \cdots, t-1)$ **do**
16: Compute the update weight $w_{t,n}(x_j)$ for each sample $x_j \in S_t$ according to Eq. (7)
17: Update h_n using the sample $x_j \in S_t$
18: **end for**
19: **for** $j = 1, 2, \cdots, b$ **do**
20: Assign $\hat{y_j^{'}}$ using Eq. (17)
21: **end for**

3.1. Selectively resampling mechanism

In the chunk-based framework, many existing solutions oversample the current minority set using the minority instances of past blocks. The random oversampling method simply duplicates the original minority data and cannot absorb novel information into the candidate block. The random undersampling method [30] re-balances the current class distribution by removing majority samples; but the important events are easy to be removed. The synthetic oversampling technique [24] enlarges the minority class region by strategically placing synthetic examples on the line segment connecting two minority events. In reality, the past minority samples are more suitable for adjusting the bias of the candidate classifier compared to synthetic data generated by synthetic oversampling techniques [20].

Algorithm 2 presents the pseudo-code of the selectively resampling procedure. SRE preserves only the minority examples of previous w blocks in M to limit the memory usage. This is based on the assumption that the latest examples are the best representatives of the current and near-further data distributions. If the number of instances in M cannot re-balance the current skewed distribution, then all the examples in M are propagated into the current block (lines 1 and 2 of [Algorithm 2]). Otherwise, a sample set is selected from M to resample the latest minority set P_t. Not all preserved minority examples $m_i \in M$ can improve the training procedure of the candidate component classifier because some of these events may bring concept drift or locate in the wrong region.

The major contributions of the selectively resampling mechanism in SRE are twofold: to avoid drifting data (lines 9–20 of [Algorithm 2]) and to avoid unsafe data (lines 4–8 and 21–37 of [Algorithm 2]). On the one hand, the similarity between each of past preserved minority samples and the current minority set is evaluated. Past preserved minority samples, which are similar to the current minority set, have the priority to be selected to

Algorithm 2 Selectively Resampling Procedure

Input: t: Timestamp of the current block; B_t: Current block; S_t and a: Current training block and the number of instances in S_t; IR_t: Imbalance ratio specifying the proportion between the minority samples and majority samples of the current training block S_t; f: Post-balance ratio specifying the imbalance ratio after resampling the training block S_t; $f^{'}$: Imbalance ratio after resampling P_t by considering concept drift and outliers; M and $|M|$: Set of samples in previous w blocks and its cardinality, obviously $M = \emptyset$ when $t = 1$; P_t: Set of minority samples in S_t; N_t: Set of majority samples in S_t.

Output: $P_t^{'}$: Set of minority samples in the current amplified training block

1: **if** $|M| < (f - IR_t) \times |N_t|$ **then**
2: $M_2 \leftarrow M$
3: **else**
4: **for** each minority example $m_i \in M$ **do**
5: Compute the k-nearest neighbor set of m_i,
6: Find the minority set M_0 from M in which instances have no minority examples in their neighbors
7: **end for**
8: $M_1 \leftarrow M - M_0$
9: **if** $|M_1| < (f - IR_t) \times |N_t|$ **then**
10: $M_2 \leftarrow M_1$
11: **else**
12: **if** $|M_1| < (f^{'} - IR_t) \times |N_t|$ **then**
13: $M^{'} \leftarrow M_1$
14: **else**
15: Cluster P_t into several clusters, which are $Cl_1, Cl_2, \cdots$
16: Compute the distance d_i of $m_i \in M_1$ from its nearest cluster Cl_i of P_t
17: Sort d_i in ascending order and obtain the order q_i for m_i
18: Select a sample set $M^{'}$, containing the first $(f^{'} - IR_t) \times |N_t|$ observations, from M_1 according to q_i
19: **end if**
20: **end if**
21: **if** $|M^{'}| < (f - IR_t) \times |N_t|$ **then**
22: $M_2 \leftarrow M^{'}$
23: **else**
24: Find the borderline sample set Bo_t from the set $M^{'} \cup P_t \cup N_t$
25: Obtain the density of each cluster Cl_i
26: Cluster N_t into several clusters, which are $Cl_1^{'}, Cl_2^{'}, \cdots$
27: **for** each minority example $m_i \in M^{'}$ **do**
28: Compute the distance of m_i from its nearest sample $m_i^b \in Bo_t$
29: Compute the borderline factor of m_i according to Eq. (2)
30: Compute the disjunct factor of m_i according Eq. (4)
31: Compute the distance $d_i^{'}$ of m_i from its nearest cluster $Cl_i^{'}$ of N_t
32: Compute the deviation factor of m_i according to Eq. (5)
33: Obtain the selection weight $w(m_i)$ of m_i according to Eq. (1)
34: **end for**
35: Sort $w(m_i)$ in descending order and obtain the order $q_i^{'}$ for m_i
36: Select a sample set M_2, containing the first $(f - IR_t) \times |N_t|$ observations, from $M^{'}$ according to $q_i^{'}$
37: **end if**
38: **end if**
39: $P_t^{'} \leftarrow P_t \cup M_2$

enlarge the training set of the candidate classifier. The Mahalanobis distance [31] has the priority to measure such similarity. Compared to Euclidean distance, Mahalanobis distance takes the correlations of data into account. In SRE, Mahalanobis distance has the priority to be used to measure the similarity between two data points. However, Mahalanobis distance has its limitations and cannot be computed in some situations. For example, Mahalanobis distance cannot be used when the dimension of samples is larger than the number of samples. In these situations, we use Euclidean distance instead of Mahalanobis distance.

The data difficulty factors, on the other hand, should be considered to improve the instance selection scheme. The selection weights of past minority examples, which describe their probabilities of being selected to resample the current minority set based on the data complexity, should be assigned according to the following factors (A larger weight implies that the sample has a higher probability of being selected).

- **Noise factor:** Noise data are likely to pose difficulties in re-balancing imbalanced data streams. In SRE, the k-nearest

Please cite this article in press as: S. Ren, et al., Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.09.032.

neighbors framework is used to recognize noisy samples. For each considered instance, k' denotes the number of examples with different categories among its k-nearest neighbors (assuming $k = 5$). A preserved past minority sample is regarded as noise and should be avoided, if the number of majority examples among its k-nearest neighbors satisfies $k' = 5$ [32].

- **Borderline factor:** The minority samples, which are close to the decision boundary, generally contribute more to the classification task and should be adaptively assigned high selection weights. If the number of examples with different categories among an example satisfies $\frac{k}{2} \leq k' < k$, then this example is recognized as a borderline sample (assuming $k = 5$) [32].
- **Disjunct factor:** In addition to an unequal number of samples for different classes, the within-class imbalance issue, which results from small disjuncts, induces the difficulty in learning imbalanced data streams. As dense clusters contain more information than sparse clusters, the resampling task is especially crucial to sparse clusters of minority class to reduce within-class imbalance. Thus, the past minority examples located near sparse clusters of minority class should have priority to be selected.
- **Deviation factor:** The previous minority samples, which erroneously fall in the majority class region, degrade the model performance. The similarities between past minority samples and the current majority set should be calculated. If a preserved past minority sample is far away from the current majority class region, then it has the priority to be selected to enlarge the current minority set.

The whole process of the selectively resampling mechanism in SRE can be described as follows. First, the noise dataset M_0 should be recognized and removed from the past preserved minority set M, thus obtaining the filtered minority set M_1 (lines 4–8 of [Algorithm 2]). Fig. 2 presents the data difficulty factors of the selectively resampling procedure in SRE. Squares represent the minority examples in the latest training chunk S_t; circles denote the preserved minority examples in previous w blocks; stars are majority examples in the latest training block S_t. Noise data in M complicate the learning process of streaming classifiers and should be ignored in the following operation. It can be observed from Fig. 2(a) that the sample A has no minority events in its k-nearest neighbors (assuming $k = 5$). Thus, it is not suitable for re-balancing the current skewed distribution.

Second, all the samples in M_1 are selected to resample P_t if the number of instances in M_1 cannot re-balance the current distribution (lines 9 and 10 of [Algorithm 2]). f' is the imbalance ratio of the current block after the resampling procedure by considering concept drift and outliers. If all the instances in M_1 cannot obtain an amplified candidate block with the imbalance ratio f', then all instances in M_1 are selected (lines 12 and 13 of [Algorithm 2]). Otherwise, we select the minority set M' by measuring the similarity between each sample $m_i \in M_1$ and the current minority set P_t (lines 15–18 of [Algorithm 2]) to make the imbalance ratio of the current training block equal to f'. Consequently, the drifting data in M_1 are not absorbed into the current block B_t. Existing solutions for evaluating the similarities of samples in the field of imbalanced data stream classification have ignored the influence arising from disjuncts. To overcome this issue, SRE first discovers disjuncts of P_t. The density-based spatial clustering of applications with noise (DBSCAN) [33] is applied to divide samples of P_t into several clusters, where each cluster is regarded as a disjunct (line 15 of [Algorithm 2]). DBSCAN can discover clusters of arbitrary shape, which is suitable for streaming data. Then, the similarity between a past minority example $m_i \in M_1$ and P_t is evaluated according to the distances of m_i from clusters of P_t. d_i, which is the distance of m_i from its nearest cluster center of the minority set, is used to judge the rank of m_i to be selected to oversample P_t (line 16 of [Algorithm 2]). The orders of d_i are ranked in an ascending manner, then q_i, which is the order of d_i, is obtained (line 17 of [Algorithm 2]). To avoid the drifting data, only first $(f' - IR_t) \times |N_t|$ observations in M_1 are propagated into M' according to q_i (line 18 of [Algorithm 2]). $|N_t|$ is the number of majority instances in S_t and $IR_t = |P_t|/|N_t|$ is the imbalance ratio specifying the proportion between the minority samples and majority samples in S_t, where $|P_t|$ is the number of minority samples in S_t.

Third, all the samples in M' are propagated into P_t if the number of instances in M' cannot re-balance the current skewed distribution (lines 21 and 22 of [Algorithm 2]). Otherwise, according to data difficulty factors, the selection weight of an instance $m_i \in M'$, denoted as $w(m_i)$, is computed as

$$w(m_i) = Bf(m_i) \times Df(m_i) \times Of(m_i) \tag{1}$$

where $Bf(m_i)$ denotes the borderline factor, $Df(m_i)$ is the disjunct factor, and $Of(m_i)$ represents the deviation factor (line 33 of [Algorithm 2]). The selection weights are ranked in a descending manner, then the order of $w(m_i)$, denoted by q'_i, is obtained (line 35 of [Algorithm 2]). The selectively resampling mechanism of SRE selects only first $(f - IR_t) \times |N_t|$ observations from M' according to q'_i to avoid propagating unsafe data into P_t, where f is the post-balance ratio specifying the imbalance ratio after resampling the training block S_t (line 36 of [Algorithm 2]).

A borderline sample set Bo_t can be recognized from the set $M' \cup P_t \cup N_t$ according to [32] (line 24 of [Algorithm 2]). A past minority objective $m_i \in M'$ located near the decision boundary or belonging to Bo_t makes more contributions to the classification task and should have the priority to be selected to resample the current minority class set. $m_i^b \in Bo_t$ is the borderline instance nearest to m_i. The borderline factor of m_i is based on the distance of m_i from m_i^b as

$$Bf(m_i) = \begin{cases} \frac{1}{\sum_{m_i \in M',\ m_i^b \in Bo_t} g(dist(m_i, m_i^b))}, & \text{if } m_i \in Bo_t \\ \frac{g(dist(m_i, m_i^b))}{\sum_{m_i \in M',\ m_i^b \in Bo_t} g(dist(m_i, m_i^b))}, & \text{otherwise} \end{cases} \tag{2}$$

The distance of m_i from m_i^b, denoted by $dist(m_i, m_i^b)$, is applied to the function g. We define g as

$$g(x) = e^{-x} \tag{3}$$

Thus, $g(dist(m_i, m_i^b)) = e^{-dist(m_i, m_i^b)}$ is obtained. The selection weight of a borderline sample is higher than that of other samples according to the borderline factor. It can be seen from Fig. 2(b) that borderline data, including majority and minority samples, have been marked. Here, the noise sample A has been removed in Fig. 2(b)–(d). The sample E_b is the nearest borderline objective of E, and thus the probability of E to be selected based on the borderline factor relies on the value of $dist(E, E_b)$.

It is assumed that Cl_i is the nearest minority class cluster of the sample $m_i \in M'$. The disjunct factor of m_i, denoted by $Df(m_i)$, can be calculated as

$$Df(m_i) = \begin{cases} \frac{1}{\sum_i g(d_i \times den_i)}, & \text{if } d_i = 0 \\ \frac{g(d_i \times den_i)}{\sum_i g(d_i \times den_i)}, & \text{otherwise} \end{cases} \tag{4}$$

where d_i denotes the distance of m_i from the cluster center of Cl_i and den_i is the density of Cl_i (line 30 of [Algorithm 2]). A sample m_i located in the cluster center of Cl_i, that is $d_i = 0$, obtains a higher selection weight based on the disjunct factor than other samples. The density of a specific cluster can be transformed into the inverse value of the sum of distance values between any of two data points in this cluster (line 25 of [Algorithm 2]). As a sparse cluster does

Please cite this article in press as: S. Ren, et al., Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.09.032.

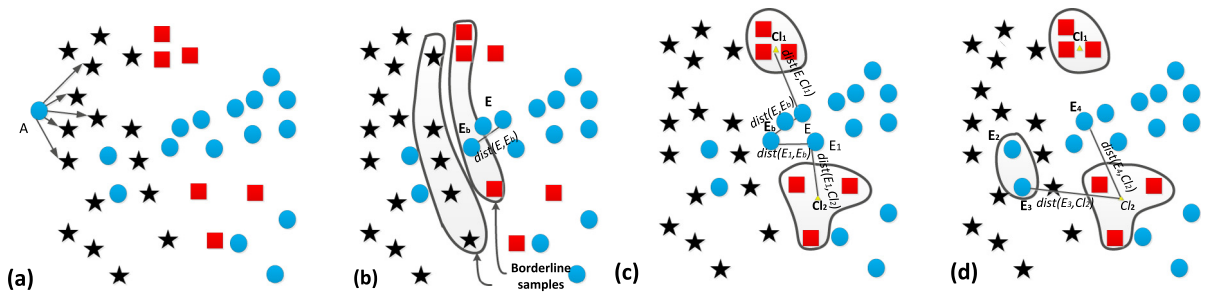

Fig. 2. The data difficulty factors of the selectively resampling procedure in SRE. (a) noise factor, (b) borderline factor, (c) disjunct factor, and (d) deviation factor.

not have enough observations to represent its concept, the within-class imbalance issue makes it difficult to correctly learn a sparse cluster. d_i and den_i are applied to g, and thus $g(d_i \times den_i) = e^{-d_i \times den_i}$. To understand the disjunct factor clearly, consider Fig. 2(c) and assume $dist(E, E_b) = dist(E_1, E_b)$. If E_b is the nearest borderline example of E and E_1, then E and E_1 have the same probability to be selected according to the borderline factor. However, the density of cluster Cl_1 is higher than that of cluster Cl_2, although Cl_1 and Cl_2 contain an equal number of data points. Therefore, E_1 is assigned a higher selection weight than E to overcome the within-class imbalance and small disjuncts problems (assuming $dist(E, Cl_1) = dist(E_1, Cl_2)$).

In addition to the borderline factor and disjunct factor, the deviation factor should be considered to refrain the class boundaries of minority events from spreading further into the majority class area. Thus, the dissimilarity of a past preserved minority example $m_i \in M'$ and the current majority set N_t should be evaluated. The clustering procedure based on DBSCAN [33] is first used to discover disjuncts of N_t (line 26 of [Algorithm 2]). Then, the dissimilarity is transformed into the distance from the nearest cluster center of the current majority set. $Of(m_i)$, denoted by the deviation factor of m_i, can be calculated as

$$Of(m_i) = \frac{g'(d'_i)}{\sum_i g'(d'_i)} \tag{5}$$

where d'_i is the distance of m_i from its nearest majority class cluster Cl'_i (lines 31 and 32 of [Algorithm 2]). d'_i is applied to the function $g'(x)$. $g'(x)$ is defined as

$$g'(x) = 1 - e^{-x} \tag{6}$$

The deviation factor assigns high weights to the members which are relatively far away from the majority class region to reduce the risk of overlapping between different classes. It can be observed from Fig. 2(d) that E_2 and E_3 deeply locate in the majority class region. If E_3 and E_4 are at the same distance from the decision boundary, then E_4 has a higher probability of being selected than E_3 according to the deviation factor because the data point E_3 falls into the dangerous region (assuming $dist(E_3, Cl_2) = dist(E_4, Cl_2)$).

3.2. Periodical update mechanism

In the chunk-based ensemble framework, most of the existing techniques to classify imbalanced data streams re-balance the current class distribution by reusing minority examples in the past blocks. However, the selected past minority samples of these blind resampling methods may be inconsistent with the current concept or locate in the dangerous region, which absorbs wrong samples into the candidate block and makes the learning task complicated.

To overcome this issue, the resampling technique in SRE, which simultaneously considers concept drift and data difficulty factors, selects past minority examples to broaden the current minority class region. This way, SRE can improve the performance of the candidate component classifier. However, the number of minority samples in the amplified candidate block may not make the current class distribution reach a desired level of balance when datasets are strongly imbalanced. To obtain balanced data chunks, resampling training minority sets of past ensemble members using the latest training minority events is an effective strategy. The objective of the periodical update mechanism is threefold: to re-balance the class distribution of past blocks, to emphasize the costly misclassification instances, and to make the ensemble quickly react to different types of concept drift.

In SRE, the minority set of the current block is amplified by selectively extracting previous minority samples (lines 1–4 of [Algorithm 1]). Then, a candidate classifier h_t is built over the amplified block (line 5 of [Algorithm 1]). To limit the time and memory usage, a performance-based pruning technique is applied to the ensemble ε. If ε contains less than c components, the newly built classifier h_t is added to ε (lines 10 and 11 of [Algorithm 1]). Otherwise, the worst-performing classifier in ε is replaced with h_t (lines 12 and 13 of [Algorithm 1]). The previous hypotheses should be periodically updated using data characterized by the current concept (lines 15–18 of [Algorithm 1]).

The update weight $w_{t,n}(x_j)$ $(1 \le n \le t-1)$, which describes the probability of instance $x_j \in S_t$ to be selected to update the nth component classifier in the ensemble, is expressed as

$$w_{t,n}(x_j) = w_t^c(x_j) \times w_{t,n}^{IR}(x_j) \tag{7}$$

where $w_t^c(x_j)$ and $w_{t,n}^{IR}(x_j)$ are the cost weight and category weight of sample x_j, respectively (line 16 of [Algorithm 1]). The update weight of samples x_j is the product of sample labels, classifier outputs over successive previous components, and the imbalance ratio of the current condition. The computations of $w_t^c(x_j)$ and $w_{t,n}^{IR}(x_j)$ are described as follows.

The cost weight $w_t^c(x_j)$ of instance x_j is based on its misclassification cost and classifier outputs over successive previous ensemble members. SRE emphasizes the costly misclassification instances when updating previous components. $w_0^c(x_j) = IR_t/Z_0$ denotes the initialization cost weight of x_j if x_j is a majority sample, then $w_0^c(x_j) = 1/Z_0$ otherwise, in which Z_0 is the initial normalization factor. IR_t is the imbalance ratio of the training set of the candidate component classifier. Then, the cost weight of x_j should be updated by the predictive results of previous components in the ensemble group. The update rule of the cost weight is

$$w_n^c(x_j) = \frac{w_{n-1}^c(x_j)\exp(-w_{h_n} \times y_j \times h_n(x_j) \times \beta(j))}{Z_n} \tag{8}$$

where $w_n^c(x_j)$ and $w_{n-1}^c(x_j)$ are cost weights of x_j at the nth and $(n-1)$th rounds, respectively. w_{h_n}, described in Section 3.3, is the weight of the nth ensemble member, y_j is the true label of x_j, $h_n(x_j)$ is the label of x_j predicted by the nth hypothesis, and $Z_n = \sum_{j=1}^{a} w_{n-1}^c(x_j)\exp(-w_{h_n} \times y_j \times h_n(x_j) \times \beta(j))$ is the normalization factor at the nth round. $\beta(j)$ presented in [34] is the cost adjustment function. It is defined as

$$\beta(j) = \begin{cases} -0.5C_j + 0.5, & \text{sign}(y_j h_n(x_j)) = +1 \\ 0.5C_j + 0.5, & \text{sign}(y_j h_n(x_j)) = -1 \end{cases} \tag{9}$$

Please cite this article in press as: S. Ren, et al., Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.09.032.

where C_j is the misclassification cost of x_j and can be calculated as

$$C_j = \begin{cases} IR_t, & y_j = -1 \\ 1, & y_j = +1 \end{cases} \tag{10}$$

where IR_t is the imbalance ratio of S_t. The misclassification cost of a majority sample is based on the imbalanced ratio of the latest training block. Meanwhile, minority samples are assigned high misclassification costs compared to majority samples. The cost weight of a sample x_j is continuously updated based on its outputs of previous component classifiers at successive rounds. For a sample with a high misclassification cost, its cost weight increases more when its label is wrongly identified by a component, but decreases less otherwise. On the contrary, the cost weight of an instance with a low misclassification cost increases conversely when it is misclassified, but decreases aggressively otherwise. In this way, the costly misclassification instances in the latest block outweigh others in terms of the cost weight.

The selectively resampling mechanism, where drifting data and unsafe examples are avoided, may not achieve adequate minority objectives to re-balance the current class distribution. Therefore, minority samples in the candidate block generally have relatively high probabilities to be selected to update the training sets of previous ensemble members compared with those majority samples with same cost weights. In addition to the cost weight, the categories of samples in the latest chunk should be considered when updating previous hypotheses. The category weight of x_j, denoted by $w^{IR}_{t,n}(x_j)$, can be calculated as

$$w^{IR}_{t,n}(x_j) = \begin{cases} \frac{IR_{t,n}}{Z_w}, & y_j = -1 \\ \frac{1}{Z_w}, & y_j = +1 \end{cases} \tag{11}$$

where $Z_w = \sum_{j=1}^{a} w^{IR}_{t,n}(x_j)$ is the normalization factor. $IR_{t,n}$, which is imbalance ratio of the union set of training events of the tth and nth ensemble members, can be expressed by

$$IR_{t,n} = \frac{|P_t| + |P_n|}{|N_t| + |N_n|} \tag{12}$$

where $|P_n|$ and $|N_n|$ are the numbers of minority and majority samples in the training set of the nth component classifier, respectively. Because class distributions of previous training blocks are generally imbalanced, minority samples are assigned higher category weights than majority samples. This way, class distributions of previous training chunks are further re-balanced to improve the recognition ratio of ensemble members on minority samples.

After assigning an update weight to each instance of the latest chunk, the costly misclassification instances and minority instances are given higher identification importance. Thus, the data distribution of the latest block is updated. In this paper, we leverage very fast decision tree (VFDT) [35] as component classifiers of SRE. VFDT is an incremental classifier and can be constantly modified without the need of storing examples. Thus, VFDT can be updated conveniently. After obtaining each incoming data block, previous components should be updated using the latest observations with an updated data distribution, i.e., incrementally trained (line 17 of [Algorithm 1]). Although minority instances are very sparse in a dataset with skewed distribution, as the time approaches infinity, data chunks with relatively balanced class distribution can be obtained by resampling the current minority set and periodically updating previous training blocks. It is often assumed that the candidate block implicitly describes the current and near-future concepts. Once the class distribution of a past training chunk has been re-balanced, SRE still performs the periodical update operation, which makes the ensemble quickly react to different types of concept drift. Different from all the existing chunk ensembles, the performance of SRE is robust against chunk size as the training sets of previous classifiers are constantly enlarged by the latest objectives.

3.3. Weighting component classifiers in the ensemble group

In the chunk-based ensemble framework, SRE can leverage past knowledge of data streams in the form of component classifiers. Therefore, SRE has good generalization ability compared with single classifiers. However, an ensemble of a large number of members is likely to accommodate obsolete knowledge. Obsolete members cannot enhance model performance, but will increase the memory and time consumption. In SRE, a hypothesis is built over an amplified block, and c hypotheses are preserved in the ensemble.

The importance of each past component is based on the model performance on the events in the latest chunk. As the minority samples are underrepresented, the previous component classifiers that misclassify minority examples should be penalized more heavily than those members that wrongly identify the class label of majority examples. The weight of the nth component classifier h_n, denoted by w_{h_n} $(1 \le n \le t-1)$, is shown in Eqs. (13)-(15)

$$w_{h_n} = e^{\mathrm{MSE_r} - \mathrm{MSE_n}} \tag{13}$$

$$\mathrm{MSE}_r = \sum_{y_j} p(y_j)(1 - p(y_j))^2 \tag{14}$$

$$\mathrm{MSE}_n = \frac{1}{a} \sum_{(x_j, y_j) \in S_t} C_j (1 - f^n_{y_j}(x_j))^2. \tag{15}$$

The weighting mechanism presented in Eq. (13) combines the elements of classification performance and current class distribution (lines 7–9 of [Algorithm 1]). MSE_n is the mean square error of h_n on the examples in S_t. The weights of previous hypotheses should be reversely proportional to MSE_n. $f^n_{y_j}(x_j)$ is the probability of sample x_j being classified as category y_j by h_n. The data items of the latest chunk are divided into two parts, in which a instances are treated as the training set S_t for building a candidate hypothesis and the remaining b instances are used as the testing set T_t for evaluating model performance. Therefore, the weighting formula presented in Eq. (13) needs to calculate the mean square error of previous members on $x_j \in S_t$. C_j presented in Eq. (10) is the misclassification cost of x_j. The weighting mechanism considers misclassification costs of samples, which is suitable for handling imbalanced datasets. MSE_r presented in Eq. (14) is the mean square error of a randomly predicting classifier. $P(y_j)$ is the y_j's class distribution. A random prediction does not contain any valuable information and is treated as a reference point to the current class distribution.

On the contrary, the weight of the candidate classifier h_t is directly assigned the maximum value, regardless of its performance. This setting is reasonable when assuming the data distribution of the latest chunk is nearest to the current and near-future concepts. The weight of h_t, denoted by w_{h_c}, is formulated as

$$w_{h_c} = e^{\mathrm{MSE}_r} \tag{16}$$

where MSE_r is defined in Eq. (14) (line 6 of [Algorithm 1]). The importance of h_t is proportional to the mean square error of a random prediction. Thus, the weight of the candidate hypothesis is based on the current class distribution. The candidate component is treated as the best member, which especially helps in reacting to sudden changes. Meanwhile, cross-validation of the candidate member can be avoided to handle high-speed data streams.

After obtaining all the weights of ensemble members, the decision of a testing instance $x'_j \in T_t$ can be expressed as

$$\hat{y}'_j = \underset{y'_j \in \{+1,-1\}}{\arg\max} \left(\left(\sum_{n=1}^{t-1} w_{h_n} \times y(h_n(x'_j), y'_j) \right) + \left(w_{h_c} \times y(h_t(x'_j), y'_j) \right) \right) \tag{17}$$

Please cite this article in press as: S. Ren, et al., Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.09.032.

in which $\hat{y}'_j$ is the label of x'_j predicted by the ensemble. $h_n(x'_j)$ and $h_t(x'_j)$ are the labels of x'_j predicted by the nth hypothesis and the candidate hypothesis, respectively. $y(h_n(x'_j), y'_j)$ and $y(h_t(x'_j), y'_j)$ are the indicator functions and can be defined as

$$y(h_n(x'_j), y'_j) = \begin{cases} 1, & \text{if } h_n(x'_j) = y'_j \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

$$y(h_t(x'_j), y'_j) = \begin{cases} 1, & \text{if } h_t(x'_j) = y'_j \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

Therefore, the predictive result of a testing instance is the weighted voting of all the ensemble members (lines 19–21 of [Algorithm 1]).

4. Computational complexity of the proposed method

In the chunk-by-chunk framework, SRE resamples the minority samples of the latest chunk to improve the predictive ability of the candidate component on minority events. The selectively resampling procedure, which simultaneously considers data difficulty factors and concept drift, is used to extract previous minority samples to enlarge the current minority set. First, noise data are identified and removed. Second, the similarities between past minority data and the current minority set are evaluated to remove drifting data. Third, the selection weight of $m_i \in M'$ is computed. Thus, the selectively resampling operation requires the complexity of $O(|M|\log(|M| + a) + a\log a + 3|M|)$ in a data chunk, where a is the size of training chunks and $|M|$ is the number of examples in M. The periodical update operation needs to adjust previous hypotheses using the instances of the latest training chunk, where the update weights of events are evaluated. The cost weight presented in Eq. (8) and the category weight presented in Eq. (11) consume $O(a)$ and $O(1)$ time, respectively. Thus, the complexity of the periodical update operation for each past hypothesis is $O(a)$ every a observations. The weighting settings presented in Eqs. (13) and (16) consume a constant number of operations. The base classifier of SRE is VFDT that can learn a decision tree using constant time per example [36]. Meanwhile, the time consumption of the testing phase is linearly proportional to the number of testing examples. Therefore, the selectively resampling operation is the most time-consuming part of the SRE algorithm. If a data stream contains N data blocks, then the complexity of this procedure is $O(N|M|\log(|M| + a) + aN\log a + 3|M|N)$, but this is just an upper bound.

5. Experimental studies

To observe the properties and performance of SRE, three experiments have been developed:

1. Experiment 1: The goal of this experiment is to prove that SRE is robust against the predefined chunk size. Thus, an experimental analysis of SRE with different chunk sizes is performed in Section 5.4.
2. Experiment 2: The goal of this experiment is to analyze the effect of different imbalance ratios on SRE performance, as discussed in Section 5.5.
3. Experiment 3: The goal of this experiment is to prove that SRE can learn different types of concept drift from imbalanced data streams. In Section 5.6, SRE is compared with several state-of-the-art methods in a wide spectrum of concept drift scenarios.

5.1. Experimental setup

The experimental studies are performed in Java using the massive online analysis (MOA) environment as a test-bed [37]. The tested algorithms in the comparative experiment are listed as follows.

1. AWE. Accuracy weighted ensemble (AWE) [38] is treated as the benchmark algorithm. As a representative of chunk-based ensembles, AWE is strictly designed for nonstationary data streams and has no mechanisms to learn imbalanced data. $c = 10$ component classifiers are preserved in the ensemble group.
2. SERA. In SERA [20], we preserve only minority examples in the previous $w = 30$ blocks to limit the memory usage. The similarity between each of preserved minority examples and the current minority set is evaluated by Mahalanobis distance. Instead of propagating all the preserved minority samples into the candidate block, SERA applies $f = 0.5$ to proportionally select those minority samples with high similarity to re-balance the current skewed distribution. $c = 10$ ensemble members are built over the amplified candidate block.
3. MuSeRA. Compared to SERA, MuSeRA [21] preserves all the hypotheses derived from consecutive blocks, which avoids forgetting knowledge catastrophically (same as suggested by the paper's authors). Only minority examples in previous $w = 30$ blocks are conserved. MuSeRA leverages Mahalanobis distance to measure similarities of preserved examples and the current minority set. The sample size parameter, f, is set to 0.5.
4. SMOTE. A data stream is first divided into a series of fixed-size blocks, then SMOTE [24] is used to adjust the class distribution of the latest block by generating novel minority examples. The post-balance ratio, f, is equal to 0.5. A single classifier based on VFDT is trained over the amplified chunk.
5. UB. UB [19] is one of the earliest algorithms for learning imbalanced data streams with evolving concepts. We collect only minority instances in the previous $w = 30$ blocks. All the preserved minority examples are propagated into the candidate block. Thus, drifting data and complex data distribution are easy to be absorbed into the training sets of ensemble members. $c = 10$ component classifiers are built over the amplified candidate chunk.
6. OOB. OOB [28] is a totally incremental algorithm since no previous data need to be stored. It overcomes class imbalance through resampling and time-decayed metrics. The ensemble size is set to 10 to afford fair comparison. The time-decayed factor is set to 0.9 (same as suggested by the paper's authors).

During the execution of the comparative experiment, the settings of the MOA framework are as follows.

1. The holdout evaluation [39] is treated as the evaluation technique.
2. The base classifier of all tested algorithms is VFDT [35] to afford fair comparison between models. The parameters of VFDT are adopted as default values of the MOA framework.
3. The chunk size d of all chunk-based ensembles, including AWE, SERA, MuSeRA, and UB, is set to 1000. Meanwhile, the selected single classifier is periodically rebuilt every 1000 observations.

SRE, which is a hybrid ensemble, builds components over consecutive blocks and then periodically updates previous ensemble members using observations in the latest training block. Thus, the performance of SRE does not rely on chunk size (discussed in more

Please cite this article in press as: S. Ren, et al., Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.09.032.

detail in Section 5.4). To make comparison fair, $d = 1000$ is the predefined chunk size. The minority examples in the previous $w = 30$ blocks are preserved in M. A selectively resampling technique is used to re-balance the class distribution of the candidate block. In the first phase, a training block with the imbalance ratio $f' = 0.8$ is obtained by the resampling procedure considering the factor of concept drift. In the second phase, SRE resamples the current training block to make the post-balance ratio f equal to 0.5 based on the data difficulty factors. Then, a new component based on VFDT is built over the amplified training block. SRE maintains $c = 10$ components in the ensemble to limit the time and memory usage.

Holdout evaluation can depict the model performance over time, the testing set (holdout set) is extracted from the latest chunk. For each chunk, a total of 70% of instances are used to train a candidate classifier, update previous ensemble members, and evaluate the weights of previous components. The remaining data comprise the testing set. The weight of the candidate classifier does not consider the model performance, and thus no additional sets for validation are required.

5.2. Assessment metrics

In this paper, we only deal with the two-class problems. To evaluate the model performance on imbalanced datasets, the number of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) samples are counted and the confusion matrix can be formed. Accuracy is the standard evaluation metric. It describes the proportion of all correct predictions to the total number of instances. The accuracy performance is primarily dominated by the majority class and is not suitable for learning imbalanced problems.

According to the confusion matrix, several metrics in the field of class imbalance classification have been derived. They are recall_{min}, recall_{maj}, precision, F-measure, G-mean, and AUC. The first two are the average recall on the minority class and majority class, respectively. They are denoted by recall_{min} and recall_{maj}. Precision is the proportion of minority examples that are correctly identified to the examples classified as minority class by a classifier. It measures the ability of a classifier to remove majority observations from all events being misclassified as minority class. F-measure combines recall and precision for maximizing the model performance on a single class. $\beta = 1$ measures the relative importance of recall and precision. The learning object of class imbalance learning is to improve the minority-class recall performance without degrading the precision performance. G-mean is an overall indicator that combines a classifier's performance on minority and majority class recalls. A large value of G-mean indicates that a classifier performs well on samples of both classes. Another popular measure known as AUC is the area under the Receiver Operating Curve [40]. By adjusting the decision threshold of classifiers, AUC provides a single value to evaluate the classification performance. Compared to above metrics, AUC is robust against the distribution between different classes. In our experimental setting, accuracy, F-measure, G-mean, AUC, recall_{min}, and recall_{maj} are selected to analyze the model performance from different aspects.

A data chunk is divided into a training set and a testing set. All the metrics are calculated on periodical holdout sets, thus creating performance curves for each tested algorithm. The observations points are selected within a series of data chunks throughout the life of data streams to observe the model performance in the different context. Meanwhile, a single value of each metric for each tested algorithm, which is the average value over the performance at all time steps, is provided. For each metric, the averages are ranked from (1) to (7), where (1) is the best and (7) is the worst. Then, the Friedman test is used to compare algorithms over all datasets and the Wilcoxon test is used to compare SRE with other comparative algorithms in a pairwise manner [41].

Table 2
Description of the datasets.

Dataset	#Inst	#Attrs	#Classes	Noise	IR	#Drifts	Drift type
SEA_S	667k	3	2	10%	2:23	3	Sudden
SEA_G	558k	3	2	10%	2:23	9	Gradual
Hyper	543k	50	2	5%	2:23	1	Incremental
RanRBF_{GR}	503k	50	2	0%	2:23	4	Gradual recurrent
SEA_{SR}	686k	3	2	10%	2:23	4	Sudden recurrent
Domain_1	1M	2	2	0%	2:23	3	Sudden
Domain_2	1M	2	2	0%	2:23	3	Sudden
Elec	28k	8	2	–	2:23	–	–

5.3. Dataset description

In this section, the details of synthetic and real-world datasets used in the experiment are provided. To research the ability of the proposed algorithm to learn concept drift from imbalanced data, the datasets which exhibit concept drift, class imbalance, and data difficulty factors are designed. Table 2 presents a brief description of each dataset.

There is a shortage of suitable and publicly available real-world benchmark datasets in the field of data stream classification. Therefore, synthetic datasets are generated in the MOA framework. Compared to real-world datasets, the details of synthetic datasets can be obtained in advance. Two families of synthetic datasets are generated. For one thing, two synthetic two-class problem domains, which contain small disjuncts, are generated according to [42]. First, the unit square is divided into $C \times C$ squared grids. The input of each dimension is divided into several equal-size intervals and contiguous intervals have different labels, but the middle grid is kept empty. Fig. 3(a) shows the structure of the Domain_1 dataset with $C = 3$. The complexity level of Domain_1 is equal to 3×3. Second, three types of concepts, which are D_a, D_b, and D_c, are designed. Concept drifts are introduced by changing the categories within grids in addition to those within the middle one from D_a to D_b. Unlike D_a and D_b, the sample attributes of D_c are in the interval [0, 3]. Concept drifts occur each 250,000 observations, and thus 4 concepts and 3 concept changes are on the Domain_1 dataset. Third, we increase the data complexity level to 5×5 on the Domain_2 dataset. In Fig. 3(b), D'_a, D'_b, and D'_c are designed. Concept drifts occur each 250,000 objects on the Domain_2 dataset. The labels of D'_b are reversed compared to the same positions of D'_a. Meanwhile, the sample features in D'_c are adjusted to the interval [0, 5]. Finally, the generation probability of majority samples is 11.5 times that of minority samples to create two datasets with IR = 2:23 for each data chunk.

For another, five datasets are generated by data stream generators available in the MOA framework to analyze a algorithm's ability to learn different types of concept drift. The details of them are presented as follows.

1. SEA. The SEA generator [43] is used to create three datasets, 10% of noise each. It contains three attributes ranging from 0 to 10. Only the first two attributes are relevant and the third attribute is treated as noise. The class labels are determined by comparing the sum of two relevant attributes and a predefined threshold. Concept drifts are simulated by adjusting the threshold. First, SEA_S contains three sudden drifts. Second, nine gradual drifts are brought on the SEA_G dataset. Third, SEA_{SR} contains four sudden recurrent drifts, where the fifth concept is the recurrent concept of the first one. Then, we undersample one of the classes every 1000 instances to induce class imbalance. The cardinality of minority class is 8 percent of total data.
2. Hyper. The Hyperplane generator [35] is used to create the Hyper dataset of 543,000 instances described by 50

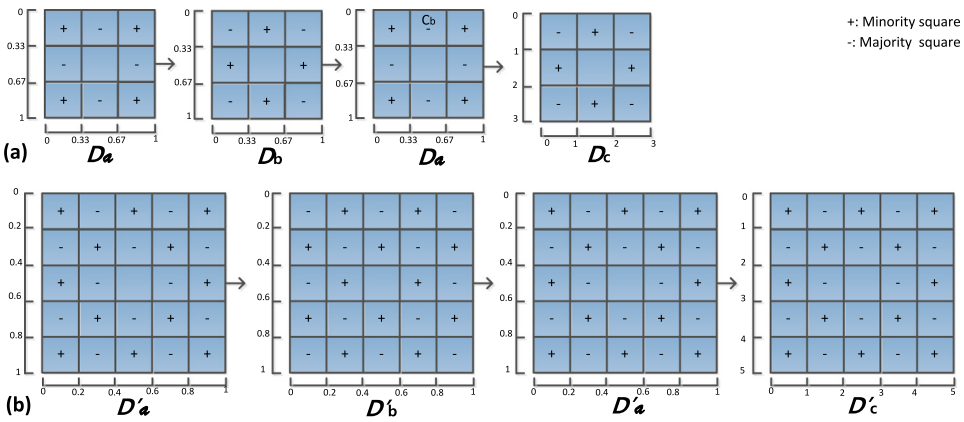

Fig. 3. Structures of the first family synthetic datasets. (a) Domain$_1$ and (b) Domain$_2$.

attributes and two classes. The Hyper dataset contains an incremental drift by changing the decision hyperplane gradually. We then induce class imbalance by undersampling one of the classes every 1000 instances. The minority class consists of 8 percent of the total data size.

3. RanRBF. The radial basis function (RBF) generator creates several drifting centroids, each defined by a class label, weight, position, and standard deviation. We use this generator to create the RanRBF$_{GR}$ dataset containing four gradual recurrent drifts. RanRBF$_{GR}$ consists of 503,000 instances described by 50 attributes and two classes. One of the classes is undersampled to create an imbalance ratio of 2 : 23 every 1000 observations.

In addition to synthetic datasets, we analyze the model performance in the real-life scenario. The Electricity Market data (Elec), which is collected from the electricity market in New South Wales, Australian, contains the changes of electricity price according to time and demand [44]. Since the original dataset does not contain class imbalance, the examples which represent the rising prices are randomly undersampled and treated as the minority samples. Elec contains 28,000 observations and the minority samples are 8 percent of the total data size.

5.4. Experiment 1: study of the impact of chunk size

The goal of this experiment is to validate the robustness of SRE to chunk size d. In SRE, the frequency of updating previous ensemble members can be controlled by the predefined chunk size. Once a new block arrives, previous hypotheses are updated using instances in this block to make the ensemble quickly adapt to concept drift. The performance of chunk-based ensembles generally relies on chunk size. Chunk ensembles can be divided into two groups. The first group methods, such as UB [19] and SERA [20], build all ensemble members over the current block. Thus, the chunk size directly determines the number of training instances. The second group methods, such as AWE [38] and MuSeRA [21], build a component classifier over each block and then all classifiers are combined to form an ensemble. Using small-size chunks will result in insufficient training data for component classifiers, which degrades the model performance under stationary conditions. On the contrary, concept drifts are easy to be involved in a big-size block.

SRE is treated as a hybrid ensemble [17]. A candidate hypothesis is first built over the latest block, then the latest instances are weighted and used to update previous hypotheses. Thus, SRE constantly enlarges the past training chunks. The predefined chunk size cannot have a significant effect on SRE performance. Table 3 summarizes the AUC performances (means and standard deviations) and average ranks of SRE with different chunk sizes on all datasets, averaged over 50 runs. The chunk size is varied from 400 to 2000 and the best result is highlighted with bold-face type. Following the suggestion in [45], we apply the Friedman test to compare the different settings over multiple datasets. The null-hypothesis for this test is that there are no significant differences among different variants of SRE. Then, $F_F = -0.936$ is calculated. If the significant level is 0.05, then the null-hypothesis is accepted. This indicates that differences are not significant and no post-hoc analyses are required. Therefore, we can conclude that the proposed algorithm is robust against chunk size. $d = 1000$ is selected as the default value in the comparative experiment.

5.5. Experiment 2: study of the impact of imbalance ratio

The goal of this experiment is to analyze the impact of different imbalance ratios on the performance of the proposed algorithm. The imbalance ratio of data, which is the ratio between the number of minority and majority examples, can directly affect the performance of models. A smaller imbalance ratio means a smaller probability to obtain the minority examples, and thus the classification task becomes much harder. In this experiment, we modify the imbalance ratios of datasets presented in Table 2. We consider five imbalance levels, e.g., 3:47, 7:93, 2:23, 9:91, and 1:9 for each dataset. Thus, the cardinalities of minority class are 6%, 7%, 8%, 9%, and 10% of total data, respectively.

Table 4 shows the AUC performances (means and standard deviations) and average ranks of SRE on datasets with different imbalance ratios. Each value is the average of 50 runs and the best result is highlighted with bold-face type. We also give a statistical analysis of how the AUC performance of SRE is affected by the imbalance ratio of datasets through factorial ANOVA [45]. Then, $p = 0.000$ and $\eta^2 = 0.99$ are obtained. If the significant level is 0.05, then we can conclude that the factor of imbalance ratio has a significant effect on the AUC performance of SRE. This is because the classification task becomes much harder when the percentage of minority examples decreases. The post-balance ratio f remains fixed, and thus more minority examples need to be selected from the past preserved minority data for resampling the training set of the candidate classifier if the imbalance ratio of datasets is small. As the resampling mechanism is constantly applied to the consecutive blocks, the training sets of component classifiers become similar. Thus, the diversity of ensemble members is reduced, which degrades the classification performance of SRE.

5.6. Experiment 3: comparative experiment

In this section, we compare SRE with six state-of-the-art methods on accuracy, F-measure, G-mean, recall$_{min}$, AUC, and recall$_{maj}$.

Please cite this article in press as: S. Ren, et al., Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.09.032.

Table 3
AUC performance of SRE with different chunk sizes d on all datasets (mean ± standard variance(rank)).

	$d = 400$	$d = 800$	$d = 1000$	$d = 1200$	$d = 1600$	$d = 2000$
SEA_S	87.59 ± 0.11(6)	87.76 ± 0.00(2)	87.73 ± 0.10(3)	87.70 ± 0.00(4)	**87.77 ± 0.00(1)**	87.70 ± 0.00(4)
SEA_G	**89.28 ± 0.11(1)**	89.22 ± 0.10(5)	**89.28 ± 0.00(1)**	88.25 ± 0.11(3)	89.21 ± 0.00(6)	89.23 ± 0.00(4)
Hyper	81.58 ± 0.11(3)	81.31 ± 0.12(6)	81.78 ± 0.11(2)	**81.82 ± 0.10(1)**	81.49 ± 0.10(5)	81.56 ± 0.11(4)
$RanRBF_{GR}$	99.80 ± 0.00(5)	99.80 ± 0.00(5)	99.83 ± 0.00(3)	99.81 ± 0.00(4)	99.85 ± 0.00(2)	**99.86 ± 0.00(1)**
SEA_{SR}	87.46 ± 0.11(5)	87.55 ± 0.00(2)	87.51 ± 0.10(3)	**87.58 ± 0.11(1)**	87.51 ± 0.10(3)	87.45 ± 0.11(6)
$Domain_1$	99.71 ± 0.00(3)	99.70 ± 0.00(4)	**99.75 ± 0.00(1)**	99.64 ± 0.00(5)	99.61 ± 0.00(6)	99.72 ± 0.00(2)
$Domain_2$	98.57 ± 0.00(4)	**98.65 ± 0.00(1)**	98.63 ± 0.00(3)	98.48 ± 0.00(6)	98.53 ± 0.00(5)	**98.65 ± 0.00(1)**
Elec	**88.25 ± 0.77(1)**	85.95 ± 0.55(2)	85.73 ± 0.78(3)	85.24 ± 0.88(4)	84.50 ± 0.82(5)	83.64 ± 0.50(6)
Average rank	3.500	3.375	2.375	3.500	4.125	3.500

The best result is in boldface.

Table 4
Effect of different imbalance ratios on the AUC performance of SRE (mean ± standard variance(rank)).

	IR=3:47	IR=7:93	IR=2:23	IR=9:91	IR=1:9
SEA_S	87.60 ± 0.11(5)	87.67 ± 0.10(4)	87.73 ± 0.10(2)	**87.75 ± 0.00(1)**	87.73 ± 0.00(2)
SEA_G	88.95 ± 0.10(5)	89.27 ± 0.10(4)	89.28 ± 0.00(3)	89.40 ± 0.00(2)	**89.43 ± 0.10(1)**
Hyper	81.70 ± 0.13(5)	81.77 ± 0.13(4)	81.78 ± 0.11(3)	81.86 ± 0.10(2)	**81.98 ± 0.11(1)**
$RanRBF_{GR}$	99.82 ± 0.00(5)	99.84 ± 0.00(3)	99.83 ± 0.00(4)	99.85 ± 0.00(2)	**99.86 ± 0.00(1)**
SEA_{SR}	87.46 ± 0.11(5)	87.51 ± 0.10(3)	87.51 ± 0.10(3)	**87.70 ± 0.11(1)**	87.68 ± 0.10(2)
$Domain_1$	99.74 ± 0.00(4)	99.74 ± 0.00(4)	99.75 ± 0.00(2)	99.75 ± 0.00(2)	**99.79 ± 0.00(1)**
$Domain_2$	98.29 ± 0.00(5)	98.32 ± 0.00(4)	98.63 ± 0.00(3)	98.86 ± 0.00(2)	**98.89 ± 0.00(1)**
Elec	83.36 ± 0.77(5)	83.54 ± 0.75(4)	85.73 ± 0.78(3)	85.87 ± 0.88(2)	**87.17 ± 0.82(1)**
Average rank	4.875	3.750	2.875	1.750	1.250

The best result is in boldface.

5.6.1. Experimental results

Table 5 provides the average performances (means ± standard variances) and ranks of algorithms on all datasets, averaged over 50 runs. Meanwhile, the performance curves of algorithms are visualized in Figs. 4–6. For the space consideration, we only present the graphical plots of tested algorithms on the SEA_S, $RanRBF_{GR}$, and $Domain_1$ datasets. Meanwhile, the graphical plots of comparative algorithms on other datasets are presented in the supplementary material.

Fig. 4 shows the results on the SEA_S datasets. There are several observations we can make from these results. First, sudden changes incur drops in most of the metrics of nearly all algorithms. SERA and SMOTE maintain constant accuracies for nearly all time stamps. This is because SMOTE leverages only the latest examples to build a classifier and SERA does not retain any prior knowledge about the majority class. Second, we observe that while OOB maintains the best rank for accuracy (closely followed by SRE), it drops to rank 5 for minority class recall. AWE has good accuracy, $recall_{maj}$, and AUC performances but ranks the lowest in terms of F-measure, G-mean, and $recall_{min}$ because it lacks a mechanism to handle imbalanced datasets. Third, UB seems to be the most robust in terms of minority class recall, with SMOTE and SRE catching up in the latter part of the simulation. However, the boost in $recall_{min}$ for UB comes at the cost of accuracy and $recall_{maj}$. This is a common trend for UB on nearly all datasets tested as well. Finally, SMOTE effectively improves the model performance on minority examples by generating novel minority examples. Thus, SMOTE obtains good $recall_{min}$ and G-mean performances. However, it performs poorly on accuracy, F-measure, $recall_{maj}$, and AUC. It is should be noted that our goal is to improve the minority class performance without significantly sacrificing the majority class performance. SRE has the best rank for F-measure and G-mean. Meanwhile, it does well on accuracy, $recall_{min}$, $recall_{maj}$, and AUC.

On the dataset with gradual drifts (SEA_G), we do observe several trends which seem to be same as those observed on the SEA_S dataset. First, SRE outranks other algorithms in terms of F-measure and G-mean. Second, UB is the best-performing algorithm in terms of $recall_{min}$ (followed by SRE), which causes a large drop in $recall_{maj}$, accuracy, and F-measure. This is because UB propagates all the preserved minority examples into the training sets of component classifiers. Finally, SMOTE's good showing on $recall_{min}$ and G-mean comes at the cost of very poor $recall_{maj}$, accuracy, and AUC. A few additional observations: First, AWE has the best classification accuracy and $recall_{maj}$ performances, but primarily due to its performance on majority examples. However, AWE ranks the lowest in terms of F-measure, G-mean, and $recall_{min}$ because it has no mechanisms to handle imbalanced datasets. Second, the accuracy performance of MuSeRA is better than that of SERA. This is because MuSeRA preserves all the component classifiers built over consecutive blocks. However, SERA leverages only the latest examples to train all the components. Thus, MuSeRA can make full use of relevant knowledge of data streams compared with SERA. Third, while OOB has good accuracy and AUC performances, it performs poorly on $recall_{min}$ and G-mean. Finally, SRE is highly competitive with other algorithms in terms of AUC.

On the dataset with an incremental drift (Hyper), SRE performs consistently well in terms of F-measure, G-mean, and AUC. UB provides the best minority class recall (closely followed by SRE) but drops to rank 7, 7, and 5 for accuracy, $recall_{maj}$, and F-measure, respectively. While OOB is the best-performing algorithm in terms of accuracy and $recall_{maj}$, it has the lowest rank for F-measure, G-mean, and $recall_{min}$. Meanwhile, lacking a mechanism to accommodate imbalanced datasets, AWE has good accuracy, AUC, and $recall_{maj}$ performances but performs rather poorly on other figures of merit.

Fig. 5 shows the results on the $RanRBF_{GR}$ dataset. Recurrent concept drift is brought to validate whether the tested algorithms can use old information stored in the ensemble to improve the model performance. Several observations should be noticed. First, OOB performs particularly well on the $RanRBF_{GR}$ dataset. It maintains rank 2 for accuracy, F-measure, G-mean, $recall_{min}$, and AUC. Meanwhile, it has the best rank for $recall_{maj}$. Second, the resampling mechanism based on Mahalanobis distance fails to improve the model performance on minority examples in SERA and MuSeRA. With regard to $recall_{min}$, SERA obtains rank 5 and MuSeRA ranks the lowest. Finally, SRE maintains a series of ensemble members, which can reactive past classifiers corresponding to the recurrent concepts. Then, it obtains the best rank for accuracy, F-measure, G-mean, $recall_{min}$, and AUC. Meanwhile, SRE maintains the second rank for $recall_{maj}$. Thus, SRE provides a good balance in accuracy, F-measure, G-mean, $recall_{min}$, AUC, and $recall_{maj}$.

Please cite this article in press as: S. Ren, et al., Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.09.032.

Table 5
Performance comparison of different algorithms on all datasets (mean ± standard variance(rank)).

Datasets	Methods	Accuracy	F-measure	G-mean	$Recall_{min}$	AUC	$Recall_{maj}$
SEA_S	AWE	92.26 ± 0.00(3)	22.72 ± 0.50(7)	36.96 ± 0.42(7)	14.55 ± 0.40(7)	**87.89 ± 0.12(1)**	**98.99 ± 0.01(1)**
	SERA	89.92 ± 0.72(5)	53.61 ± 0.88(3)	79.88 ± 0.35(3)	70.16 ± 0.46(4)	84.69 ± 0.11(4)	92.05 ± 0.72(5)
	MuSeRA	91.61 ± 0.18(4)	50.16 ± 0.96(4)	70.19 ± 1.08(5)	56.65 ± 0.95(6)	81.72 ± 0.53(6)	93.65 ± 0.20(4)
	SMOTE	83.87 ± 0.59(6)	44.42 ± 0.64(5)	80.03 ± 0.19(2)	76.29 ± 0.68(2)	72.99 ± 0.43(7)	84.40 ± 0.60(6)
	UB	44.83 ± 0.69(7)	23.36 ± 0.29(6)	58.07 ± 0.58(6)	**93.03 ± 0.17(1)**	84.61 ± 0.16(5)	42.90 ± 0.65(7)
	OOB	**93.05 ± 0.00(1)**	58.81 ± 0.59(2)	76.98 ± 0.56(4)	62.68 ± 0.89(5)	87.71 ± 0.12(3)	93.68 ± 0.04(3)
	SRE	92.52 ± 0.12(2)	**62.22 ± 0.37(1)**	**84.35 ± 0.52(1)**	76.13 ± 1.04(3)	87.73 ± 0.10(2)	94.04 ± 0.10(2)
SEA_G	AWE	**91.82 ± 0.00(1)**	19.28 ± 0.58(7)	33.94 ± 0.47(7)	12.60 ± 0.49(7)	89.18 ± 0.00(3)	**98.76 ± 0.02(1)**
	SERA	87.97 ± 0.22(5)	50.38 ± 0.43(2)	81.32 ± 0.44(3)	74.75 ± 0.55(4)	86.20 ± 0.14(5)	89.55 ± 0.22(5)
	MuSeRA	90.21 ± 0.00(3)	45.14 ± 2.30(4)	67.85 ± 3.30(5)	56.97 ± 2.90(5)	83.42 ± 0.43(6)	90.20 ± 0.02(4)
	SMOTE	84.20 ± 0.25(6)	45.37 ± 0.30(3)	81.52 ± 0.39(2)	79.05 ± 0.87(3)	71.26 ± 0.50(7)	84.05 ± 0.22(6)
	UB	66.84 ± 0.91(7)	32.40 ± 0.45(6)	76.40 ± 0.60(4)	**91.63 ± 0.27(1)**	87.57 ± 0.14(4)	65.58 ± 0.82(7)
	OOB	91.36 ± 0.00(2)	43.29 ± 0.71(5)	62.76 ± 1.09(6)	42.43 ± 1.60(6)	89.24 ± 0.11(2)	90.59 ± 0.05(3)
	SRE	90.01 ± 0.13(4)	**56.53 ± 0.34(1)**	**85.14 ± 0.33(1)**	80.22 ± 0.60(2)	**89.28 ± 0.00(1)**	90.81 ± 0.02(2)
Hyper	AWE	91.63 ± 0.10(2)	7.49 ± 1.52(6)	17.23 ± 2.79(6)	4.52 ± 0.96(6)	80.80 ± 0.11(2)	99.20 ± 0.12(2)
	SERA	83.85 ± 0.33(6)	35.82 ± 0.29(2)	69.13 ± 0.46(2)	56.40 ± 0.85(3)	77.43 ± 0.23(4)	86.24 ± 0.30(6)
	MuSeRA	87.42 ± 0.13(3)	29.35 ± 0.42(3)	54.07 ± 0.46(4)	33.92 ± 0.82(4)	66.14 ± 0.24(6)	92.07 ± 0.15(4)
	SMOTE	87.20 ± 0.38(4)	25.22 ± 0.52(4)	49.33 ± 0.52(5)	28.52 ± 1.12(5)	62.11 ± 0.90(7)	92.31 ± 0.32(3)
	UB	38.65 ± 1.79(7)	18.30 ± 0.46(5)	54.63 ± 1.32(3)	**91.04 ± 0.53(1)**	78.05 ± 0.22(3)	34.44 ± 1.80(7)
	OOB	**91.99 ± 0.00(1)**	0.28 ± 0.75(7)	0.80 ± 1.86(7)	0.15 ± 0.45(7)	73.83 ± 0.18(5)	**99.98 ± 0.01(1)**
	SRE	84.69 ± 0.18(5)	**39.81 ± 0.25(1)**	**73.51 ± 0.31(1)**	63.65 ± 0.72(2)	**81.78 ± 0.11(1)**	86.52 ± 0.12(5)
$RanRBF_{GR}$	AWE	97.06 ± 0.10(3)	77.86 ± 1.31(3)	82.37 ± 1.18(4)	69.37 ± 1.64(4)	98.33 ± 0.25(3)	99.47 ± 0.13(3)
	SERA	95.98 ± 0.20(5)	70.36 ± 0.75(4)	79.30 ± 0.50(5)	66.13 ± 0.82(5)	86.17 ± 0.20(6)	98.35 ± 0.22(4)
	MuSeRA	93.95 ± 0.12(6)	60.46 ± 0.75(6)	73.56 ± 1.11(7)	58.80 ± 1.44(7)	87.61 ± 0.23(5)	96.90 ± 0.12(6)
	SMOTE	93.19 ± 0.39(7)	62.45 ± 0.79(5)	78.30 ± 0.25(6)	65.92 ± 0.20(6)	80.25 ± 0.25(7)	95.52 ± 0.32(7)
	UB	96.96 ± 0.10(4)	59.60 ± 0.33(7)	94.21 ± 0.15(3)	93.42 ± 0.30(3)	95.03 ± 0.25(4)	97.00 ± 0.22(5)
	OOB	99.67 ± 0.00(2)	97.68 ± 0.53(2)	97.92 ± 0.13(2)	96.28 ± 0.71(2)	99.41 ± 0.01(2)	**99.97 ± 0.00(1)**
	SRE	**99.76 ± 0.00(1)**	**98.39 ± 0.33(1)**	**98.72 ± 0.16(1)**	**97.74 ± 0.59(1)**	**99.83 ± 0.00(1)**	99.94 ± 0.01(2)
SEA_{SR}	AWE	92.33 ± 0.00(3)	24.90 ± 1.07(6)	39.19 ± 1.06(7)	16.28 ± 0.93(7)	**87.61 ± 0.00(1)**	**98.95 ± 0.03(1)**
	SERA	90.41 ± 0.23(5)	54.24 ± 0.41(3)	79.76 ± 0.56(3)	69.41 ± 1.12(4)	84.57 ± 0.16(4)	92.44 ± 0.22(5)
	MuSeRA	91.83 ± 0.10(4)	50.71 ± 1.03(4)	70.01 ± 1.22(5)	56.42 ± 1.29(6)	81.63 ± 0.36(6)	95.25 ± 0.20(4)
	SMOTE	83.74 ± 0.36(6)	44.06 ± 0.34(5)	79.90 ± 0.26(2)	76.18 ± 0.62(2)	72.38 ± 0.31(7)	84.72 ± 0.36(6)
	UB	40.19 ± 1.39(7)	21.20 ± 0.32(7)	55.59 ± 1.34(6)	**93.10 ± 0.49(1)**	84.20 ± 0.18(5)	35.80 ± 1.35(7)
	OOB	**93.35 ± 0.00(1)**	60.65 ± 0.80(2)	78.38 ± 0.97(4)	64.72 ± 1.56(5)	87.48 ± 0.12(3)	95.83 ± 0.02(3)
	SRE	92.86 ± 0.10(2)	**63.04 ± 0.38(1)**	**84.29 ± 0.19(1)**	75.63 ± 0.37(3)	87.51 ± 0.10(2)	95.93 ± 0.12(2)
$Domain_1$	AWE	92.00 ± 0.00(4)	0.00 ± 0.00(7)	0.00 ± 0.00(7)	0.00 ± 0.00(7)	81.25 ± 0.79(4)	**100.00 ± 0.00(1)**
	SERA	92.12 ± 0.24(5)	37.10 ± 0.59(4)	54.44 ± 0.65(5)	33.71 ± 0.54(5)	75.56 ± 0.23(6)	96.56 ± 0.22(5)
	MuSeRA	92.06 ± 0.00(3)	12.36 ± 0.43(6)	21.07 ± 0.40(6)	7.80 ± 0.28(6)	77.10 ± 0.11(5)	99.39 ± 0.02(3)
	SMOTE	74.04 ± 0.28(7)	28.36 ± 0.00(5)	68.00 ± 0.17(4)	62.42 ± 0.40(4)	50.33 ± 0.12(7)	75.16 ± 0.25(7)
	UB	81.45 ± 0.54(6)	54.01 ± 0.92(3)	86.96 ± 0.35(3)	**99.16 ± 0.11(1)**	95.40 ± 0.17(3)	80.66 ± 0.55(6)
	OOB	98.38 ± 0.33(2)	91.62 ± 1.66(2)	94.32 ± 1.13(2)	91.14 ± 1.70(3)	98.12 ± 0.32(2)	98.95 ± 0.12(4)
	SRE	**99.63 ± 0.00(1)**	**98.06 ± 0.29(1)**	**99.03 ± 0.00(1)**	98.55 ± 0.16(2)	**99.75 ± 0.00(1)**	99.66 ± 0.02(2)
$Domain_2$	AWE	92.00 ± 0.00(3)	0.00 ± 0.00(7)	0.00 ± 0.00(7)	0.00 ± 0.00(7)	55.52 ± 0.13(4)	**100.00 ± 0.00(1)**
	SERA	88.32 ± 0.26(5)	11.23 ± 0.60(5)	26.65 ± 1.05(5)	10.35 ± 0.63(5)	53.95 ± 0.18(7)	95.78 ± 0.20(5)
	MuSeRA	91.52 ± 0.00(4)	2.46 ± 0.21(6)	6.08 ± 0.50(6)	1.76 ± 0.12(6)	54.24 ± 0.14(5)	99.28 ± 0.02(3)
	SMOTE	60.79 ± 0.24(6)	14.07 ± 0.11(4)	47.10 ± 0.41(4)	42.29 ± 0.72(4)	53.99 ± 0.00(6)	62.48 ± 0.25(6)
	UB	36.65 ± 1.01(7)	20.39 ± 0.29(3)	52.81 ± 1.10(3)	**98.19 ± 0.00(1)**	76.62 ± 0.25(3)	31.87 ± 1.42(7)
	OOB	95.63 ± 0.31(2)	76.96 ± 1.75(2)	79.20 ± 1.52(2)	68.52 ± 2.16(3)	92.09 ± 0.41(2)	98.12 ± 0.32(4)
	SRE	**98.84 ± 0.00(1)**	**94.16 ± 0.16(1)**	**96.00 ± 0.23(1)**	93.49 ± 0.38(2)	**98.63 ± 0.00(1)**	99.32 ± 0.02(2)
Elec	AWE	90.76 ± 1.30(5)	43.78 ± 3.12(3)	62.67 ± 3.44(4)	44.39 ± 3.66(4)	83.22 ± 1.41(3)	96.16 ± 1.23(4)
	SERA	93.03 ± 1.08(3)	42.71 ± 3.29(4)	55.04 ± 6.05(5)	35.45 ± 4.33(5)	72.63 ± 2.10(5)	**99.07 ± 0.12(1)**
	MuSeRA	89.71 ± 0.35(6)	25.63 ± 4.42(7)	32.16 ± 6.10(7)	25.10 ± 3.18(7)	62.80 ± 3.84(7)	95.10 ± 0.32(6)
	SMOTE	83.17 ± 2.22(7)	40.76 ± 0.96(5)	**73.41 ± 0.82(1)**	**65.30 ± 3.06(1)**	71.44 ± 2.05(6)	86.19 ± 2.20(7)
	UB	92.18 ± 1.94(4)	46.07 ± 7.72(2)	68.85 ± 4.60(3)	51.32 ± 4.80(3)	81.97 ± 1.45(4)	95.96 ± 1.92(5)
	OOB	93.50 ± 0.17(2)	40.67 ± 2.40(6)	52.44 ± 2.65(6)	30.82 ± 2.18(6)	84.53 ± 1.55(2)	99.04 ± 0.15(2)
	SRE	**93.90 ± 0.73(1)**	**46.77 ± 1.81(1)**	70.33 ± 3.23(2)	53.53 ± 5.30(2)	**85.73 ± 0.78(1)**	96.78 ± 0.42(3)

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

For the dataset with sudden recurrent drifts (SEA_{SR}), the fifth concept is the recurrent concept of the first one. We first observe that SRE outranks other algorithms in terms of F-measure and G-mean. Meanwhile, it maintains rank 2 for accuracy, AUC, and $recall_{maj}$. Second, UB outperforms all other algorithms in terms of minority class recall (followed by SMOTE and SRE) because it blindly absorbs all the preserved minority examples into the training sets of component classifiers. However, UB has the worst rank for accuracy, F-measure, and $recall_{maj}$. Thus, UB's strong performance on minority examples comes at the cost of very poor performance on majority examples. Third, OOB is the top ranking algorithm in terms of accuracy (followed by SRE), but primarily due to its performance on majority examples. However, OOB experiences a large drop in rank when using $recall_{min}$ for evaluation. Fourth, AWE is the best-performing algorithm in terms of AUC and $recall_{maj}$ (followed by SRE) but has very poor performance in terms of F-measure, G-mean, and $recall_{min}$. Finally, we observe, again, that SMOTE has good $recall_{min}$ and G-mean performances but performs poorly on accuracy, AUC, and $recall_{maj}$.

On the $Domain_1$ and $Domain_2$ datasets, we analyze algorithms' ability to handle the within-class imbalance problem. Sudden drifts and small disjuncts are involved on these two synthetic datasets. Table 5 summarizes the results of all comparative algorithms on $Domain_1$ and $Domain_2$. Fig. 6 presents the results on the $Domain_1$ dataset. Most of the algorithms experience major drops in every

Please cite this article in press as: S. Ren, et al., Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.09.032.

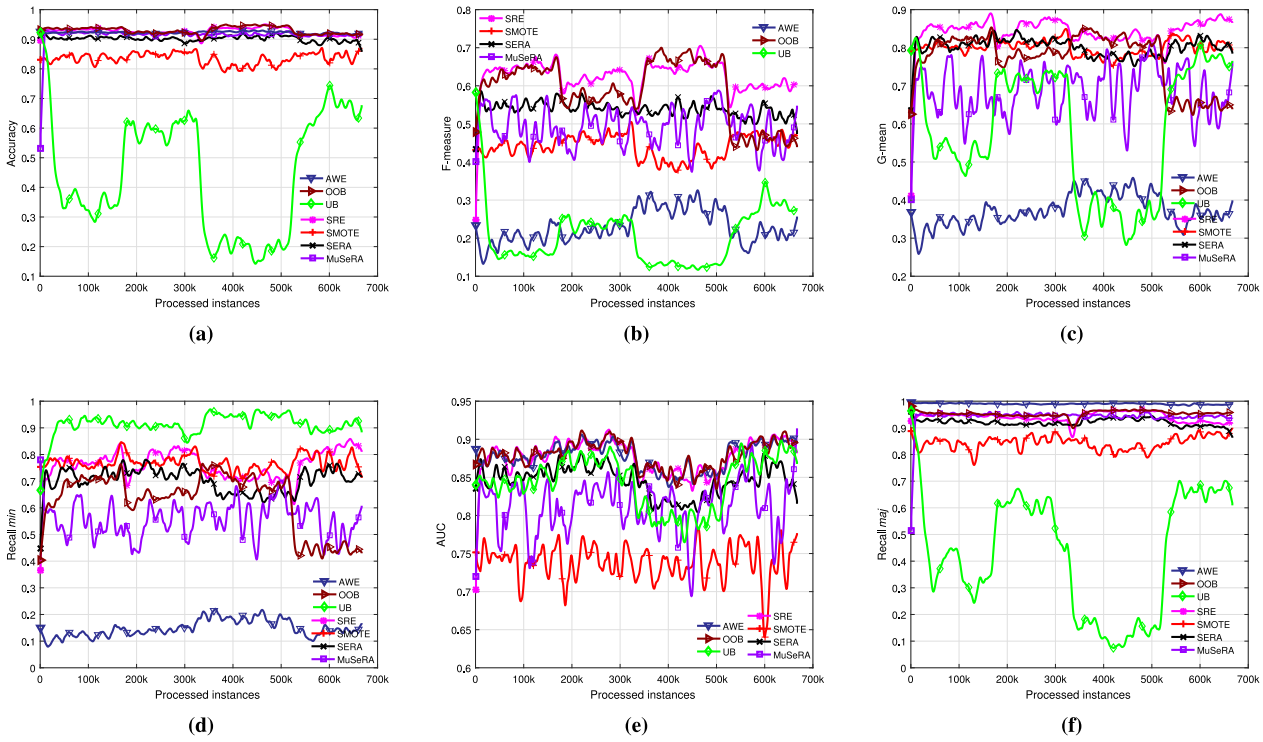

Fig. 4. Classification performance on SEA_S. (a) Accuracy, (b) F-measure, (c) G-mean, (d) $Recall_{min}$, (e) AUC, and (f) $Recall_{maj}$.

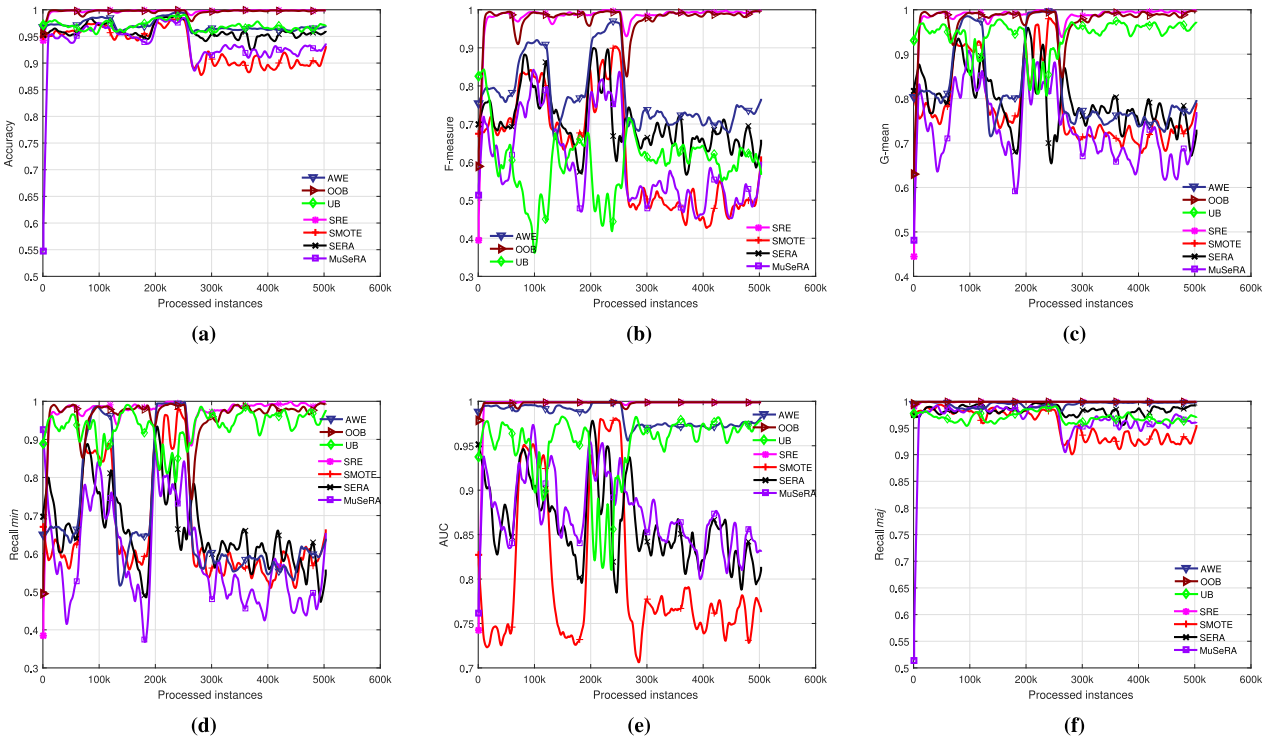

Fig. 5. Classification performance on $RanRBF_{GR}$. (a) Accuracy, (b) F-measure, (c) G-mean, (d) $Recall_{min}$, (e) AUC, and (f) $Recall_{maj}$.

measure around time steps 250 000, 500 000, and 750 000, which precisely correspond to the positions of concept drift. We can observe that SRE maintains good speedy recovery from sudden drifts and a high level of performance during the stationary periods. On $Domain_1$, we first observe that AWE misclassifies all minority examples as majority class. Thus, AWE drops to rank 7 for F-measure, G-mean, and $recall_{min}$. Second, UB's strong $recall_{min}$ has deteriorated the classification accuracy and $recall_{maj}$. Third, we do observe that OOB can well adapt to the complex data distribution. Through the oversampling technique and time-decayed metrics, OOB performs consistently well across all figures of merit. Followed by SRE, OOB obtains rank 2 for accuracy, F-measure, G-mean, and AUC. Finally, SRE is highly competitive with other algorithms on the $Domain_1$ dataset. This is because the resampling

Please cite this article in press as: S. Ren, et al., Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.09.032.

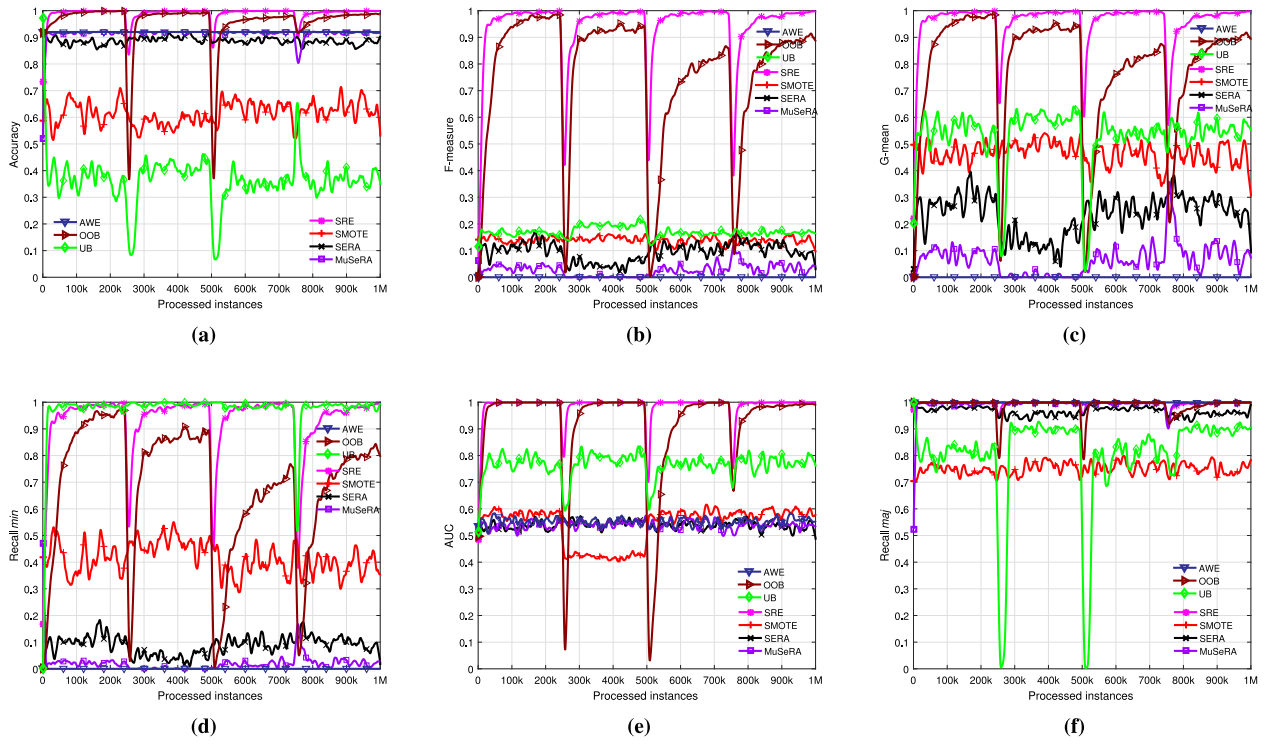

Fig. 6. Classification performance on Domain$_1$. (a) Accuracy, (b) F-measure, (c) G-mean, (d) Recall$_{min}$, (e) AUC, and (f) Recall$_{maj}$.

mechanism of SRE takes concept drift and data difficulty factors into account. SRE is the winner of accuracy, F-measure, G-mean, and AUC. Meanwhile, it maintains rank 2 for recall$_{min}$ and recall$_{maj}$. On the Domain$_2$ dataset, the complexity level increases from 3×3 to 5×5. We do observe that almost all algorithms exhibit degraded performance in most of the metrics. As in the Domain$_1$ dataset, SRE outperforms all other algorithms in terms of accuracy, F-measure, G-mean, and AUC (followed by OOB). UB has the best recall$_{min}$ (followed by SRE), but ranks the lowest in terms of classification accuracy and recall$_{maj}$. AWE fails to classify any minority examples and performs particularly poorly with the worst rank in terms of F-measure, G-mean, and recall$_{min}$.

On the Elec dataset, the average values of all evaluation metrics and ranks of algorithms are shown in Table 5. A few observations on this dataset: First, SMOTE has the best G-mean and recall$_{min}$ performances by generating novel minority examples to re-balance the current class distribution, followed by SRE. However, SMOTE performs poorly on other figures of merit used in the evaluation. Second, UB has good accuracy but performs poorly in terms of recall$_{min}$ compared with its performance on nearly all other datasets. This is mainly because the data size of Elec is much smaller than that of other datasets. Thus, the size of minority examples is not easy to exceed that of the current majority set after the oversampling procedure. Finally, SRE has the best rank for accuracy, F-measure, and AUC. Followed by SMOTE, SRE obtains rank 2 for G-mean and recall$_{min}$. Meanwhile, SRE maintains rank 3 for recall$_{maj}$. Thus, SRE improves the minority class performance without significantly deteriorating the majority class performance on the Elec dataset.

5.6.2. Summary of comparative results

The algorithms' ranks in terms of each metric on each dataset are provided in Table 5. Then, these ranks are averaged to compute the mean ranks of comparative algorithms on all datasets considering each evaluation metric, as shown in Table 6. Smaller value indicates a higher rank of performance. In addition to the detailed analyses of models described in Section 5.6.1, we summarize key observations that are consistent across a wide range of datasets:

Table 6
Mean ranking results of comparative methods over all datasets.

Rank	AWE	SERA	MuSeRA	SMOTE	UB	OOB	SRE
Accuracy	3.000	4.875	4.125	6.125	6.125	1.625	2.125
F-measure	5.750	3.375	5.000	4.500	4.875	3.500	1.000
G-mean	6.125	3.875	5.625	3.250	3.875	4.125	1.125
Recall$_{min}$	6.125	4.375	5.875	2.375	1.500	4.625	2.125
AUC	2.625	5.125	5.750	6.750	3.875	2.625	1.250
Recall$_{maj}$	1.750	4.500	4.250	6.000	6.375	2.625	2.500

- SRE maintains the best mean rank in terms of F-measure, G-mean, and AUC on all datasets. We do observe that SRE achieves the highest F-measure on all datasets. Meanwhile, SRE outperforms other algorithms in terms of G-mean and AUC in nearly all cases. With regard to accuracy, recall$_{min}$, and recall$_{maj}$, SRE obtains the second mean rank. Thus, SRE improves the minority class performance without the cost of severely damaging the majority class performance.
- UB shows the best mean rank for recall$_{min}$ (followed by SRE), but that comes at the cost of accuracy and recall$_{maj}$. Meanwhile, UB is a poor-performing classifier in terms of F-measure and AUC. This is because UB blindly propagates all the preserved minority samples into the training sets of new classifiers.
- AWE has good performance on majority samples. Thus, it obtains good mean rank in terms of accuracy and recall$_{maj}$, but shows poor F-measure, G-mean, and recall$_{min}$ performances. This is because AWE is a traditional classifier and lacks a mechanism to adapt to class-imbalance conditions.
- MuSeRA and SERA adopt a similarity measure to select previous minority examples that are similar to those in the latest block. Compared to SERA, MuSeRA maintains all the knowledge of data streams by preserving ensemble members built over consecutive blocks. Thus, MuSeRA generally outperforms SERA on accuracy.
- Compared to traditional classifiers, SMOTE improves the minority class performance by generating novel minority

Please cite this article in press as: S. Ren, et al., Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.09.032.

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

	Friedman	AWE	SERA	MuSeRA	SMOTE	UB	OOB
F-measure	Reject	**4.750**	2.375	**4.000**	**3.500**	**3.875**	2.500
G-mean	Reject	**5.000**	2.750	**4.500**	2.125	2.750	**3.000**
AUC	Reject	1.375	**3.875**	**4.500**	**5.500**	2.625	1.375

A value greater than the CD (CD = 2.849) indicates statistically significant differences between the methods, which are highlighted in boldface.

events. Thus, SMOTE obtains good recall$_{min}$ and G-mean performances but performs poorly on accuracy, AUC, and recall$_{maj}$. While OOB maintains the best mean rank for accuracy (followed by SRE), it performs poorly on recall$_{min}$ and G-mean.

5.6.3. Statistical analysis of comparative results

In this section, we formally validate the effectiveness of SRE in terms of F-measure, G-mean, and AUC. The main goal of the statistical analysis is to check the following hypotheses:

- H_1: There are no significant differences among F-measure performances of comparative algorithms.
- H_2: There are no significant differences among G-mean performances of comparative algorithms.
- H_3: There are no significant differences among AUC performances of comparative algorithms.

The Friedman test [41], which is the most suitable tool to detect differences among the ranks of comparative algorithms across multiple datasets, is used to validate H_1, H_2, and H_3. Based on Tables 6, 7 shows the results of Friedman tests and the post-hoc analyses on F-measure, G-mean, and AUC. If the significant level is 0.05, then H_1, H_2, and H_3 are rejected. Thus, significant differences exist in the F-measure, G-mean, and AUC performances of tested algorithms. Then, the Bonferroni–Dunn post-hoc test [41] is used to compare SRE with other algorithms. The critical difference (CD) for $\alpha = 0.05$ is 2.849. Table 7 presents the differences between the mean ranks of two algorithms using SRE as the control algorithm. If the difference between the mean ranks of two comparative methods is greater than CD, the difference between these two algorithms is statistically significant. These significant differences are highlighted in boldface in the post-hoc table. In addition, the Wilcoxon signed-rank test [46] is further applied to compare SRE with the remaining algorithms in a pairwise manner. Table 8 shows the results of the Wilcoxon test in terms of F-measure, G-mean, and AUC using SRE as the control method. First, the p-values in terms of F-measure are: $p_{\text{SERA}} = 0.0039$ and $p_{\text{OOB}} = 0.0039$. Second, with regard to the G-mean performance, the p-values for SERA, SMOTE, and UB are $p_{\text{SERA}} = 0.0039$, $p_{\text{SMOTE}} = 0.0078$, and $p_{\text{UB}} = 0.0039$. Third, SRE statistically outperforms the remaining algorithms in terms of AUC ($p_{\text{AWE}} = 0.0313$, $p_{\text{UB}} = 0.0039$, and $p_{\text{OOB}} = 0.0039$).

Generally speaking, the F-measure, G-mean, and AUC performances of SRE are significantly better than those of other comparative algorithms. Meanwhile, SRE maintains good accuracy, recall$_{min}$, and recall$_{maj}$ performances on all datasets. While no algorithms can outperform others on all figures of merit, SRE obtains a good overall balance in accuracy, recall$_{min}$, recall$_{maj}$, G-mean, F-measure, and AUC in a wide spectrum of concept drift scenarios. The boost in the minority class performance for SRE does not significantly deteriorate the majority class performance.

5.6.4. Running time efficiency

In this section, we analyze the running time efficiency of the tested algorithms. The hardware configuration used for simulation is an Intel Core i7 Processor with 8 GB RAM. Table 9 records the running time of comparative methods on all the datasets. Each result is the average of 50 independent runs.

Table 8
Wilcoxon's test results for the comparison of SRE versus the remaining methods on all datasets.

F-measure		G-mean		AUC	
Methods	P-values	Methods	P-values	Methods	P-values
SRE vs. SERA	0.0039	SRE vs. SERA	0.0039	SRE vs. AWE	0.0313
SRE vs. OOB	0.0039	SRE vs. SMOTE	0.0078	SRE vs. UB	0.0039
		SRE vs. UB	0.0039	SRE vs. OOB	0.0039

Table 9
Runtime of the comparative algorithms (s)

	AWE	SERA	MuSeRA	SMOTE	UB	OOB	SRE
SEA$_S$	15.28	35.70	66.67	2.67	13.06	38.25	39.82
SEA$_G$	11.98	26.21	49.85	2.11	11.16	25.24	27.44
Hyper	128.94	806.80	990.71	17.71	150.59	153.75	900.08
RanRBF$_{GR}$	137.76	737.67	1072.41	14.92	113.88	136.78	400.65
SEA$_{SR}$	15.53	37.14	75.80	2.82	14.54	38.63	40.26
Domain$_1$	17.80	43.01	115.87	3.49	15.06	30.12	33.34
Domain$_2$	16.52	43.68	113.31	3.08	16.95	32.57	38.03
Elec	0.91	1.83	1.87	0.89	0.59	1.64	1.71

From the results presented in Table 9, we can draw the following conclusions. First, SMOTE periodically builds only one classifier to predict the labels of testing data. Thus, SMOTE generally consumes relatively small amount of time compared to ensemble-based models. Second, MuSeRA is the most time-consuming algorithm on all the datasets tested. To preserve all knowledge of data streams, MuSeRA does not leverage the ensemble pruning mechanism to remove obsolete classifiers. Thus, MuSeRA maintains all the component classifiers built over consecutive blocks, which would cost a considerable amount of time. Third, UB and SERA adopt the same strategy of building ensemble members. All the component classifiers of UB and SERA are trained over the latest chunk. Meanwhile, UB and SERA preserve the minority examples in the latest w blocks to afford fair comparison. However, SERA generally requires longer time than UB. This is because SERA needs to evaluate the similarity between each of the past preserved minority examples and the current minority set. Finally, the selectively resampling mechanism of SRE used to re-balance the candidate block considers both concept drift and data difficulty factors. Thus, the time consumption of SRE is generally higher than that of most of the comparative algorithms. We can observe that SRE does not consume too much time compared with the remaining algorithms on most of the datasets. However, the running time of SRE, SERA, and MuSeRA on the RanRBF$_{GR}$ and Hyper datasets is much longer than that of other approaches. This is mainly because the dimensions of these two datasets are relatively high and the resampling procedure of SRE, SERA, and MuSeRA needs a lot of distance calculations. Thus, we can conclude that SRE has a satisfactory time efficiency on low-dimensional datasets. Meanwhile, SRE typically provides a much better-balanced performance than other tested algorithms.

6. Conclusions

In this paper, we present a novel streaming classifier to learn concept drift from imbalanced data. The existing techniques to handle nonstationary imbalanced data streams have been reviewed. The main deficiencies of them mainly reflect in two aspects. On the one hand, they are generally designed to specialize in only one type of concept drift. SRE can quickly react to different kinds of concept drift by periodically updating previous ensemble members using the latest instances. The update weights are assigned to emphasize costly misclassification examples and minority examples in the training procedure of previous component classifiers. On the other hand, most of the existing methods have

Please cite this article in press as: S. Ren, et al., Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.09.032.

ignored the effect of complex data distribution on the classification task. The selection-based resampling procedure of SRE, which simultaneously considers concept drift and data difficulty factors, is used to extract past minority examples for re-balancing the current class distribution. The similarities between the preserved minority examples and the current minority set are first evaluated to avoid propagating drifting data into the training set of the candidate classifier. Then, selection weights of past minority objects are appropriately determined by borderline factor, disjunct factor, and deviation factor. The previous minority samples with high selection weights have the priority to be reused for resampling the current minority set.

SRE maintains a fixed-size ensemble framework to limit the time and memory usage. The final decision of a testing event is a weighted voting of ensemble members. The weights of past components are based on their performance. However, the candidate classifier is treated as the best-performing member and assigned the highest weight. This way, SRE quickly adapts to sudden drift and the cross-validation of the candidate classifier is avoided.

To validate the effectiveness of SRE, we compare it with 6 state-of-the-art data stream algorithms on accuracy, F-measure, G-mean, $recall_{min}$, $recall_{maj}$, and AUC performances. Two families of synthetic datasets, which contain different levels of data complexity and various types of concept drift, are designed. The obtained results show that SRE effectively improves the minority class performance without significantly deteriorating the majority class performance. The effects of chunk size and imbalance ratio on the performance of SRE are also discussed in the experimental studies.

Three issues of the SRE algorithm are left to be resolved in the future. First, we would like to extend our work to the multi-class cases. Second, the similarity evaluation among samples is transformed into the distance computation. Therefore, SRE can only handle datasets with continuous attributes. We plan to extend SRE to datasets with nominal attributes. Third, the time consumption of SRE on high-dimensional datasets is high because the resampling procedure needs a large number of distance calculations. In fact, these distance computations are independent and can be done in parallel. In the future, we plan to improve the time efficiency of SRE on high-dimensional datasets by executing the resampling procedure in parallel.

Acknowledgments

This study is supported by the National Natural Science Foundation of China (Grant Number: 61863010, 61873076, 61370171, 61572178, 61672214, 61672223 and 61772192).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.knosys.2018.09.032.

References

[1] J. Gama, Knowledge Discovery from Data Streams, CRC Press, 2010.
[2] I. Žliobaitė, M. Pechenizkiy, J. Gama, An overview of concept drift applications, in: Big Data Analysis: New Algorithms for a New Society, Springer, 2016, pp. 91–114.
[3] J. Sun, H. Fujita, P. Chen, H. Li, Dynamic financial distress prediction with concept drift based on time weighting combined with adaboost support vector machine ensemble, Knowl.-Based Syst. 120 (2017) 4–14.
[4] H. He, Y. Ma, Imbalanced Learning: Foundations, Algorithms, and Applications, John Wiley & Sons, 2013.
[5] T. Zhu, Y. Lin, Y. Liu, Synthetic minority oversampling technique for multiclass imbalance problems, Pattern Recognit. 72 (2017) 327–340.
[6] B. Krawczyk, L.L. Minku, J. Gama, J. Stefanowski, M. Woźniak, Ensemble learning for data stream analysis: a survey, Inf. Fusion 37 (2017) 132–156.
[7] S. Garcı, I. Triguero, C.J. Carmona, F. Herrera, et al., Evolutionary-based selection of generalized instances for imbalanced classification, Knowl.-Based Syst. 25 (1) (2012) 3–12.
[8] V. García, J.S. Sánchez, R.A. Mollineda, On the effectiveness of preprocessing methods when dealing with different levels of class imbalance, Knowl.-Based Syst. 25 (1) (2012) 13–21.
[9] S. Maldonado, J. López, Imbalanced data classification using second-order cone programming support vector machines, Pattern Recognit. 47 (5) (2014) 2070–2079.
[10] P. Tapkan, L. Özbakır, S. Kulluk, A. Baykasoğlu, A cost-sensitive classification algorithm: Bee-miner, Knowl.-Based Syst. 95 (2016) 99–113.
[11] J. Sun, J. Lang, H. Fujita, H. Li, Imbalanced enterprise credit evaluation with dte-sbd: Decision tree ensemble based on smote and bagging with differentiated sampling rates, Inform. Sci. 425 (2018) 76–91.
[12] L. Zhou, K.P. Tam, H. Fujita, Predicting the listing status of Chinese listed companies with multi-class classification models, Inform. Sci. 328 (2016) 222–236.
[13] L. Zhou, Q. Wang, H. Fujita, One versus one multi-class classification fusion using optimizing decision directed acyclic graph for predicting listing status of companies, Inf. Fusion 36 (2017) 80–89.
[14] J. Stefanowski, Dealing with data difficulty factors while learning from imbalanced data, in: Challenges in Computational Statistics and Data Mining, Springer, 2016, pp. 333–363.
[15] J.A. Sáez, B. Krawczyk, M. Woźniak, Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets, Pattern Recognit. 57 (2016) 164–178.
[16] S. Wang, L.L. Minku, X. Yao, Online class imbalance learning and its applications in fault detection, Int. J. Comput. Intell. Appl. 12 (04) (2013) 1340001.
[17] S. Ren, B. Liao, W. Zhu, K. Li, Knowledge-maximized ensemble algorithm for different types of concept drift, Inform. Sci. 430–431 (2018) 261–281.
[18] 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.
[19] J. Gao, W. Fan, J. Han, S.Y. Philip, A general framework for mining concept-drifting data streams with skewed distributions., in: SDM, SIAM, 2007, pp. 3–14.
[20] S. Chen, H. He, Sera: selectively recursive approach towards nonstationary imbalanced stream data mining, in: 2009 International Joint Conference on Neural Networks, pp. 522–529.
[21] S. Chen, H. He, K. Li, S. Desai, Musera: multiple selectively recursive approach towards imbalanced stream data mining, in: The 2010 International Joint Conference on Neural Networks (IJCNN), IEEE, 2010, pp. 1–8.
[22] G. Ditzler, R. Polikar, Incremental learning of concept drift from streaming imbalanced data, IEEE Trans. Knowl. Data Eng. 25 (10) (2013) 2283–2301.
[23] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Trans. Neural Netw. 22 (10) (2011) 1517–1531.
[24] 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.
[25] T.R. Hoens, N.V. Chawla, R. Polikar, Heuristic updatable weighted random subspaces for non-stationary environments, in: Data Mining (ICDM), 2011 IEEE 11th International Conference on, IEEE, 2011, pp. 241–250.
[26] T.R. Hoens, N.V. Chawla, Learning in non-stationary environments with class imbalance, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2012, pp. 168–176.
[27] K. Wu, A. Edwards, W. Fan, J. Gao, K. Zhang, Classifying imbalanced data streams via dynamic feature group weighting with importance sampling, in: SDM, SIAM, 2014, pp. 722–730.
[28] 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.
[29] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, Y. Zhou, A novel ensemble method for classifying imbalanced data, Pattern Recognit. 48 (5) (2015) 1623–1637.
[30] M. Tahir, J. Kittler, K. Mikolajczyk, F. Yan, A multiple expert approach to the class imbalance problem using inverse random under sampling, Multiple Classifier Syst. (2009) 82–91.
[31] P.C. Mahalanobis, On the generalized distance in statistics, Proc. Nat. Inst. Sci. (Calcutta) 2 (1936) 49–55.
[32] H. Han, W.Y. Wang, B.H. Mao, Borderline-smote: A new over-sampling method in imbalanced data sets learning, in: International Conference on Intelligent Computing, 2005, pp. 878–887.
[33] 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: Kdd, Vol. 96, 1996, pp. 226–231.
[34] W. Fan, S.J. Stolfo, J. Zhang, P.K. Chan, Adacost: misclassification cost-sensitive boosting, in: Icml, Vol. 99, 1999, pp. 97–105.
[35] 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.
[36] 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.

[37] A. Bifet, R. Kirkby, Massive online analysis, 2010, pp. 1601–1604.
[38] 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.
[39] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, Moa: Massive online analysis, J. Mach. Learn. Res. 11 (2010) 1601–1604.
[40] T. Fawcett, Roc graphs: Notes and practical considerations for data mining researchers, Mach. Learn. 31 (2003) 1–38.
[41] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[42] N. Japkowicz, S. Stephen, The Class Imbalance Problem: A Systematic Study, IOS Press, 2002, pp. 429–449.
[43] W.N. Street, Y. Kim, A streaming ensemble algorithm (SEA) for large-scale classification, in: Proceedings of the Seventh ACM SIGKDD Iternational Conference on Knowledge Discovery and Data Mining, ACM, 2001, pp. 377–382.
[44] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Brazilian Symposium on Artificial Intelligence, Springer, 2004, pp. 286–295.
[45] J. Ar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (1) (2006) 1–30.
[46] G.W. Corder, D.I. Foreman, Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach, 2009.

Please cite this article in press as: S. Ren, et al., Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.09.032.