A Multi-Objective Ensemble Method for Online Class Imbalance Learning

Shuo Wang, Leandro L. Minku and Xin Yao

Abstract— **Online class imbalance learning is an emerging learning area that combines the challenges of both online learning and class imbalance learning. In addition to the learning difficulty from the imbalanced distribution, another major challenge is that the imbalanced rate in a data stream can be dynamically changing. OOB and UOB are two state-of-the-art methods for online class imbalance problems [1]. UOB is better at recognizing minority-class examples when the imbalance rate does not change much over time, while OOB is more prepared for the case with a dynamic rate. Aiming for an effective method for both static and dynamic cases, this paper proposes a multi-objective ensemble method MOSOB that combines OOB and UOB. MOSOB finds the Pareto-optimal weights for OOB and UOB at each time step, to maximize minority-class recall and majority-class recall simultaneously. Experiments on five real-world data applications show that MOSOB performs well in both static and dynamic data streams. Furthermore, we look into its performance on a group of highly imbalanced data streams. To respond to the minority class within 10000 time steps, the imbalance rate can be as low as 0.1% for easy data streams; at least 3% of imbalance rate is required to classify difficult data streams.**

I. Introduction

ONLINE class imbalance learning studies the combined issue of online learning and class imbalance learning, which has received growing attention in recent years. Specifically, online learning is concerned with the development of methods that process each data example one-by-one arriving in the form of data streams and maintain a model reflecting the current concept to make a prediction at each time step [2]. Class imbalance learning handles a type of classification problems where some classes of data (minority) are heavily underrepresented compared to other classes (majority), due to the fact that some data are very difficult or expensive to be collected [3]. Although each topic has been extensively discussed in the literature individually, new challenges arise when both exist. Two essential issues in online class imbalance learning are: how to overcome class imbalance online and how to adapt online models to dynamic changes. When dealing with class imbalance online, new problems include how to determine the current imbalance status without a general view of data and how to estimate the current imbalance degree. More difficult problems appear when the imbalance status is changing over time, such as how to detect the change and how to adapt the online model for good performance on the current minority class without damaging the performance on the current majority class.

Shuo Wang, Leandro L. Minku, and Xin Yao are with CERCIA, School of Computer Science, University of Birmingham, Birmingham, UK, B15 2TT (email: {S.Wang, L.L.Minku, X.Yao}@cs.bham.ac.uk).

In our recent work, we proposed an online definition of class imbalance through two indicators, and a class imbalance detection method to report the current class imbalance status in the data stream [1]. Based on the status information, we proposed two online learning methods – Oversampling-based Online Bagging (OOB) and Undersampling-based Online Bagging (UOB) [1]. UOB was shown to be very good at recognizing minority-class examples when the imbalance rate does not change much over time, and OOB is quite robust to cases with a dynamic imbalance rate. Imbalance rate (IR) is defined as the occurrence probability (percentage) of the minority class in data. It would be ideal if we can integrate the advantages of OOB and UOB into one model, to overcome both issues of poor classification performance and adaptation to dynamic environments.

It is commonly agreed that there is a performance trade-off between minority and majority classes of imbalanced data [4] [5]. Solely improving the recognition accuracy on one class often harms the accuracy on the other. Therefore, a multi-objective method can be a good option here, to find the optimal performance balance between minority and majority classes. This idea has been applied to offline class imbalance learning successfully [6] [7].

In order to overcome present limitations and obtain an effective and robust online model, this paper proposes a multi-objective approach – Multi-Objective Sampling-based Online Bagging (MOSOB), which builds a hybrid online model combining OOB and UOB. It aims to maximize accuracies on minority and majority classes simultaneously and tackle both static and dynamic data streams. The accuracy on each class is termed "recall" in the rest of the paper for consistency. At each time step, MOSOB finds the Pareto-optimal weights assigned to OOB and UOB, by searching in a finite set of solutions. For a swift response to the new class imbalance status in the dynamic data stream, time-decayed recall [1] is used to evaluate the fitness. The performance of MOSOB is discussed through extensive experiments in comparison with four other methods – OOB, UOB, RLSACP [8] and WOS-ELM [9]. MOSOB is shown to successfully combine the advantages of OOB and UOB, and suitable for both static and dynamic data streams.

Finally, for a thorough understanding of the predictive ability of MOSOB, we look into its performance on highly imbalanced data streams with a fixed IR ranging between 0.1% and 3%. To respond to the minority class within 10000 time steps, MOSOB needs at least 3% IR for difficult data; IR can be as low as 0.1% for easy data. Otherwise, auxiliary data knowledge and methods must be used to help the learning.

The *classification difficulty* here is categorized according to the performance of MOSOB on the data sets we discussed in the experiments.

II. Learning from Imbalanced Data Streams

In this section, we review the research progress in learning from imbalanced data streams, define class imbalance under online scenarios, and introduce the two classification methods (i.e. OOB and UOB) to learn from imbalanced data streams. They form the basis of this paper.

A. Research Progress

Most existing algorithms dealing with imbalanced data streams require processing data in batches (incremental learning), such as MuSeRA [10] and REA [11] proposed by Chen et al., and Learn++.CDS and Learn++.NIE [12] proposed by Ditzler and Polikar. Among limited class imbalance solutions strictly for online processing, Nguyen et al. first proposed an algorithm dealing with imbalanced data streams through random undersampling [13]. It assumes that the information of which class belongs to the minority/majority is known and the imbalance rate does not change over time. Minku et al. [14] proposed to use undersampling and oversampling to deal with class imbalance in online learning by changing the parameter corresponding to Online Bagging's [2] 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 [8] and WOS-ELM [9] were proposed, which assign different misclassification costs to classes to adjust the weights between perceptrons. The error committed on the minority class suffers a higher cost. RLSACP adopts a window-based strategy to update misclassification costs based on the number of examples in each class at a predefined speed. WOS-ELM requires a validation set to adjust misclassification costs based on classification performance, which however may not be available in many real-world applications. They were tested in static scenarios with a fixed imbalance rate and shown to be effective.

B. Defining Class Imbalance Online

To handle class imbalance online, we first need to define it by answering the following three questions: 1) is the data stream currently imbalanced? 2) Which classes belong to the minority/majority? 3) What is the imbalance rate currently? We answered the questions by defining two online indicators – time-decayed class size and recall calculated for each class [1]. Different from the traditional way of considering all observed examples so far equally, they are updated incrementally by using a time decay (forgetting) factor to emphasize the current status of data and weaken the effect of old data.

Suppose a sequence of examples (x_t, y_t) arriving one at a time. x_t is a p-dimensional vector belonging to an input space X observed at time t, and y_t is the corresponding label belonging to the label set $Y = \{c_1, \ldots, c_N\}$. For any class $c_k \in Y$, the class size indicates the occurrence probability (percentage) of examples belonging to c_k. If the label set contains only two classes, then the size of the minority class is referred to as the imbalance rate (IR) of the data stream, reflecting how imbalanced the data stream is at the current moment. The recall of class c_k indicates the classification accuracy on this class. To reflect the current characteristics of the data stream, these two indicators are incrementally updated at each time step. When a new example x_t arrives, the size of each class, denoted by $w_k^{(t)}$, is updated by [1]:

$$w_k^{(t)} = \theta w_k^{(t-1)} + (1-\theta)\left[(x_t, c_k)\right], (k = 1, \ldots, N) \quad (1)$$

where $[(x_t, c_k)] = 1$ if the real class label y_t of x_t is c_k, otherwise 0. θ $(0 < \theta < 1)$ is a pre-defined time decay factor, which forces older data to affect the class percentage less along with time through the exponential smoothing. Thus, $w_k^{(t)}$ is adjusted more based on new data.

If x_t's real label y_t is c_k, the recall of class c_k, denoted by $R_k^{(t)}$, is updated by [1]:

$$R_k^{(t)} = \theta' R_k^{(t-1)} + (1-\theta')\left[x_t \leftarrow y_t\right]. \quad (2)$$

For the recall of all the other classes ($c_k \neq y_t$), $R_k^{(t)}$ is updated by:

$$R_k^{(t)} = R_k^{(t-1)}, (k = 1, \ldots, N, c_k \neq y_t). \quad (3)$$

In Eq.2, θ' $(0 < \theta' < 1)$ is the time decay factor for emphasizing the learner's performance at the current moment. $[x_t \leftarrow y_t]$ is equal to 1 if x is correctly classified, and 0 otherwise. $\theta = 0.9$ and $\theta' = 0.9$ were shown to be a reasonable setting to balance the responding speed and the estimation variance in our experiments [1].

$w^{(t)}$ and $R^{(t)}$ values are then used by a class imbalance detection method, which outputs the information of whether the data stream should be regarded as imbalanced and which classes should be treated as the minority class. It examines two conditions for any two classes c_i and c_j:

- $w_i - w_j > \delta_1$ $(0 < \delta_1 < 1)$
- $R_i - R_j > \delta_2$ $(0 < \delta_2 < 1)$

If both conditions are satisfied, then the small class is sent to the minority class label set Y_{min} and the large class is sent to the majority class label set Y_{maj}. If Y_{min} and Y_{maj} are not empty, it means that the data stream is imbalanced, which will then invoke the class imbalance techniques running in the online model to tackle the imbalanced distribution.

C. OOB and UOB Methods

With the information from the class imbalance detection method, we proposed OOB and UOB to learn imbalanced data streams [1]. They integrate resampling into Online Bagging (OB) [2]. Once Y_{min} and Y_{maj} are not empty, oversampling or undersampling embedded in Online Bagging will be triggered to either increase the chance of training minority class examples (in OOB) or reduce the chance of training majority class examples (in UOB), through the

TABLE I: OOB and UOB Training Procedures.

Input: label sets Y_{min} and Y_{maj}, an ensemble with M base learners, and current training example (x_t, y_t).

for each base learner f_m $(m = 1, 2, \ldots, M)$ **do**
  if $y_t \in Y_{min}$
    set $K \sim Poisson\,(\lambda_1)$
  else
    set $K \sim Poisson\,(\lambda_2)$
  end if
  update f_m K times
end for

parameter of Poisson distribution. Their training procedures are given in Table I.

In OOB, λ_1 is set to $1/w_k^{(t)}$; λ_2 is set to 1. In UOB, λ_1 is set to 1, λ_2 is set to $\left(1 - w_k^{(t)}\right)$. Both OOB and UOB showed improved classification performance. Particularly, UOB performs better on the minority class. However, it suffers from performance reduction when there is a sudden and severe change in the class imbalance status, due to the information loss from undersampling during the old status. OOB is more robust against this type of changes. It is worth clarifying that the dynamic change in class imbalance discussed in this paper is referred to as the change in class priors usually caused by the nonstationary data source. It can be viewed as a form of concept drift [15].

III. A Multi-Objective Ensemble Method

With the aim of combining the strength of OOB and UOB and achieving the best performance trade-off between minority and majority classes, this section proposes the multi-objective sample-based Online Bagging method (MOSOB), maximizing the minority- and majority-class recall as competing objectives in the learning process. It builds a hybrid online model, which trains and keeps both OOB and UOB. Combining multiple models have been shown to be beneficial to the generalization in class imbalance learning [16]. For the final prediction, MOSOB maintains a set of model weights, optimized at each time step. Due to the requirement of real-time processing and the nonstationary feature, a simple and efficient optimization strategy is adopted here. The pseudo-code of MOSOB at each training step t is given in Table II, assuming two possible classes $Y = \{+1, -1\}$ (denoted by '+' and '-' for simplicity).

Before the learning starts, an OOB model and an UOB model are initialized, with a random weight assigned to each. The weight of UOB is denoted by w_{uob}; the weight of OOB is always $(1 - w_{uob})$. At each time step, we first obtain the weighted vote $\hat{y}$ from OOB (with output $\hat{y}_{oob}$) and UOB (with output $\hat{y}_{uob}$) as the prediction of MOSOB on the current example x_t (lines 1-3). After we receive its expected label y_t, time-decayed class size $w^{(t)}$ and recall of OOB ($R_{(o)}^{(t)}$) and UOB ($R_{(u)}^{(t)}$) are updated based on the values from the previous time step (lines 4-6), in which $w^{(t)} = \left(w_+^{(t)}, w_-^{(t)}\right)$,

TABLE II: Multi-Objective Sample-based Online Bagging (MOSOB).

Input: label sets Y_{min} and Y_{maj}, two ensembles OOB and UOB with M base learners each, finite set of possible model weights W, UOB weight $w_{uob} \in W$, OOB recall $R_{(o)}^{(t-1)}$, UOB recall $R_{(u)}^{(t-1)}$, MOSOB recall $R^{(t-1)}$, class sizes $w^{(t-1)}$, and current example (x_t, y_t).

// make the prediction and update status
1. $\hat{y}_{oob} = OOB\,(x_t)$
2. $\hat{y}_{uob} = UOB\,(x_t)$
3. $\hat{y} = \arg\max_{y \in Y} \left((1 - w_{uob})\,\hat{y}_{oob} + w_{uob}\hat{y}_{uob}\right)$
4. Update and obtain class size $w^{(t)}$ using Eq.1.
5. Update and obtain OOB recall $R_{(o)}^{(t)} = \left(R_{+(o)}^{(t)}, R_{-(o)}^{(t)}\right)$ using Eq.2 and Eq.3.
6. Update and obtain UOB recall $R_{(u)}^{(t)} = \left(R_{+(u)}^{(t)}, R_{-(u)}^{(t)}\right)$ using Eq.2 and Eq.3.

// optimize w_{uob}
7. **for** all $w_{uob} \in W$ do
8. $\quad \hat{y}_{cdd} = \arg\max_{\hat{y} \in Y} \left((1 - w_{uob})\,\hat{y}_{oob} + w_{uob}\hat{y}_{uob}\right)$
9. $\quad$ Get a candidate $R = (R_+, R_-)$ using Eq.2 and Eq.3 based on $\hat{y}_{cdd}$ and $R^{(t-1)}$.
10. $\quad$ Save current R.
11. **end for**
// set the Pareto-optimum to w_{uob} based on candidate R_k's
12. Let $\{w_{opt}\} = \arg\min_{w_{uob} \in W} \left(\|(R_+, R_-)\,, (1, 1)\|\right)$.
13. **if** $|\{w_{opt}\}| = 1$
14. $\quad w_{uob} = w_{opt}$
15. **else**
16. $\quad$ **if** $\left\|\left(R_{+(o)}^{(t)}, R_{-(o)}^{(t)}\right), (1, 1)\right\| > \left\|\left(R_{+(u)}^{(t)}, R_{-(u)}^{(t)}\right), (1, 1)\right\|$
17. $\quad\quad w_{uob} = 1$
18. $\quad$ **else**
19. $\quad\quad w_{uob} = 0$
20. $\quad$ **end if**
21. **end if**
22. Update and obtain $R^{(t)} = \left(R_+^{(t)}, R_-^{(t)}\right)$ using Eq.2 and Eq.3 based on $\hat{y}$ and $R^{(t-1)}$.

// train online models and update class imbalance status
23. Train OOB using (x_t, y_t).
24. Train UOB using (x_t, y_t).
25. Update Y_{min} and Y_{maj} based on $w^{(t)}$ and $R^{(t)}$.

$R_{(o)}^{(t)} = \left(R_{+(o)}^{(t)}, R_{-(o)}^{(t)}\right)$ and $R_{(u)}^{(t)} = \left(R_{+(u)}^{(t)}, R_{-(u)}^{(t)}\right)$.

Next, we optimize UOB's weight w_{uob} based on the respective output of OOB and UOB ($\hat{y}_{oob}$ and $\hat{y}_{uob}$). MOSOB searches in a finite set of candidate weights W in the range of $[0, 1]$. By setting w_{uob} to each element in W at each iteration, we get a weighted vote as one candidate output of MOSOB (denoted by $\hat{y}_{cdd}$, line 8). Based on $\hat{y}_{cdd}$ and recall of the previous time step $R^{(t-1)}$, we obtain a temporary time-decayed recall pair $R = (R_+, R_-)$ using Eq.2 and Eq.3 (lines 9-10). Each w_{uob} in W results in a pair of recall values. After going through all candidate w_{uob} values, the optimal weight for UOB is set to the one having (R_+, R_-) closest to the ideal point (1, 1), denoted by w_{opt} (line 14). This choice is a Pareto optimum of this multi-objective maximization problem [7] according to Kuhn-Tucker Theorem [17], based on the current status of data stream. This optimal weight will be used for classifying the next data point. It is possible that

different w_{uob} values lead to the same distance between the recall pair of MOSOB and point (1, 1), i.e. the cardinality of set $\{w_{opt}\}$ is larger than 1. When it happens, we simply assign weight value 1 to the model, whose *own* recall values ($R^{(t)}_{(o)}$ or $R^{(t)}_{(u)}$) are closer to (1, 1), and assign 0 to the other (lines 16-20). After the optimization steps are completed, $R^{(t)}$ of MOSOB is updated based on its original output $\hat{y}$ and $R^{(t-1)}$ (line 22).

Finally, example (x_t, y_t) is sent to OOB and UOB for training, following the procedures in Table I (lines 23-24). The class imbalance status of data stream is also updated according to $w^{(t)}$ and $R^{(t)}$ (line 25). The use of time-decayed recall and class size guarantees a timely reflection of online status of the data stream. The optimization procedure returns a Pareto-optimal solution for model weights that best balance the online performance between classes.

IV. Performance Analysis

This section examines the performance of MOSOB, in comparison with OOB, UOB, and two recently proposed algorithms RLSACP [8] and WOS-ELM [9]. The experiments here aim to answer the following two questions: first, is MOSOB effective on both types of data streams with static and dynamic imbalance rates? Second, is MOSOB effective on highly imbalanced data streams?

A. Effectiveness on Data Streams with Static and Dynamic Imbalance Rates

We consider two online imbalanced scenarios – data streams with a fixed IR (static) and data streams with a changing IR (dynamic). Dynamic ones are more challenging, because they require that the online model can sense the change and then adjust its learning bias quickly to maintain its performance. We generate both types for this experiment, using data from five real-world applications. Therefore, ten data streams are produced in total. Detailed data information and experimental settings will be given next.

1) Data and experimental design: The five real-world applications are: Gearbox fault detection data [18], Smart Building fault detection data [19], PAKDD 2009 credit card data [20], KDD Cup 1999 network intrusion detection [21] and iNemo robotic platform data with faults [22]. They are inherently imbalanced data obtained online and have been used in related studies [14] [23].

Gearbox aims to detect faults in a running gearbox. The original data contain multiple types of faults. To simplify the problem, we pick one type of faults that happens to the helical gear with 24 teeth. Smart Building is a two-class fault detection data set, aiming to identify sensor faults in smart buildings. In this data set, the sensor placed in the kitchen can be faulty. iNemo is a multi-sensing platform developed for robotic systems and HCI applications. To avoid any functional disruption caused by signalling faults in iNemo, a fault emulator is developed for producing and analysing different types of faults. In our study, we introduce the offset type of faults into x-axis of sensor gyroscope.

The task of the above three data applications is to build and maintain an effective online learner to detect faults. Due to the rarity and importance of faults, fault detection in engineering systems is a typical problem of learning from imbalanced data streams. There are two classes in the data stream – nonfaulty and faulty. The faulty class is usually the minority as it is much less likely to happen than the nonfaulty class. Nevertheless, it could also happen that those faults become very frequent suddenly, when the damaged condition gets worse, or the faults are not likely to happen any more, when the faulty system is repaired.

Data in PAKDD 2009 are collected from the private label credit card operation of a Brazilian retail chain. The task of this problem is to identify whether the client has a good or bad credit. The "bad" credit is the minority class, taking round 19.75% of the data. Because data has been collected from a time interval in the past, gradual market change occurs and may reduce the performance of the online model.

The task of KDD Cup 1999 data set is to distinguish between attacks and normal connections in computer networks. This data contain a total of 24 attack types. We only use the attack class "back" and class "normal" in this experiment, in which around 2.21% of data belong to the attack class.

For each of the data applications, we generate a data stream with static IR and a data stream with dynamic IR. For a clear observation, we choose 1000 examples forming the static one and 5000 examples forming the dynamic one without changing the original time order within each class. For the static data stream, the minority class is fixed, which is the smaller class in the original data set. The dynamic data stream contains 5 static periods of 1000 time steps. Each class takes turns to be the minority class after every 1000 examples has arrived. Although some changes may not seem to be practical, more challenging scenarios can better test our methods. For the three fault detection data, IR is always 5% in both static and dynamic scenarios. The original IR is kept for the other two data sets.

MOSOB, OOB, UOB, RLSACP [8] and WOS-ELM [9] are compared. The latter two are perceptron-based approaches. We include them in our comparisons because they are recent online class imbalance learning algorithms. There are few other online class imbalance learning algorithms in the literature. Following the choice in the original papers, we set the number of neurons to the number of features of data, with the sigmoidal activation function. RLSACP includes two error weighting strategies to tackle class imbalance. The second strategy RLSACPII is adopted here, which has fewer predefined parameters and was shown to produce similar results to the first error weighting strategy RLSACPI. RLSACPII penalizes errors differently between classes, depending on the imbalance ratio. This ratio is updated at every 100 time steps. WOS-ELM requires a validation data set for adjusting class weights to overcome class imbalance. Considering that the validation set is not always available and it may expire over time, we modify its weight-updating strategy and use time-decayed size percentage instead. Specifically,

TABLE III: Means and standard deviations of **minority-class recall** on the last time step for static data streams.

Method		Gearbox	Smart Building	PAKDD	iNemo	KDD
Tree ensemble	MOSOB	0.092±0.018	0.439±0.022	0.281±0.022	0.887±0.023	0.635±0.012
	OOB	0.000±0.000	0.097±0.026	0.331±0.018	0.888±0.010	0.631±0.000
	UOB	0.190±0.025	0.525±0.026	0.187±0.022	0.896±0.016	0.684±0.000
Perceptron ensemble	MOSOB	0.007±0.010	0.000±0.000	0.034±0.053	0.004±0.011	0.000±0.000
	OOB	0.000±0.000	0.000±0.000	0.001±0.001	0.001±0.003	0.000±0.000
	UOB	0.028±0.020	0.001±0.006	0.051±0.084	0.002±0.007	0.000±0.000
RLSACP		0.299±0.085	0.949±0.122	0.000±0.000	0.558±0.292	0.000±0.000
WOS-ELM		0.003±0.012	0.023±0.000	0.063±0.009	0.000±0.000	0.000±0.000

TABLE IV: Means and standard deviations of **G-mean** on the last time step for static data streams.

Method		Gearbox	Smart Building	PAKDD	iNemo	KDD
Tree ensemble	MOSOB	0.287±0.029	0.640±0.017	0.477±0.016	0.940±0.012	0.800±0.007
	OOB	0.000±0.000	0.306±0.042	0.509±0.011	0.940±0.005	0.795±0.000
	UOB	0.397±0.027	0.683±0.020	0.399±0.022	0.946±0.008	0.827±0.000
Perceptron ensemble	MOSOB	0.046±0.068	0.000±0.000	0.139±0.105	0.022±0.060	0.000±0.000
	OOB	0.000±0.000	0.000±0.000	0.010±0.025	0.004±0.025	0.000±0.000
	UOB	0.138±0.087	0.002±0.024	0.178±0.102	0.014±0.044	0.000±0.000
RLSACP		0.450±0.029	0.072±0.095	0.000±0.000	0.334±0.219	0.000±0.000
WOS-ELM		0.012±0.051	0.150±0.000	0.243±0.017	0.000±0.000	0.000±0.000

the error cost for the majority class is fixed to 1, while that for the minority class is set to $w_{maj}^{(t)}/w_{min}^{(t)}$, where $w_{maj}^{(t)}$ ($w_{min}^{(t)}$) denotes the time-decayed recall of the majority (minority) class calculated by Eq.1. There are two reasons for making this modification: 1) avoid using validation sets; 2) our preliminary results show that using fixed class weights without the validation set in WOS-ELM causes very poor performance in dynamic data streams.

MOSOB, OOB and UOB are ensemble methods composed of multiple base classifiers. Hoeffding tree [24] and single perceptron classifier are chosen to be the base classifier. Hoeffding tree is a fast decision tree induction algorithm that is capable of learning from massive data streams. It is shown to be effective in OOB and UOB [15] [23]. Single perceptron classifier is included here for a fair comparison with RLSACP and WOS-ELM. OOB and UOB consist of 50 base classifiers. MOSOB includes one OOB and one UOB. The candidate set for optimizing model weights is a set of values in range [0, 1] with interval 0.1. The implementation of base classifiers was provided by the Massive Online Analysis (MOA) tool with its default settings [25]. All the five methods are repeated for 100 times on every data stream.

Our performance comparison and evaluation are based on prequential test [26], 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. The model is always being tested on examples it has not seen, and thus reflects its prediction performance so far. Because class imbalance status changes after every 1000 examples arrive in dynamic data streams, we reset performance metrics to 0 right after the change occurs. This ensures that the performance observed after the change is not affected by the performance before the change, allowing us to analyse the behaviour of the models before and after the change adequately. G-mean and minority-class recall are recorded at each time step for performance analysis, which are the two most commonly used evaluation criteria in the class imbalance learning literature. Recall reflects classification accuracy on a single class. G-mean measures the overall accuracy by calculating the geometric mean of recalls over all classes [27]. They are better metrics than the traditional accuracy, because the latter can be overwhelmed by the high accuracy on the majority class and hide the poor performance on the minority class. All the comparative analysis below is based on the statistical test of Wilcoxon Sign Rank test with Holm-Bonferroni corrections at the significance level of 0.05. Holm-Bonferroni corrections are performed to counteract the problem of multiple comparisons.

2) Results and Analysis: Tables III - IV present the last-step minority-class recall and G-mean from all the methods learning the static data streams. Due to the limited space and great performance difference between methods, the statistical test results of comparison for the static data are not shown in the tables. We can see that the single perceptron classifier is generally not a good base classifier for the three sampling-based ensemble methods, which produce poor recall and G-mean in all cases. Also using perceptrons, RLSACP and WOS-ELM perform better than the perceptron-based ensembles in some cases, such as Gearbox and iNemo for RLSACP and Smart Building and PAKDD for WOS-ELM, but the improvement is limited. The performance of RLSACP seems to be very dependent on data sets. For example, it achieves the highest minority-class recall and G-mean in Gearbox among all, but produces zero in PAKDD and KDD. Besides, although it shows the highest minority-class recall in Smart Building (0.949%), majority-class performance is sacrificed greatly so that final G-mean on this data is only 0.072%.

For tree-based MOSOB, OOB and UOB, they outperform the others significantly in all cases in terms of G-mean, except for Gearbox. Among these three tree-based ensembles, consistent with our previous work [1], UOB is the best significantly in four out of five cases regarding both recall and G-mean. It is more aggressive at finding minority-class

TABLE V: Means and standard deviations of **average G-mean** during every 1000 time steps from moment 1001. P-values of the Wilcoxon Sign Rank tests between MOSOB and every other method are given in brackets, based on 16 comparisons performed on each dynamic data stream. P-values in bold italics indicate statistically significant differences.

	Duration	(1000, 2000]	(2000, 3000]	(3000, 4000]	(4000, 5000]
Gearbox	MOSOB	0.462±0.043	0.497±0.008	0.399±0.046	0.502±0.018
	OOB	0.235±0.022 ***(0.00000)***	0.446±0.020 ***(0.00000)***	0.429±0.057 ***(0.00001)***	0.420±0.019 ***(0.00000)***
	UOB	0.205±0.076 ***(0.00000)***	0.408±0.041 ***(0.00000)***	0.360±0.050 ***(0.00000)***	0.404±0.019 ***(0.00000)***
	RLSACP	0.346±0.060 ***(0.00000)***	0.008±0.019 ***(0.00000)***	0.086±0.146 ***(0.00000)***	0.082±0.144 ***(0.00000)***
	WOS-ELM	0.000±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***
Smart Building	MOSOB	0.769±0.011	0.799±0.019	0.819±0.004	0.850±0.003
	OOB	0.711±0.028 ***(0.00000)***	0.832±0.009 ***(0.00000)***	0.822±0.003 ***(0.00000)***	0.870±0.005 ***(0.00000)***
	UOB	0.776±0.016 ***(0.00002)***	0.807±0.025 ***(0.00052)***	0.826±0.005 ***(0.00000)***	0.840±0.019 ***(0.00007)***
	RLSACP	0.053±0.115 ***(0.00000)***	0.003±0.032 ***(0.00000)***	0.003±0.035 ***(0.00000)***	0.013±0.053 ***(0.00000)***
	WOS-ELM	0.001±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***	0.009±0.002 ***(0.00000)***	0.214±0.050 ***(0.00000)***
PAKDD	MOSOB	0.555±0.009	0.444±0.013	0.589±0.011	0.529±0.011
	OOB	0.564±0.005 ***(0.00000)***	0.453±0.011 ***(0.00000)***	0.599±0.014 ***(0.00000)***	0.540±0.006 ***(0.00000)***
	UOB	0.516±0.016 ***(0.00000)***	0.421±0.006 ***(0.00000)***	0.434±0.023 ***(0.00000)***	0.512±0.031 ***(0.00000)***
	RLSACP	0.000±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***
	WOS-ELM	0.000±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***
iNemo	MOSOB	0.991±0.000	0.997±0.000	0.993±0.000	0.996±0.000
	OOB	0.991±0.000 (NaN)	0.997±0.000 (NaN)	0.993±0.000 (0.08270)	0.996±0.000 (NaN)
	UOB	0.990±0.000 (0.31610)	0.996±0.000 ***(0.00067)***	0.993±0.000 (0.09010)	0.996±0.000 (0.01330)
	RLSACP	0.316±0.175 ***(0.00000)***	0.123±0.133 ***(0.00000)***	0.239±0.150 ***(0.00000)***	0.104±0.092 ***(0.00000)***
	WOS-ELM	0.023±0.046 ***(0.00000)***	0.124±0.159 ***(0.00000)***	0.050±0.101 ***(0.00000)***	0.105±0.159 ***(0.00000)***
KDD	MOSOB	0.977±0.001	0.924±0.004	0.920±0.009	0.998±0.000
	OOB	0.984±0.001 ***(0.00000)***	0.932±0.021 (0.03820)	0.912±0.000 ***(0.00000)***	0.997±0.000 ***(0.00000)***
	UOB	0.978±0.001 ***(0.00470)***	0.925±0.000 (0.04440)	0.921±0.001 (0.13790)	0.998±0.000 (0.70180)
	RLSACP	0.000±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***	0.000±0.000 ***(0.00000)***
	WOS-ELM	0.141±0.041 ***(0.00000)***	0.143±0.110 ***(0.00000)***	0.262±0.160 ***(0.00000)***	0.023±0.073 ***(0.00000)***

examples. As the hybrid model of OOB and UOB, MOSOB's performance always lies in between them. It improves the performance of OOB, but it is not as aggressive as UOB.

Now we divert our attention to dynamic data streams. Because of the class imbalance change and performance resetting at every 1000 time steps, we compare the average prequential G-mean during each static interval, from the instant when the first status change occurs (time step 1001). Minority-class recall is omitted here, because the minority class label changes over time. The perceptron-based MOSOB, OOB and UOB are also not included, for their poor performance in the static data streams. Therefore, five methods are compared in four time intervals from each data stream here. The Wilcoxon Sign Rank test is carried out between MOSOB and each of the others. Sixteen pairs of comparisons are involved for each data stream in this analysis. Table V presents the mean and standard deviation of G-mean and the corresponding statistical test results including p-values. P-values in ***bold italics*** indicate statistically significant differences.

We can see that, different from static cases, UOB is worse than OOB in most cases, because of the lack of training on the majority class before the change. When this class turns into the minority, UOB suffers more performance reduction. OOB is more prepared for dynamic data. With the optimized model weights, MOSOB inherits the characteristic of OOB and presents to be more robust than UOB. Gearbox is a quite difficult data set, based on its overall poor G-mean compared to that of the other data sets. MOSOB performs particularly well on Gearbox. After the imbalance status changes every time, MOSOB achieves the significantly best G-mean in 3 out of 4 intervals, and lies in between OOB and UOB in the remaining one. Smart Building and PAKDD are easier data than Gearbox. UOB is not always worse than OOB in Smart Building. On these two data streams, MOSOB's G-mean lies in between OOB and UOB's G-mean with a significant difference in most cases. iNemo and KDD are the easiest data, which receive G-mean higher than 0.9 from all the three tree-based ensembles. Their G-mean is very close to each other. "NaN" p-value in iNemo means that the two groups of samples for the statistical test are exactly the same. These two data streams are easy enough to counteract the negative effect of class imbalance changes on UOB. Comparing to MOSOB, OOB and UOB, RLSACP and WOS-ELM produce quite low G-mean in all the five data streams. Even though WOS-ELM applies the time-decayed recall for updating misclassification costs, the results are still not good enough. In conclusion, considering all the five cases, tree-based MOSOB is more robust against dynamic imbalance status than UOB.

For a deeper understanding, we observe the optimal model weights of OOB and UOB during the learning process. As we expect, OOB is more likely to have a higher weight than UOB after a few time steps of the status change. Although the time-decayed metrics vary with the current status of the data stream, the model priority does not change frequently between OOB and UOB. Therefore, the performance of MOSOB is quite stable. Because there is no known noise in the data used in this paper, it is hard to say whether MOSOB will become sensitive to noisy data. It is worth looking into this issue in our future work.

In terms of computational time, because of the training of multiple classifiers and the searching effort for optimal

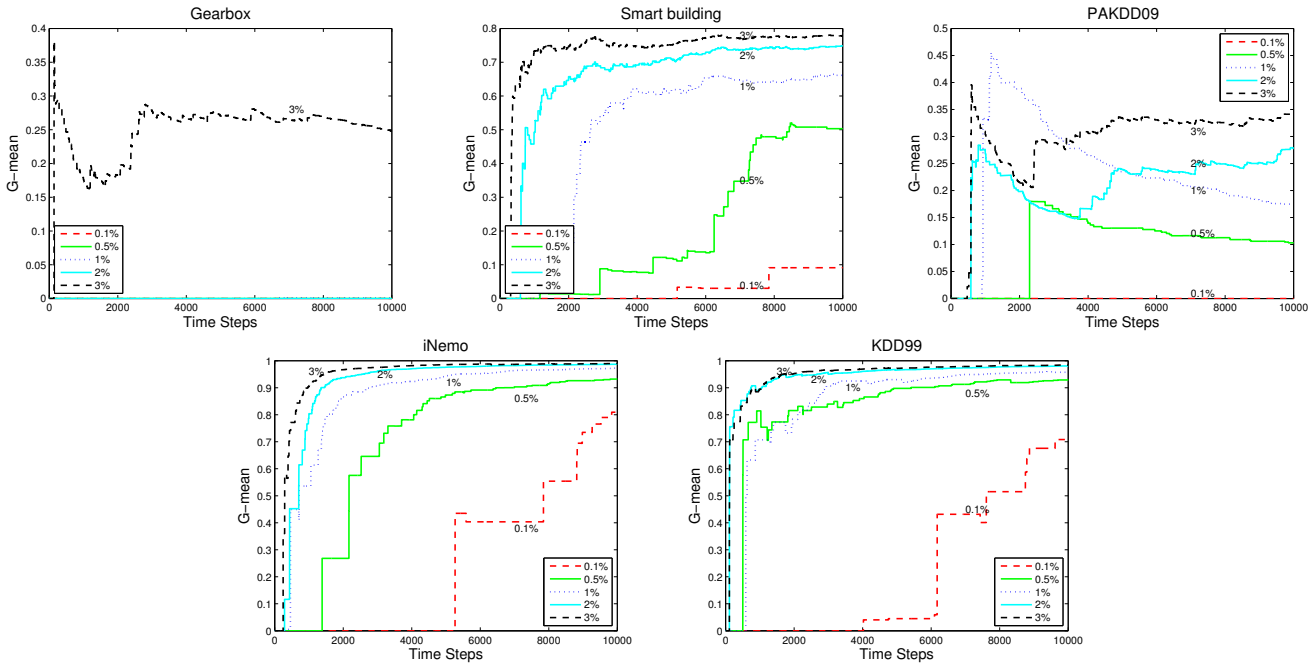

Fig. 1: Prequential G-mean curves of MOSOB on highly imbalanced data streams.

weights, MOSOB shows a much higher computational cost than the other compared methods. It relies on multiple factors, including the choice of base classifiers in OOB and UOB, the ensemble size, and the pool size of candidate weights. A concrete comparison will be included in our future work.

B. Effectiveness on Highly Imbalanced Data Streams

Following on the promising result of MOSOB using decision tree base classifiers, this section aims to understand its effectiveness from a different perspective. Data from many real-world applications with inherent class imbalance problems can be highly skewed. In the fault detection data "Smart Building", for example, we fix the imbalance rate to 5% in the above experiment. In the real world, however, the chance of a sensor in the building becoming faulty is usually very low, such as 0.5%, 0.1% or even lower. As a part of class imbalance study, it is necessary and useful to know how imbalanced the data stream can be, on which the learning algorithm is still effective. Therefore, we focus on highly imbalanced data streams in this section.

1) Data and experimental design: We use the above five data applications and generate very imbalanced data streams, with the length of 10000 time steps. Five data streams are produced for each application, which have a fixed IR $\in \{0.1\%, 0.5\%, 1\%, 2\%, 3\%\}$. For the lowest case IR=0.1%, there are only 10 minority-class examples in the whole data stream. Tree-based MOSOB is applied here with the same settings as in Section IV-A. Average prequential G-mean over 100 runs is recorded at each time step without resetting.

2) Results and Analysis: Fig. 1 compares the prequential G-mean curves with different imbalance rates. For the difficult data Gearbox, G-mean remains zero until IR is increased to 3%. Zero G-mean is caused by zero minority-class recall. For Smart Building and PAKDD with medium classification difficulty, IR=0.1% is still too small for the online model to identify any minority-class examples. G-mean grows faster and earlier as IR increases, when IR$\geq$0.1%. G-mean in PAKDD is decreasing along with learning during some period of time, probably because the data concept is drifting as stated in the data description. For the easiest data iNemo and KDD, MOSOB responds to the minority class in all cases, with G-mean exceeding 0.7 when the learning procedure stops. A higher IR leads to a faster and quicker response. For the most imbalanced case with IR=0.1%, MOSOB does not detect any minority-class examples until time steps 4000-5000. In other words, even for easy data, an online learner armed with class imbalance techniques needs at least 4-5 minority-class examples to learn the concept of this class. Therefore, we do not recommend using this type of learning methods solely learning from very imbalanced data with IR smaller than 0.1%, without any pre-knowledge. Otherwise, auxiliary methods must be applied to improve the prediction accuracy on the minority class, such as one-class learning to train the online model with only one class of data [28] and transfer learning to introduce new data knowledge [29].

V. Conclusions

This paper proposed a multi-objective ensemble method for online class imbalance learning, called MOSOB, which aims to maximize minority-class and majority-class recall simultaneously for the best performance trade-off between classes. It maintains two ensemble models OOB and UOB and finds the Pareto-optimal weights to combine their predictions at each time step. It is motivated by the observation

that UOB has better classification performance and OOB has better robustness against dynamic imbalance status. MOSOB is expected to combine their advantages with improved classification accuracy and learning adaptivity.

The performance of MOSOB is studied from two perspectives. First, we tested MOSOB on ten static and dynamic data streams from five real-world applications, in comparison with OOB, UOB and two state-of-the-art methods. MOSOB using decision tree base classifiers is shown to be the most stable and accurate model that outperforms OOB in static cases in terms of G-mean and minority-class recall and is more robust against class imbalance changes than UOB. Second, we explored its ability in a group of highly imbalanced data streams with fixed IR ranging between 0.1% and 3%. We showed that MOSOB needs at least 3% IR to respond to the minority class on difficult data. IR can be as low as 0.1% on easy data. Otherwise, extra data knowledge and auxiliary methods must be used to help the learning.

Currently, this work is limited to two-class problems (i.e. two objectives). In addition, the searching technique in MOSOB is a brute-force approach, looking for the optimal within a finite set of candidates. More advanced optimization methods will be considered in the future work, such as MOEAs [30]. It is also important to study MOSOB's performance in data streams with classification boundary drift and noise.

ACKNOWLEDGMENT

This work was supported by two EU FP7 Grants (Nos. 270428 and 257906) and a NSFC Grant (No. 61329302). Xin Yao was also supported by a Royal Society Wolfson Research Merit Award.

REFERENCES

[1] S. Wang, L. L. Minku, and X. Yao, "A learning framework for online class imbalance learning," in *IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL)*, 2013, pp. 36–45.

[2] N. C. Oza, "Online bagging and boosting," *IEEE International Conference on Systems, Man and Cybernetics*, pp. 2340–2345, 2005.

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

[4] S. Wang and X. Yao, "Relationships between diversity of classification ensembles and single-class performance measures," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 206–219, 2013.

[5] V. Lopez, A. Fernandez, S. Garcia, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, pp. 113–141, 2013.

[6] U. Bhowan, M. Johnston, M. Zhang, and X. Yao, "Evolving diverse ensembles using genetic programming for classification with unbalanced data," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 368–386, 2013.

[7] P. Soda, "A multi-objective optimisation approach for class imbalance learning," *Pattern Recognition*, vol. 44, no. 8, pp. 1801–1810, 2011.

[8] A. Ghazikhani, R. Monsefi, and H. S. Yazdi, "Recursive least square perceptron model for non-stationary and imbalanced data stream classification," *Evolving Systems*, vol. 4, no. 2, pp. 119–131, 2013.

[9] B. Mirza, Z. Lin, and K.-A. Toh, "Weighted online sequential extreme learning machine for class imbalance learning," *Neural Processing Letters*, vol. 38, no. 3, pp. 465–486, 2013.

[10] S. Chen, H. He, K. Li, and S. Desai, "MuSeRA: Multiple selectively recursive approach towards imbalanced stream data mining," in *International Joint Conference on Neural Networks*, 2010, pp. 1–8.

[11] S. Chen and H. He, "Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach," *Evolving Systems*, vol. 2, no. 1, pp. 35–50, 2010.

[12] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 10, pp. 2283 – 2301, 2013.

[13] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Online learning from imbalanced data streams," in *International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, 2011, pp. 347–352.

[14] L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 4, pp. 619 –633, 2012.

[15] S. Wang, L. L. Minku, D. Ghezzi, D. Caltabiano, P. Tino, and X. Yao, "Concept drift detection for online class imbalance learning," in *International Joint Conference on Neural Networks (IJCNN '13)*, 2013, pp. 1–8.

[16] S. Wang and X. Yao, "Multi-class imbalance problems: Analysis and potential solutions," *IEEE Transactions on Systems, Man and Cybernetics, PartB: Cybernetics*, vol. 42, no. 4, pp. 1119–1130, 2012.

[17] H. W. Kuhn, "Nonlinear programming: a historical view," in *ACM SIGMAP Bulletin*, 1982, pp. 6–18.

[18] "2009 PHM challenge competition data set," The Prognostics and Health Management Society (PHM Society). [Online]. Available: http://www.phmsociety.org/references/datasets

[19] M. P. Michaelides, V. Reppa, C. Panayiotou, and M. 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)*, vol. 8, 2012, pp. 492–497.

[20] C. Linhart, G. Harari, S. Abramovich, and A. Buchris, "PAKDD data mining competition 2009: New ways of using known methods," *New Frontiers in Applied Data Mining, Lecture Notes in Computer Science*, vol. 5669, pp. 99–105, 2010.

[21] K. Bache and M. Lichman, "UCI machine learning repository: http://archive.ics.uci.edu/ml," 2013.

[22] STMicroelectronics, "iNemo: iNErtial MOdule V2 demonstration board." [Online]. Available: http://www.st.com/internet/evalboard/product/250367.jsp

[23] S. Wang, L. L. Minku, and X. Yao, "Online class imbalance learning and its applications in fault detection," *International Journal of Computational Intelligence and Applications*, vol. 12, pp. 1 340 001(1–19), 2013.

[24] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001, pp. 97–106.

[25] MOA Massive online analysis: Real time analytics for data streams. [Online]. Available: http://moa.cms.waikato.ac.nz/

[26] A. P. Dawid and V. G. Vovk, "Prequential probability: Principles and properties," *Bernoulli*, vol. 5, no. 1, pp. 125–162, 1999.

[27] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: One-sided selection," in *Proc. 14th International Conference on Machine Learning*, 1997, pp. 179–186.

[28] N. Japkowicz, C. Myers, and M. A. Gluck, "A novelty detection approach to classification," in *Proceedings of the 14th international joint conference on Artificial intelligence*, 1995, pp. 518–523.

[29] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[30] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, 1995.