International Journal of Computational Intelligence and Applications

ONLINE CLASS IMBALANCE LEARNING AND ITS APPLICATIONS IN FAULT DETECTION

SHUO WANG, LEANDRO L. MINKU, XIN YAO
CERCIA, School of Computer Science, University of Birmingham
Birmingham, B15 2TT, UK
E-mail: {S.Wang, L.L.Minku, X.Yao}@cs.bham.ac.uk

Although class imbalance learning and online learning have been extensively studied in the literature separately, online class imbalance learning that considers the challenges of both fields has not drawn much attention. It deals with data streams having very skewed class distributions, such as fault diagnosis of real-time control monitoring systems and intrusion detection in computer networks. To fill in this research gap and contribute to a wide range of real-world applications, this paper first formulates online class imbalance learning problems. Based on the problem formulation, a new online learning algorithm, Sampling-based Online Bagging (SOB), is proposed to tackle class imbalance adaptively. Then, we study how SOB and other state-of-the-art methods can benefit a class of fault detection data under various scenarios and analyse their performance in depth. Through extensive experiments, we find that SOB can balance the performance between classes very well across different data domains and produce stable G-mean when learning constantly imbalanced data streams, but it is sensitive to sudden changes in class imbalance, in which case SOB's predecessor Undersampling-based Online Bagging (UOB) is more robust.

Keywords: Online class imbalance; resampling; fault detection.

1. Introduction

Online learning has been extensively studied and applied in recent years. It refers to learning from data streams where data arrive continuously and a timely response is required. Strictly speaking, online learning algorithms process each training example once "on arrival" without the need for storage and reprocessing, and maintain a current model that reflects the current concept to make a prediction at each time step [35]. The online learner is not given any process statistics for the observation sequence, and thus no statistical assumptions can be made in advance [32]. Online learning has contributed to various real-world applications, such as sponsored search from web click data [11], credit card transactions [40] and spam filtering [34 12].

In many of such data stream applications, skewed class distributions exist and cause great learning difficulties, such as fault diagnosis of control monitoring systems and fault detection in software engineering. In these cases, some classes of data are much more difficult or expensive to be collected than the other classes, referred to as class imbalance. Given such imbalanced data streams, existing online learn-

ing algorithms can suffer severe performance degradation, since the large number of majority-class examples overwhelms the incremental update of the model and minority-class examples are likely to be ignored [20]. Unfortunately, most existing solutions for class imbalance are restricted to the offline mode.

Very little work has been devoted to learning from skewed data streams online. New challenges arise in online class imbalance learning, such as imbalance rate estimation [44], concept drift detection in imbalanced data streams [43] and online leaning techniques to overcome class imbalance [44] [33]. Although a few learning algorithms have been proposed for imbalanced data streams very recently, some essential questions are still open. Focusing on the fundamental concepts of online learning and class imbalance learning, this paper will answer the following research questions: what is the formal definition of online class imbalance learning? Based on the formal definition, how should we tackle class imbalance effectively during the online processing? How would online class imbalance algorithms contribute to real-world applications? What are their advantages and disadvantages?

For the first two questions, we formulate online class imbalance problems with the objective of maximizing G-mean, based on which we develop a learning algorithm called Sampling-based Online Bagging (SOB) to deal with class imbalance and make real-time predictions. SOB introduces sampling techniques into Online Bagging algorithm [35], in which the sampling rate is determined adaptively based on the current imbalance status and classification performance. With the theoretical underpinning, it is expected to outperform its earlier versions Oversampling-based Online Bagging (OOB) and Undersampling-based Online Bagging (UOB) [44]. For the last two questions, we examine the performance of SOB, OOB, UOB and two other recently proposed algorithms on a set of fault detection problems from real-world projects. Fault detection has drawn growing interest from both academia and industry. A good fault detection method can automate engineering procedures, reduce unnecessary costs and prevent fault events that may cause severe system failure. A typical fault detection task is inherently imbalanced between classes, which aims to distinguish faulty examples (minority) from non-faulty examples (majority) accurately. Since class imbalance methods have shown benefits to offline fault detection applications [46], this paper studies their role in online scenarios. Through comprehensive experimental discussions, we find that SOB can balance the performance between classes very well across different data domains and produce stable G-mean under static scenarios. However, it is sensitive to data sequences, especially when there is a sudden change in class imbalance. UOB, by contrast, is more robust against this situation. An in-depth analysis is given.

The rest of this paper is organized as follows. Section 2 gives the background knowledge about class imbalance learning and research progress in learning from imbalanced data streams, and describes fault detection problems. Section 3 gives the problem formulation of online class imbalance learning. Section 4 proposes the new online learning algorithm SOB. Section 5 discusses its performance in fault detection data sets in comparison with other existing algorithms. Section 6 draws

the conclusions and points out our future work.

2. Background

In this section, we first introduce class imbalance learning and the state-of-the-art methods in this area. Then, we briefly review the current research progress in learning from imbalanced data streams. Finally, we describe the classification problem of fault detection and explain how class imbalance learning can contribute to this type of tasks.

2.1. *Class imbalance learning*

Class imbalance learning refers to learning from data sets that exhibit significant imbalance among or within classes. The common understanding about "imbalance" in the literature is concerned with the situation, in which some classes of data are highly under-represented compared to other classes [20]. By convention, we call the classes having more examples the majority classes, and the ones having fewer examples the minority classes. The recognition of minority class is more important, because misclassifying an example from the minority class is usually more costly [46].

The challenge of learning from imbalanced data is that the relatively or absolutely underrepresented class cannot draw equal attention to the learning algorithm compared to the majority class, which often leads to very specific classification rules or missing rules for the minority class without much generalization ability for future prediction [48]. How to better recognize data from the minority class is a major research question in class imbalance learning. Its learning objective can be generally described as "obtaining a classifier that will provide high accuracy for the minority class without severely jeopardizing the accuracy of the majority class" [41].

Numerous methods have been proposed to tackle class imbalance problems offline at data and algorithm levels. Data-level methods include a variety of resampling techniques, manipulating training data to rectify the skewed class distributions, such as random over/under-sampling, SMOTE [6] and some dynamic sampling approaches [25]. Algorithm-level methods address class imbalance by modifying their training mechanism directly with the goal of better accuracy on the minority class, including one-class learning [22] and cost-sensitive learning algorithms [49 20]. In addition, ensemble learning [37] has become another major category of approaches to handling imbalanced data by combining multiple classifiers, such as SMOTEBoost [7], AdaBoost.NC [42 45] and the Multi-objective Genetic Programming ensemble [2]. Although class imbalance learning has been extensively studied, none of the proposed methods can process imbalanced data online.

In class imbalance learning, the traditional accuracy metric measuring the percentage of correctly classified examples is not appropriate for performance evaluation anymore, because it can be overwhelmed by the high accuracy on the majority class and hide the poor performance on the minority class. Instead, recall

and G-mean are more frequently used. Recall measures the accuracy of recognizing a specific class of data. Minority-class recall tells us how many minority-class examples can be correctly identified. However, it does not provide any performance information of other classes. To measure the overall accuracy, G-mean calculates the geometric mean of recalls over all classes [23]. More details will be given in Section 3.

2.2. *Learning from imbalanced data streams*

Most existing algorithms dealing with imbalanced data streams require processing data in batches (incremental learning). The first attempt was made by Gao et al. [17] [16]. They proposed an uncorrelated bagging strategy, based on the subset of majority-class data in the current data chunk and the union of all minority-class data collected thus far. Wang et al. proposed an ensemble algorithm based on clustering and sampling [47], in which K-means clustering algorithm is used to undersample the majority class in the current data chunk. Chen et al. proposed SERA [8] and its improved versions MuSeRA [10] and REA [9], which selectively add minority examples from previous chunks into current training chunk to balance data. Because these methods require access to previous data, they are more suitable to the problem where the minority data concept is stationary or historical data can be retained. Lichtenwalter and Chawla proposed a new metric to measure the distance between data chunks, which is used to weigh the ensemble members learnt from imbalanced data stream [24]. It is shown to be more suitable to extremely complicated data streams with complex concept drifts and high degrees of imbalance. Ditzler and Polikar proposed two ensemble approaches, Learn++.CDS and Learn++.NIE [13]. The former applies the well-established oversampling technique SMOTE [6] and the latter uses a Bagging based sub-ensemble method to rebalance data. They were shown to outperform earlier methods in general but at the cost of higher computational complexity due to the oversampling and sub-ensemble strategies.

All the aforementioned methods need to collect full data chunks for training. Therefore, they cannot be applied to online problems directly. Nguyen et al. proposed an online algorithm to deal with imbalanced data streams based on random undersampling [33]. The majority class examples have lower probability to be selected for training. It assumes that the information of which class belongs to the minority/majority is known and the imbalance rate does not change over time. Besides, it requires a training set to initialize the classification model before learning. Minku et al. [30] proposed to use undersampling and oversampling to deal with class imbalance in online learning by changing the parameter corresponding to Online Bagging's sampling rate. However, the sampling parameters need to be set prior to learning and cannot be adjusted to changing imbalance rates. Very recently, two perceptron-based methods RLSACP [18] and WOS-ELM [31] were proposed to tackle imbalanced data streams online. They adjust the weights between perceptrons depending on the class label. A higher weight is assigned to the minority class. They were tested in static scenarios with fixed imbalanced degree. In addition, WOS-ELM

requires a validation set to adjust class weights, which may not be available in many real-world applications.

2.3. *Fault detection problems*

Fault detection is an emerging and challenging practical task, aiming to find faulty data produced in any engineering systems in either real time or offline mode, such as sensor faults in smart buildings [28] and robotic systems [36], or identify programming faults in software engineering in a wider sense [5]. For the example of smart buildings with sensors installed to monitor hazardous events, any sensor fault can cause catastrophic failures. Detecting faults accurately and timely is crucial to the operation and stability of the overall system. Fault predictors for software engineering also become popular since 2005 after the PROMISE repository was created [3]. PROMISE includes a collection of fault prediction data sets from real-world applications for public use, among which the percentage of defective modules varies from 6.94% to 32.29%. An accurate software fault predictor can help to improve software quality and reduce the cost of delivering software systems.

Fault detection data are inherently imbalanced. In other words, a faulty example is much less likely to be observed than an example produced from the system in a normal condition. Obtaining a fault in such systems can be very expensive. Recent studies have employed cognition and machine learning methods to enhance fault diagnosis performance offline, but very few considered the highly imbalanced nature between faulty and non-faulty classes. The obtained classifier tends to perform poorly in predicting faulty examples due to the rarity of this class. Therefore, it is worthwhile exploring class imbalance methods to alleviate this learning difficulty. Although some researchers have noticed the class imbalance issue in fault detection and attempted to apply class imbalance techniques to improve the prediction accuracy [22] [46] [26], they are limited to offline data. Encouraged by their promising results, this paper explores how class imbalance learning can facilitate fault detection problems during online processing.

3. Problem Formulation of Online Class Imbalance Learning

Consider an online binary classification problem. We build an online classifier F that receives a new example x_t at each time step t and predicts its class label as $\hat{y}_t$. After making the prediction, the classifier receives the expected label y_t of x_t, evaluates its loss and gets updated. Both predicted label $\hat{y}_t$ and expected label y_t belong to label set $Y = \{+1, -1\}$. If $\hat{y}_t = y_t$, the prediction is correct. Without loss of generality, we assume the positive class to be the minority, which is much less likely to occur than the negative class, such as spam in email filter applications and faults in engineering systems. The imbalance rate (IR) is defined as the occurrence probability of the minority class. It is not easy to estimate IR in online scenarios, as we cannot obtain a whole picture of data and IR can change over time. The IR

estimation and performance evaluation are usually based on received examples so far.

Given a sequence of training examples with T time steps $(x_1, y_1), \ldots, (x_T, y_T)$, we can classify the prediction results from classifier F into four categories – true positives (TP) if $\hat{y}_t = y_t = +1$, true negatives (TN) if $\hat{y}_t = y_t = -1$, false positives (FP) if $\hat{y}_t = +1$ and $y_t = -1$, and false negatives (FN) if $\hat{y}_t = -1$ and $y_t = +1$. Based on these four metrics, recall and G-mean can be calculated, which have been frequently used as performance criteria in class imbalance learning. We use P to denote the number of positive examples and N to denote the number of negative examples. Minority-class recall (R_p), majority-class recall (R_n) and G-mean (G) are defined as:

$$R_p = \frac{TP}{TP + FN} = \frac{P - FN}{P}, R_n = \frac{TN}{TN + FP} = \frac{N - FP}{N};$$

$$G = \sqrt{R_p \cdot R_n} = \sqrt{\left(\frac{P - FN}{P}\right)\left(\frac{N - FP}{N}\right)}.$$

Minority-class recall is the ratio of the number of correctly classified minority-class examples to the number of all minority-class examples, reflecting the classification accuracy on the minority class. G-mean is the geometric mean of recall of both classes, an overall performance measure for imbalanced data. Due to different misclassification costs, recall and G-mean are more appropriate metrics than the traditional accuracy. Particularly, G-mean equally considers both classes with easy calculation.

With the goal of maximizing G-mean, we proof next that maximizing G-mean is equivalent to minimizing the objective:

$$\sum_{y_t=+1} \frac{N}{P} I\left(\hat{y}_t \neq y_t\right) + \sum_{y_t=-1} R_p I\left(\hat{y}_t \neq y_t\right), \tag{1}$$

where I is the indicator function. Then, we propose online learning algorithm – Sampling-based Online Bagging (SOB) to achieve this goal.

By expanding the G-mean expression, to maximize G-mean, we need to

$$\begin{aligned} &\text{minimize } \frac{FN}{P} + \frac{FP}{N} - \frac{FN \cdot FP}{P \cdot N} \Leftrightarrow \\ &\text{minimize } \frac{N}{P} FN + \frac{P - FN}{P} FP. \end{aligned}$$

Thus, we obtain Eq. (1). We can see from the equation that there is a performance trade-off between two classes to achieve the best G-mean. It suggests that we need to consider the current performance of the classifier when it is dealing with class imbalance online, in addition to the imbalance rate. Inspired by Eq. (1), we can

handle class imbalance by setting different misclassification costs – N/P (equivalent to $(1/IR - 1)$) for the minority class and R_p for the majority class. In this way, the minority class will receive a higher cost (≥ 1) than the majority class. The cost of majority class will depend on minority-class recall. If R_p is quite high, then the majority-class cost will not be lowered down that much. It is reasonable, because we only need to deemphasize the majority class when necessary.

4. Sampling-based Online Bagging

As one of the most popular methods applying different misclassification costs in the learning algorithm, resampling (aka rebalancing) has been theoretically and empirically proved to be an effective and simple technique [14] [38] for making the optimal decision, which manipulates the number of training examples without any extra requirements to the applied learning algorithm.

We propose sampling-based Online Bagging (SOB) that combines resampling with Online Bagging (OB) [35]. Online Bagging (OB) is a popular ensemble approach that successfully extends the well-known offline ensemble algorithm Bagging [4] to Online cases. It is based on the fact that when the number of training examples tends to infinite in offline Bagging, each training subset for base learner f_m contains K copies of each original training example, where the distribution of K tends to a $Poisson\,(\lambda)$ distribution with the parameter $\lambda = 1$. So, in Online Bagging, whenever a training example is available, it is presented K times for each ensemble member f_m, where K is drawn from $Poisson\,(1)$. It is designed for balanced data. To handle class imbalance, resampling is integrated by varying the parameter λ. If the current example belongs to the minority class, it will be oversampled through $Poisson\,(\lambda = N/P)$; otherwise, it will be undersampled through $Poisson\,(\lambda = R_p)$, according to Eq. (1). The training procedure of SOB at each time step is described in Table 1.

Table 1. Sampling-based Online Bagging.

Input: an ensemble classifier F composed of M base learners f_m, and current training example (x_t, y_t).

for each base learner f_m $(m = 1, 2, \ldots, M)$ **do**
  if y_t is $+1$
    set $K \sim Poisson\,(N/P)$
  else
    set $K \sim Poisson\,(R_p)$
  end if
  update f_m K times
end for

SOB is the modification of our recently proposed algorithms for online class imbalance problems – Oversampling-based Online Bagging (OOB) and

Undersampling-based Online Bagging (UOB) [44]. Different from the resampling strategy in SOB, OOB only oversamples minority-class examples at the rate of $\frac{P+N}{P}$, and UOB only undersamples majority-class examples at the rate of $\frac{P}{P+N}$. Their sampling rate only relies on the class size. For a more imbalanced data stream, the resampling degree will be encouraged more accordingly.

The next key issue is how to determine real-time IR and R_p in SOB. This work adopts the equations used in OOB and UOB [44]:

$$w_k^{(t)} = \eta w_k^{(t-1)} + (1-\eta)\,[(x_t, c_k)]\,(k = 1, 2, \ldots, |Y|), \tag{2}$$

$$R_k^{(t)} = \eta' R_k^{(t-1)} + (1-\eta')\,[x_t \leftarrow c_k]\,(k = 1, 2, \ldots, |Y|). \tag{3}$$

$w_k^{(t)}$ denotes the size percentage of class $c_k \in Y$ at time step t ($w_k^{(0)} = 0$). $[(x_t, c_k)] = 1$ if the true class label y_t of x_t is c_k, otherwise 0. Similarly, R_k stands for the current recall on class c_k. $[x_t \leftarrow c_k]$ is equal to 1 if x_t is correctly classified and 0 otherwise. At each time step, $w_k^{(t)}$ and $R_k^{(t)}$ are incrementally updated. Coefficients η and η' ($0 < \eta, \eta' < 1$) are time decay factors that balance the contribution between newly received examples and historical examples. Smaller time decay factors mean more focus on current data. The learning system will therefore have higher sensitivity to dynamic data streams, but less performance stability. If the class distribution of data streams is roughly known in advance, then it is better to use larger η and η' that can provide better estimation of $w_k^{(t)}$ and $R_k^{(t)}$. Our preliminary experiments suggest that the reasonable range for η and η' is $[0.9, 1)$ [44]. These equations are applicable to data streams with arbitrary number of classes. In binary cases, N/P is estimated by $w_{-1}^{(t)}/w_{+1}^{(t)}$, and $R_{+1}^{(t)}$ is used as R_p.

5. Applications in Fault Detection

Due to the rarity and importance of faults, fault detection in engineering systems is a typical problem of learning from imbalanced data. This section examines the performance of proposed algorithm SOB and explores the ability of existing online class imbalance methods on fault detection applications from real-world projects. The chosen data sets are highly imbalanced. We design and look into a series of practical scenarios, including not only data streams that are constantly imbalanced, but also data streams suffering short-term fluctuations of class imbalance status. SOB is compare with its earlier versions OOB and UOB and two most recent state-of-the-art methods RLSACP [18] and WOS-ELM [31].

5.1. *Data description*

Six real-world data sets are used in the experiments. Three of them, i.e. Gearbox, Smart building and iNemo, are collected in real time from complex engineering sys-

tems. The remaining three, i.e. JM1, KC1 and PC1, come from software engineering tasks, available in PROMISE [3].

Gearbox is the fault detection competition data from PHM society 2009 [1]. The task is to detect faults in a running gearbox using accelerometer data and information about bearing geometry. Data were sampled synchronously from accelerometers mounted on both the input and output shaft retaining plates. The original data contain more than one type of faults that can happen to gears and bearings inside the gearbox. To simplify the problem, we select one type of gear faults – the gear with a chipped tooth, which exists in the helical (spiral cut) gear with 24 teeth. The data set is thus a 2-class fault detection problem – the gear either in good condition (nonfaulty class) or having a chipped tooth (faulty class).

Smart building is a 2-class fault detection data set, aiming to identify sensor faults in smart buildings [28]. The sensors monitor the concentration of the contaminant of interest (such as CO2) in different zones in a building environment, in case any safety-critical events happen. In this data set, the sensor placed in the kitchen can be faulty. A wrong signal can lead to improper ventilation and unfavourable working conditions.

iNemo is a multi-sensing platform developed by STMicroelectronics for robotic applications, human machine interfaces and so on. It combines accelerometers, gyroscopes and magnetometers with pressure and temperature sensors to provide 3-axis sensing of linear, angular and magnetic motion in real time, complemented with temperature and barometer/altitude readings [39]. To avoid any functional disruption caused by signalling faults in iNemo, a fault emulator is developed for producing and analysing different types of faults. A fault is defined as an unpermitted deviation of at least one characteristic property or parameter of the system from the acceptable/usual/standard condition. It can be introduced into any sensor of iNemo by using the emulator given the real data sequence coming from the sensors. For this study, we generate offset faults for the feature of gyroscope x-axis, by adding an offset to its normal signal.

Examples in JM1, KC1 and PC1 describe attributes of software modules (i.e. the unit of functionality of source code) in programs, including the static code metrics defined by McCabe [27] and Halstead [19] and other complexity related attributes. McCabe metrics collect information about the complexity of pathways contained in the module through a flow graph. Halstead metrics estimate the reading complexity based on the number of operators and operands in the module. The learning objective is to find as many defective software modules as possible without hurting the overall performance of the constructed predictor (i.e. without increasing the false alarm rate).

The common features of the above data sets are: 1) the basic task is to discriminate between two classes – faulty and non-faulty; 2) the faulty examples present to be the minority in the long run, but may arrive in high frequency within a short period of time depending on the property of faults; 3) they aim to find faults accurately without degrading the performance on the other class; thus, high G-mean

(i.e. a good performance balance between classes) is desired for the obtained online predictor.

Based on the features of this type of problems, we produce highly imbalanced data streams for our experiment. For each of the three mechanical data sets, due to the large amount of collected data, we choose 5000 examples to form the data stream for training, in which only 50 examples are faulty. Besides, a data set containing 1000 examples is generated as a separate testing data. Both training and testing data have imbalance rate of 1% (i.e. the percentage of being faulty). For each of the three software engineering data sets, 90% examples are used for online training and the remaining 10% are used for testing. The original imbalance rate is kept for both. The data information is summarized in Table. 2.

Table 2. Fault Detection Data Information.

Data	Training No.	Testing No.	Attributes No.	IR
Gearbox	5000	1000	5	1%
Smart building	5000	1000	14	1%
iNemo	5000	1000	11	1%
JM1	9792	1088	21	19.35%
KC1	1898	211	21	15.45%
PC1	998	111	21	6.94%

We design four scenarios for each of the above data sets – one has a fixed imbalance rate, and the other three have class imbalance fluctuations.

- In the first scenario, the minority examples are uniformly distributed in the data stream, in which case the imbalance rate is constant. It simulates stable systems with hidden faults that are hard to be captured.
- In the second scenario, all the faults occur at the beginning of the training data. It is possible when the faulty system is fixed at some point. The data stream involves a sudden decrease of imbalance rate.
- In the third scenario, all the faults occur at the end of the training data. It can happen when a key part of the system is broken suddenly. The data stream involves a sudden increase of imbalance rate.
- In the last scenario, the faults become more and more frequent in the training data. It simulates the case when the damaged condition gets worse gradually. The data stream has a gradual increase of imbalance rate.

It is worth mentioning that the total number of faults in the training data stream is the same under all the four scenarios for each data set. Only the order of data examples is different.

5.2. *Experimental analysis*

How SOB performs on fault detection data in comparison with OOB, UOB, RLSACP [18] and WOS-ELM [31] is studied here. The SOB, OOB and UOB ensembles are composed of 50 Hoeffding trees [21]. To obtain accurate real-time class sizes and performance, the time decay factors η and η' for updating size percentage $w_k^{(t)}$ and single-class recall $R_k^{(t)}$ are set to 0.99. Based on our preliminary experiments [44], low values of η and η' can lead to unstable performance and lose the bigger picture of data. For fault detection data, since it is commonly known that the faulty class is the real minority and the imbalance rate is generally very small, we choose to set large values to balance learning adaptivity and stability.

RLSACP [18] and WOS-ELM [31] are perceptron-based approaches, adjusting weights between perceptrons and treating minority and majority classes differently to tackle class imbalance. The weights are determined by the number of examples observed in each class from the beginning or within a fix-sized window. Following the choice in the original papers, we set the forgetting factor to 0.9 and the learning rate to 0.1 in RLSACP. RLSACP has two error weighting strategies. The second strategy RLSACPII is applied here, which has fewer pre-defined parameters and was shown to produce similar results to the first error weighting strategy RLSACPI [18]. For WOS-ELM, the minority-class weight is set to the ratio of the number of majority-class examples to the number of minority-class examples observed so far, and the weight for the majority class is always equal to 1. The number of perceptrons in both RLSACP and WOS-ELM are equal to the number of attributes of the processing data stream. Their implementations are provided by the authors.

With the above settings, these algorithms are applied to the described data sets. For a clear understanding, the first scenario with a constant imbalance rate and the other three scenarios with class imbalance changes will be discussed separately.

5.2.1. *Data with constant imbalance rates*

To examine the online prediction ability, we perform prequential test, a popular performance evaluation strategy in online learning, in which each individual example is used to test the model before it is used for training, and from this the performance measures can be incrementally updated. Meanwhile, for generalization performance, we let the online model classify a separate testing data set after the online training finishes, as described in Table 2. All the discussed algorithms are run 100 times independently. Average G-mean is recorded for both prequential and after-training tests. Prequential G-mean curves along with the online processing are shown in Fig. 1. Each plot includes the curves from the five learning algorithms.

When the data stream is constantly imbalanced, we can see that SOB presents quite stable and high G-mean. It ends with the highest G-mean in Gearbox and PC1, and comparable G-mean to the highest in the remaining four cases. During the first few time steps of training, SOB may suffer from performance fluctuation to some extent, such as in KC1 and PC1. This is due to the less accurate minority-class

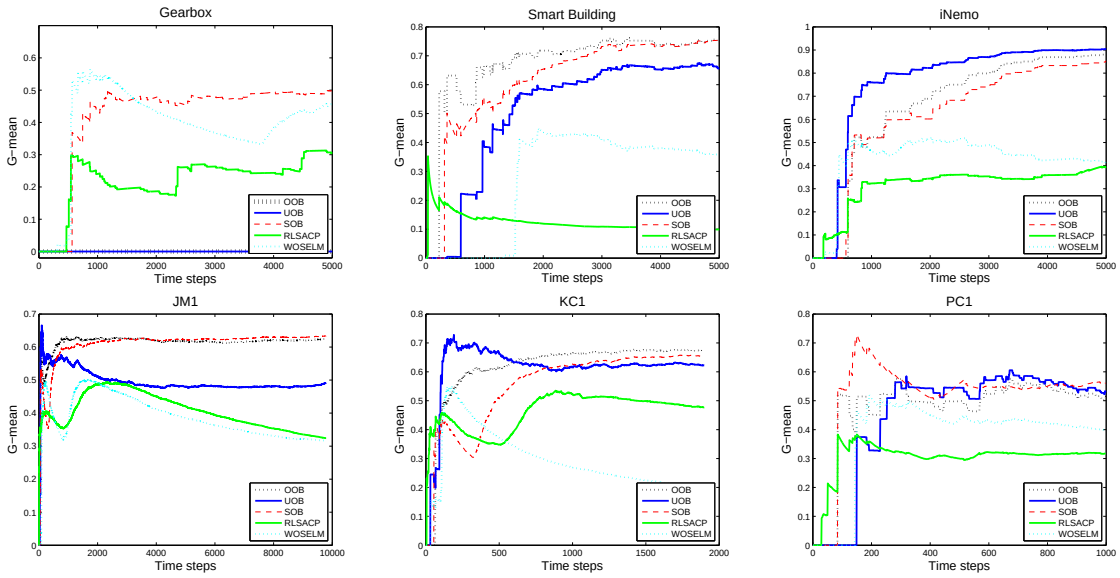

Fig. 1. Prequential G-mean curves of OOB, UOB, SOB, RLSACP and WOS-ELM on fault detection data.

recall and class size percentages used as misclassification costs. As more examples arrive, its overall performance gets better and more stable. Comparing OOB and UOB, OOB performs better in Smart Building, JM1, and KC1; UOB performs better in iNemo and PC1; both present zero G-mean in Gearbox, which is caused by zero minority-class recall. It implies that OOB and UOB' performance depends on the data domain more than SOB. Oversampling is more effective than undersampling in some cases, and the other way around in other cases. From this point of view, SOB has the advantages of both OOB and UOB, and is thus versatile in a broader range of applications. Besides, we notice that Gearbox is a more difficult classification problem than the others, in which neither oversampling in OOB nor undersampling in UOB helps to detect any faults. Compared to them, the combination of oversampling and undersampling in SOB is more aggressive.

With regards to RLSACP and WOS-ELM, their G-mean is not very satisfactory. Although they are quite aggressive in finding faults on Gearbox data, their G-mean is quite poor in all the other cases. Especially in JM1, KC1 and PC1, G-mean gets lower during learning. By observing prediction accuracy on each class, we find that faulty class is overemphasized, and RLSACP and WOS-ELM produce poor recall of non-faulty class. They are thus not good at balancing the performance trade-off between classes.

To further confirm our observations in the prequential test, we compare after-training G-mean through Wilcoxon Sign Rank tests with Holm-Bonferroni corrections at the overall level of significance of 0.05, between SOB and other methods.

Holm-Bonferroni corrections are performed to counteract the problem of multiple comparisons. 24 pairs of comparisons (6 data sets * 4 compared algorithms of SOB) are involved in this analysis. The average G-mean values and the corresponding statistical test results including p-values are shown in Table 3. If SOB achieves better performance than any of the other methods significantly, the corresponding p-value will be highlighted in bold italics.

Table 3. Means and standard deviations of G-mean from SOB, OOB, UOB, RLSACP and WOS-ELM on supplied testing data of fault detection data sets under the static scenario. P-values of the Wilcoxon Sign Rank tests between SOB and every other method are given in brackets. P-values in ***bold italics*** indicate statistically significant difference when using Holm-Bonferroni corrections at the overall level of significance of 0.05, considering the 24 comparisons performed.

	SOB	OOB	UOB	RLSACP	WOS-ELM
Gearbox	0.481±0.062	0.000±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***	0.450±0.121 (0.11140)
Smart building	0.953±0.032	0.977±0.000 ***(0.00000)***	0.797±0.008 ***(0.00000)***	0.000±0.000 ***(0.00000)***	0.428±0.228 ***(0.00000)***
iNemo	0.816±0.000	0.999±0.000 ***(0.00000)***	0.956±0.001 ***(0.00000)***	0.000±0.000 ***(0.00000)***	0.491±0.075 ***(0.00000)***
JM1	0.678±0.006	0.670±0.009 ***(0.00000)***	0.644±0.011 ***(0.00000)***	0.008±0.063 ***(0.00000)***	0.493±0.022 ***(0.00000)***
KC1	0.601±0.027	0.648±0.043 ***(0.00000)***	0.646±0.001 ***(0.00000)***	0.231±0.286 ***(0.00000)***	0.493±0.062 ***(0.00000)***
PC1	0.472±0.118	0.263±0.037 ***(0.00000)***	0.268±0.001 ***(0.00000)***	0.080±0.150 ***(0.00000)***	0.503±0.086 (0.04280)

The results in Table 3 generally tally with the prequential results. OOB's generalization ability seems to be better than UOB's in most cases; RLSACP performs the worst among all; WOS-ELM performs well in Gearbox and PC1, but quite poorly in the others. SOB is significantly better than both OOB and UOB in 3 out of 6 cases (Gearbox, JM1, PC1), worse than them in 2 cases (iNemo and KC1), and in between them in the remaining one (Smart building). In iNemo and KC1, the minority-class recall is not improved much by either oversampling or undersampling in SOB, probably because it overlearns the observed minority-class examples and lacks of generalization.

5.2.2. *Data with class imbalance changes*

When minority-class data do not come uniformly, how does the data sequence affect the performance of online class imbalance algorithms? Focusing on this research question, we adjust the order of faulty examples without changing the total number in the data stream. As described in Section 5.1, to simulate real-world situations,

we generate and discuss three different scenarios here: the scenario with a sudden IR decrease; the scenario with a sudden IR increase; the scenario with a gradual IR increase (corresponding to scenarios 2-4 in Section 5.1). Due to the poor performance of RLSACP and WOS-ELM, we only discuss OOB, UOB and SOB in this section. Table 4 presents their G-mean on the testing data to show their generalization performance under each scenario. The prequential performance is not included here, because the intensive arrival of faulty examples within a short period of time can give over-optimistic prequential results.

Table 4. Means and standard deviations of G-mean from SOB, OOB and UOB on supplied testing data of fault detection data sets under scenarios with class imbalance changes. P-values of the Wilcoxon Sign Rank tests between SOB and every other method are given in brackets. P-values in ***bold italics*** indicate statistically significant difference when using Holm-Bonferroni corrections at the overall level of significance of 0.05, considering the 12 comparisons performed for each scenario.

	Scenario 2 (sudden IR decrease)		
	SOB	OOB	UOB
Gearbox	0.000±0.000	0.000±0.000 (1.00000)	0.000±0.000 (1.00000)
Smart building	0.000±0.000	0.000±0.000 (1.00000)	0.831±0.016 ***(0.00000)***
iNemo	0.764±0.000	0.764±0.000 (1.00000)	0.969±0.001 ***(0.00000)***
JM1	0.000±0.000	0.000±0.000 (1.00000)	0.000±0.000 (1.00000)
KC1	0.647±0.001	0.646±0.000 ***(0.00000)***	0.648±0.001 ***(0.00000)***
PC1	0.268±0.000	0.268±0.001 (0.67140)	0.268±0.001 (0.01870)
	Scenario 3 (sudden IR increase)		
	SOB	OOB	UOB
Gearbox	0.001±0.004	0.466±0.087 ***(0.00000)***	0.589±0.03 ***(0.00000)***
Smart building	0.000±0.000	0.014±0.030 ***(0.00001)***	0.782±0.007 ***(0.00000)***
iNemo	0.133±0.200	0.925±0.019 ***(0.00000)***	0.963±0.003 ***(0.00000)***
JM1	0.000±0.000	0.000±0.000 (1.00000)	0.000±0.000 (1.00000)
KC1	0.095±0.088	0.055±0.031 ***(0.00000)***	0.648±0.001 ***(0.00000)***
PC1	0.265±0.026	0.267±0.001 ***(0.00530)***	0.269±0.001 ***(0.00000)***
	Scenario 4 (gradual IR increase)		
	SOB	OOB	UOB
Gearbox	0.470±0.109	0.000±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***
Smart building	0.817±0.080	0.974±0.000 ***(0.00000)***	0.935±0.000 ***(0.00000)***
iNemo	0.866±0.000	0.999±0.000 ***(0.00000)***	0.962±0.005 ***(0.00000)***
JM1	0.677±0.005	0.681±0.007 ***(0.00000)***	0.033±0.060 ***(0.00000)***
KC1	0.627±0.014	0.630±0.017 (0.16290)	0.646±0.001 ***(0.00000)***
PC1	0.487±0.135	0.242±0.081 ***(0.00000)***	0.268±0.001 ***(0.00000)***

For scenario 2, both SOB and OOB suffer a great G-mean reduction compared to their results in the static scenario. Zero G-mean is obtained in three out of six cases, which is due to the zero minority-class recall by looking into the accuracy on each class, but the majority-class recall remains one. It means that all faulty examples arrive constantly at the beginning of the data stream does not help to recognize faults better. Even if the recognition rate is high during that period, the

generalization is not improved. On the contrary, UOB's performance only gets worse in JM1, and gets even better in Smart building, iNemo and KC1. These observations suggest that the resampling strategy in UOB is more suitable to this scenario. SOB and OOB become less effective, because the oversampling level in OOB and SOB and the undersampling level in SOB are weakened. During the arrival of minority-class data, the online minority-class recall and size can get very high temporarily. Correspondingly, the oversampling and undersampling levels become lower than the ones in the static scenario. For the example of Smart building, the highest minority-class recall reaches 0.94 and the highest minority-class size reaches 0.35. So, the oversampling rate is 0.65/0.35 and the undersampling rate is 0.94 for SOB, which are a lot weaker than 0.99/0.1 and [0.25, 0.65] observed in the scenario 1. For UOB, even when the majority-class size is 0.65, the undersampling rate is still at the level of $(1 - 0.65)$, which is more aggressive than SOB. After more majority-class data come, its undersampling degree is further boosted, which does not happen to OOB and SOB.

For scenario 3, UOB still performs the best in most cases, and SOB performs the worst. Different from scenario 2, scenario 3 results in online models overemphasizing faulty examples. The very low G-mean in the table is caused by low majority-class recall, but the minority-class recall is nearly 1 in many cases. During the first stage of training, non-faulty data occupy the data stream. The observed imbalance rate is much lower than the real one. Thus, the undersampling level is more severe than what it should be. When faulty data start arriving intensively, the oversampling rate for OOB and SOB is extremely high (e.g. 1/0.001 for OOB and 0.999/0.001 for SOB) during the first few time steps, based on their sampling mechanism. Due to the double resampling procedures, SOB overlooks the majority class too much. Because OOB and UOB apply only one resampling strategy, their performance is affected less. From Section 5.2.1, we know that Gearbox is a difficult data set, in which most faulty examples tend to be ignored. In this scenario, both OOB and UOB present better generalization, because this data sequence enforces more aggressive sampling degree.

Compared to scenarios 2 and 3, scenario 4 does not affect the performance of the three methods that much, except that UOB still overemphasizes the minority class in JM1. A gradual IR change does not cause extreme performance and class size values, so it shouldn't cause as severe problems as the previous two cases.

According to the above analysis, we can see that an abrupt change in short-term imbalance rate can cause severe performance reduction, especially to SOB. When minority-class examples arrive frequently during the early stage of training, the online learner considers this class to be the majority. So, it tends to be overlooked. When minority-class examples arrive frequently during the later stage of training, the online learner exaggerates imbalance degree, and the minority class is overemphasized. One major reason is that the estimated IR and online performance are led by the local status of data and learner, which can be quite different from future status. Besides, SOB is more sensitive to the data sequence, because it uses both

oversampling and undersampling to overcome class imbalance. UOB is shown to be more robust against short-term changes. The underlying issue here is how to better recognize class imbalance status and discriminate any short-term and long-term changes, in order to reduce the impact of data sequences.

6. Conclusions and Future Work

This paper studied a new learning problem, online class imbalance learning, which considers the challenges of both online learning and class imbalance learning. To have a clear understanding of the underlying issue, we first formulated this type of problems with the aim of maximizing G-mean. It tells us that both imbalance degree and online performance need to be considered for the best G-mean. Second, based on the problem formulation, we proposed a new online algorithm SOB, making use of oversampling and undersampling to overcome class imbalance and integrating them into Online Bagging for online processing. With the theoretical underpinning, it is expected to outperform its two earlier versions OOB and UOB, and other state-of-the-art methods. To test its effectiveness, SOB was then used to solve a class of real-world applications – fault detection in engineering and software engineering systems, in comparison with OOB, UOB, RLSACP and WOS-ELM. These methods were discussed under four practical scenarios, including a constantly imbalanced scenario and three scenarios with class imbalance fluctuations. In the first scenario, we found that SOB can balance the performance between classes very well across different data domains and produce stable G-mean. However, it is sensitive to data sequences in the other scenarios, especially when there is a sudden change in class imbalance. UOB, by contrast, was shown to be more robust against this fluctuant situation.

Based on the results so far, there are several points we would like to look into in the near future. First, a weak point of SOB is its ability in dealing with dynamic data streams with concept drifts [29]. The reason could be that the proposed problem formulation aims to maximize G-mean greedily over all received examples. During the online processing, however, maximizing G-mean for examples received so far may not be the best way of maximizing G-mean later on, especially when the data stream suffers from some severe concept drifts. Therefore, it is necessary to formulate the problem that can have adaptive learning objectives to nonstationary situations. A possible solution is to not only consider past examples, but also find an objective that is robust to future environmental changes [15]. Besides, some techniques can be developed to detect changes in the data stream. Once it happens, the resampling strategy in SOB will be adapted to the change, in order to reduce the impact of concept drifts. Second, the problem formulation proposed in this paper is only applicable to two-class problems. How to formulate multi-class problems is important, as many applications involve more than one minority or majority class, and the number of classes can even change during the online processing. Third, we would like to apply our concept drift detection method DDM-OCI [43] to the discussed fault

detection tasks and examine whether it can help to further improve the prediction, considering some unknown concept drifts may happen to faulty examples over time.

Acknowledgements

The authors are grateful to Davide Ghezzi and Daniele Caltabiano for providing the iNemo data and to Michalis Michaelide for providing the smart building data used in our experiments. This work was supported by the European funded project (FP7 Grant No. 270428) "iSense: making sense of nonsense". Xin Yao was supported by a Royal Society Wolfson Research Merit Award.

References

1. 2009 PHM challenge competition data set. Online: http://www.phmsociety.org/references/datasets.
2. Urvesh Bhowan, Mark Johnston, Mengjie Zhang, and Xin Yao. Evolving diverse ensembles using genetic programming for classification with unbalanced data. *IEEE Transactions on Evolutionary Computation*, 17(3):368–386, 2013.
3. Gary Boetticher, Tim Menzies, and Thomas J. Ostrand. Promise repository of empirical software engineering data, 2007. Online: http://promisedata.org/repository.
4. Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
5. Cagatay Catal. Software fault prediction: A literature review and current trends. *Expert Systems with Applications*, 38(4):4626–4636, 2010.
6. Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:341–378, 2002.
7. Nitesh V. Chawla, Aleksandar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer. SMOTEBoost: Improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003*, volume 2838, pages 107–119.
8. Sheng Chen and Haibo He. Sera: Selectively recursive approach towards nonstationary imbalanced stream data mining. In *International Joint Conference on Neural Networks*, pages 522–529, 2009.
9. Sheng Chen and Haibo He. Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach. *Evolving Systems*, 2(1):35–50, 2010.
10. Sheng Chen, Haibo He, Kang Li, and Sachi Desai. Musera: Multiple selectively recursive approach towards imbalanced stream data mining. In *International Joint Conference on Neural Networks*, pages 1–8, 2010.
11. Massimiliano Ciaramita, Vanessa Murdock, and Vassilis Plachouras. Online learning from click data for sponsored search. In *International World Wide Web Conference*, pages 227–236, 2008.
12. Sarah Jane Delany, Padraig Cunningham, Alexey Tsymbal, and Lorcan Coyle. A case-based technique for tracking concept drift in spam filtering. *Knowledge-Based Systems*, 18(4-5):187–195, 2005.
13. Gregory Ditzler and Robi Polikar. Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 2012 (DOI: 10.1109/TKDE.2012.136).
14. Charles Elkan. The foundations of cost-sensitive learning. In *Proceedings of the Sev-*

enteenth International Joint Conference on Artificial Intelligence (IJCAI'01), pages 973–978, 2001.
15. Haobo Fu, Bernhard Sendhoff, Ke Tang, and Xin Yao. Finding robust solutions to dynamic optimization problems. In *Proceedings of the 16th European conference on Applications of Evolutionary Computation, Lecture Notes in Computer Science*, volume 7835, pages 616–625. Springer Berlin Heidelberg, 2013.
16. Jing Gao, Bolin Ding, Jiawei Han, Wei Fan, and Philip S. Yu. Classifying data streams with skewed class distributions and concept drifts. *IEEE Internet Computing*, 12(6):37–49, 2008.
17. Jing Gao, Wei Fan, Jiawei Han, and Philip S. Yu. A general framework for mining concept-drifting data streams with skewed distributions. In *Proceedings of SIAM ICDM*, pages 3–14, 2007.
18. Adel Ghazikhani, Reza Monsefi, and Hadi Sadoghi Yazdi. Recursive least square perceptron model for non-stationary and imbalanced data stream classification. *Evolving Systems*, 4(2):119–131, 2013.
19. Maurice H. Halstead. *Elements of Software Science*. Elsevier, 1977.
20. Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.
21. Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–106, 2001.
22. Nathalie Japkowicz, Catherine Myers, and Mark A. Gluck. A novelty detection approach to classification. In *Proceedings of the 14th international joint conference on Artificial intelligence*, pages 518–523, 1995.
23. Miroslav Kubat and Stan Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *Proc. 14th International Conference on Machine Learning*, pages 179–186, 1997.
24. Ryan N. Lichtenwalter and Nitesh V. Chawla. Adaptive methods for classification in arbitrarily imbalanced and drifting data streams. *New Frontiers in Applied Data Mining, Lecture Notes in Computer Science*, 5669:53–75, 2010.
25. Minlong Lin, Ke Tang, and Xin Yao. Dynamic sampling approach to training neural networks for multiclass imbalance classification. *IEEE Transactions on Neural Networks and Learning Systems*, 24(4):647 – 660, 2013.
26. Jordan McBain and Markus Timusk. Feature extraction for novelty detection as applied to fault detection in machinery. *Pattern Recognition Letters*, 32(7):1054–1061, 2011.
27. Thomas J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, 2(4):308–320, 1976.
28. Michalis P. Michaelides, Vasso Reppa, Christos Panayiotou, and Marios Polycarpou. Contaminant event monitoring in intelligent buildings using a multi-zone formulation. In *8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS)*, volume 8, pages 492–497, 2012.
29. Leandro L. Minku, Allan P. White, and Xin Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 22(5):730–742, 2010.
30. Leandro L. Minku and Xin Yao. DDD: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 24(4):619 – 633, 2012.
31. Bilal Mirza, Zhiping Lin, and Kar-Ann Toh. Weighted online sequential extreme learning machine for class imbalance learning. *Neural Processing Letters*, page (To Appear),

2013.
32. Claire E. Monteleoni. Online learning of non-stationary sequences. Technical report, Massachusetts Institute of Technology, 2003.
33. Hien M. Nguyen, Eric W. Cooper, and Katsuari Kamei. Online learning from imbalanced data streams. In *International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, pages 347–352, 2011.
34. Kyosuke Nishida, Shohei Shimada, Satoru Ishikawa, and Koichiro Yamauchi. Detecting sudden concept drift with knowledge of human behavior. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 3261–3267, 2008.
35. Nikunj C. Oza. Online bagging and boosting. *IEEE International Conference on Systems, Man and Cybernetics*, pages 2340–2345, 2005.
36. Vasso Reppa, Marios M. Polycarpou, and Christos G. Panayiotou. Adaptive approximation for multiple sensor fault detection and isolation of nonlinear uncertain systems. *IEEE Transactions on Neural Networks and Learning Systems*, 2013 (DOI: 10.1109/TNNLS.2013.2250301).
37. Lior Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2010.
38. Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Improving learner performance with data sampling and boosting. In *20th IEEE International Conference on Tools with Artificial Intelligence*, pages 452–459, 2008.
39. STMicroelectronics. iNemo: iNErtial MOdule V2 demonstration board. Online: http://www.st.com/internet/evalboard/product/250367.jsp.
40. Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235, 2003.
41. Shuo Wang. *Ensemble Diversity for Class Imbalance Learning*. PhD thesis, School of Computer Science, The University of Birmingham, 2011.
42. Shuo Wang, Huanhuan Chen, and Xin Yao. Negative correlation learning for classification ensembles. In *International Joint Conference on Neural Networks, WCCI*, pages 2893–2900. IEEE Press, 2010.
43. Shuo Wang, Leandro L. Minku, Davide Ghezzi, Daniele Caltabiano, Peter Tino, and Xin Yao. Concept drift detection for online class imbalance learning. In *International Joint Conference on Neural Networks (IJCNN '13)*, page (In Print), 2013.
44. Shuo Wang, Leandro L. Minku, and Xin Yao. A learning framework for online class imbalance learning. In *IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL)*, pages 36–45, 2013.
45. Shuo Wang and Xin Yao. The effectiveness of a new negative correlation learning algorithm for classification ensembles. In *IEEE International Conference on Data Mining Workshops*, pages 1013–1020, 2010.
46. Shuo Wang and Xin Yao. Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability*, 62(2):434–443, 2012.
47. Yi Wang, Yang Zhang, and Yong Wang. Mining data streams with skewed distribution by static classifier ensemble. *Opportunities and Challenges for Next-Generation Applied Intelligence, Studies in Computational Intelligence*, 214:65–71, 2009.
48. Gary M. Weiss. Mining with rarity: a unifying framework. *SIGKDD Explor. Newsl.*, 6(1):7–19, 2004.
49. Zhi-Hua Zhou and Xu-Ying Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):63– 77, 2006.