A Learning Framework for Online Class Imbalance Learning

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

***Abstract*—Online learning has been showing to be very useful for a large number of applications in which data arrive continuously and a timely response is required. In many online cases, the data stream can have very skewed class distributions, known as class imbalance, such as fault diagnosis of real-time control monitoring systems and intrusion detection in computer networks. Classifying imbalanced data streams poses new challenges, which have attracted very little attention so far. As the first work that formally addresses this problem, this paper looks into the underlying issues, clarifies the research questions, and proposes a framework for online class imbalance learning that decomposes the learning task into three modules. Within the framework, we use a time decay function to capture the imbalance rate dynamically. Then, we propose a class imbalance detection method, in order to decide the current imbalance status in data streams. According to this information, two resampling-based online learning algorithms are developed to tackle class imbalance in data streams. Three basic types of class imbalance change are discussed in our studies. The results suggest the usefulness of the learning framework. The proposed methods are shown to be effective on both minority-class accuracy and overall performance in all three cases we considered.**

I. Introduction

Online learning has been a hot topic in machine learning in recent years. It refers to learning from data streams where data arrive continuously and a timely response is required. It has contributed to various real-world applications, such as sponsored search from web click data [1], credit card transactions [2] and spam filtering [3] [4]. Strictly speaking, different from incremental learning algorithms that process data in batches, 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 [5]. The online learner is not given any process statistics for the observation sequence, and thus no statistical assumptions can be made in advance [6].

Online learning can be considered as a particular case of incremental learning in which the buffer size is set to one for batch processing. So, it can be used for applications where data arrive in batches, by processing each example of the batch separately. An advantage of using online learning is that it avoids the problem of choosing an ideal batch size, which can be difficult when considering changing environments [7]. Furthermore, the online learner can adapt to changes faster than the incremental learner, because it does not have to wait to receive a full data chunk for model update. As it can be updated whenever a new training example is made available, it is particularly useful for applications where data are not necessarily provided in batches, such as the examples above.

Skewed class distributions can be seen in many data stream applications, such as fault diagnosis of control monitoring systems and intrusion detection in computer networks. 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 learning algorithms can suffer great 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 [8]. Unfortunately, existing solutions for class imbalance are limited in the offline mode. Although some incremental learning methods have been proposed to tackle skewed data streams in batches [9] [10] [11] [12], very little work has been done for learning from skewed data streams online.

Online class imbalance learning is a very challenging task, because of the lack of prior knowledge about which classes in data should be regarded as the minority or majority and the uncertainty of when class imbalance happens. Moreover, the underlying data distribution can change considerably over time, referred to as concept drift or learning in nonstationary environments [13]. This brings forth some difficult issues on how to determine the class imbalance status dynamically in data streams and how to adapt the online learner to the class imbalance and new concept effectively. For example, a complex engineering system may become fault prone over time, causing the percentage of faulty and non-faulty examples to change.

As the first work that calls the attention to this problem, this paper looks into the underlying issues, clarifies the research questions, and proposes a learning framework for online class imbalance learning featured by three underlying issues, namely online processing, class imbalance and concept drift. The framework decomposes the learning task into three modules: a class imbalance detector to capture the current class imbalance status; a concept drift detector to find out whether the concept drift occurs; an adaptive online learner to learn the data stream and take corresponding actions when class imbalance and concept drift happen. For the class imbalance detector, we propose a time decay function to report the current distribution of class labels accurately and efficiently, based on which we propose an imbalance detection algorithm to determine whether the current data stream should be regarded as "imbalanced" in real time. If there is class imbalance, our resampling-based Online Bagging methods will apply either ramdom oversampling or undersampling to adjust the learning bias from the majority towards the minority. The results show that both over- and undersampling can improve the accuracy on the current minority class identified by the detector significantly. Particularly, undersampling-based Online Bagging is more aggressive at both recall and G-mean measures. Three types of data streams with different changing severity in class imbalance are considered. One with high severity is shown to be a more difficult case than the others.

The rest of this paper is organized as follows. Section II gives the background knowledge about class imbalance learning and research progress in learning from imbalanced data streams. Section III proposes the learning framework for online class imbalance learning. Section IV defines class imbalance for online scenarios and proposes a class imbalance detection method. Based on the current class imbalance status, Section V proposes adaptive online learning methods to tackle class imbalance online. Section VI draws the conclusions and points out our future work.

II. Related Work

This section introduces the two focal points of this paper. First, we describe what problems class imbalance learning aims to solve and the state-of-the-art methods in this area. Subsequently, we briefly review the current research progress in learning from imbalanced data streams.

A. 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 [8]. 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.

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 [14]. 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" [15].

Numerous methods have been proposed to tackle class imbalance problems 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 and SMOTE [16]. They are simple and efficient, but their effectiveness depends greatly on the problem and training algorithms [17]. 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 [18] and cost-sensitive learning algorithms [19] [8]. Algorithm-level methods require specific treatments for different kinds of learning algorithms, which hinders their use in many applications, since we do not know in advance which algorithm would be the best choice in most cases. In addition to the aforementioned data-level and algorithm-level solutions, ensemble learning [20] has become another major category of approaches to handling imbalanced data by combining multiple classifiers, such as SMOTEBoost [21] and AdaBoost.NC [22] [23]. Ensemble learning algorithms have been shown to be able to combine strength from individual learners and enhance the overall performance [24] [25]. They also offer additional opportunities to handle class imbalance at both the individual and ensemble levels.

Although class imbalance learning has been extensively studied, none of the proposed methods can process imbalanced data online. This paper proposes two online ensemble learning methods based on resampling techniques to deal with imbalanced data streams.

B. Learning from Imbalanced Data Streams

Most existing online learning algorithms assume that the underlying data distribution is balanced. Until recently, few attempts have been made to address the problem of dealing with imbalanced data streams. The first attempt was done by Gao et al. [26] [9]. 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. This method implicitly assumes that the minority-class data is stationary. Once violated, this assumption can lead to the misinterpretation of the true feature space of the minority class at subsequent time steps [12]. Besides, some of the accumulated minority-class data could have become irrelevant over long periods of time. Wang et al. proposed an ensemble algorithm based on clustering and sampling, which has only been tested on a few artificial data sets [10]. K-means clustering algorithm is used to undersample the majority class in the current data chunk, from which a new classifier is built. Chen et al. proposed SERA [27] and its improved versions MuSeRA [28] and REA [29], 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 history 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 [30]. It is shown to be more suitable to extremely complicated data streams with complex concept drift and high degrees of imbalance. Ditzler and Polikar proposed two ensemble approaches, Learn++.CDS and Learn++.NIE [12]. The former applies the well-established oversampling technique SMOTE [16] and the latter uses a Bagging based sub-ensemble method to rebalance data. They are shown to outperform earlier methods in general but at the cost of computational complexity due to the oversampling and sub-ensemble strategies.

All the aforementioned methods, however, need to collect full data chunks for training. Some of them require access to previous training data. 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 [31]. 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. [32] 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. Different from all the existing work, the online learning methods proposed in this paper do not need any prior knowledge about data and storage of old data. A class imbalance detection method is developed to help determine the current class imbalance status in data streams and the sampling parameters for the learner adaptively.

III. A Learning Framework for Online Class Imbalance Learning

In this section, we propose a learning framework that breaks down the online class imbalance learning process into three modules – a class imbalance detector, a concept drift detector and an adaptive online learner, as shown in Fig. 1. Each module handles one major issue of online class imbalance, and communicates with the others for the up-to-date status of data streams. The class imbalance detector reports the class imbalance status of data streams under nonstationary environments. The concept drift detector captures concept drift, including all cases with classification boundary drift, in imbalanced data streams. Based on the information provided by the first two modules, the adaptive online learner determines when and how to respond to the detected class imbalance and concept drift. This is a general framework for dealing with imbalanced data streams with arbitrary number of classes.

A. Class Imbalance Detector

When processing imbalanced data streams online, we first need to find out how imbalanced the data stream is and which classes should be regarded as the minority. The class imbalance detector aims to capture the imbalance status in real time. If the class distribution is too skewed that degrades the learner's performance, some class imbalance techniques must be triggered in a timely fashion. This module should provide the following information:

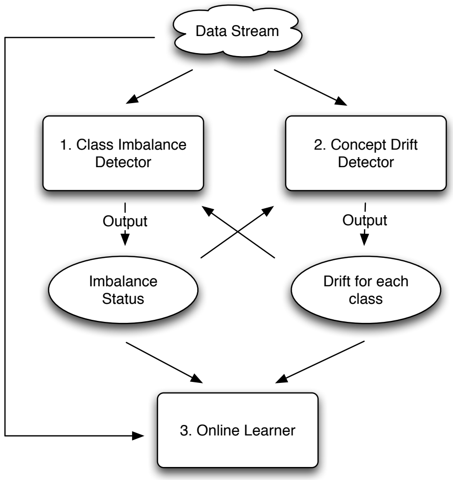


Fig. 1: Learning framework for online class imbalance learning

- Which classes can be regarded as the minority/majority? How many minority/majority classes are there in the data stream?
- How imbalanced is the current data stream? What is the current imbalance degree?
- Which classes are harder to be identified that need more focus from the learner?

What is challenging for the class imbalance detector is to decide the imbalance degree at the current moment in the dynamically changing environment. It could be the case that a class appearing to be the majority due to the small sample size within a short period of time is actually the minority in a longer run. Moreover, a change in class imbalance could happen over time. For example, a new minority or majority class of data could join in the middle of the data stream; the data stream could get imbalanced to a higher degree; the data stream could get imbalanced from balanced; or, in a more extreme case, a minority class could even convert to the majority. An illustrative example would be a fault isolation system, where a certain machine being monitored could present more faults as it becomes older. An effective class imbalance detector should be able to capture the current imbalance status correctly, and inform the online learner to handle class imbalance timely.

B. Concept Drift Detector

The concept drift detector in this framework aims to detect potential changes in the underlying data distribution with classification rules altered. Following the traditional understanding of drift detection, it allows the learner to adapt quickly and accurately to possible changes and maintain its performance. Different from the traditional methods, the detector for imbalanced data streams should be able to sense the drift in minority classes, which would be much harder than dealing with the balanced case.

When there is class imbalance, the traditional methods can become ineffective. It is commonly believed that, when concept drift occurs, a decrease in classification accuracy usually occurs because the training data the learner is built on would be carrying out-of-date concepts [9]. Thus, most existing methods detect drift based on overall accuracy/error made by the learner, such as EDDM (Early Drift Detection Method) [33] and DDM (Drift Detection Method) [34]. However, they may not be appropriate for imbalanced data streams. First of all, the overall accuracy/error is not a good performance measure for class imbalance data, since it is too sensitive to imbalanced degree and cannot reflect the performance on the minority class [8]. The minority class contributes too little to the overall accuracy compared to the majority class. Second, there may not be enough examples from minority classes in new data for the detector to discover the drift. Besides, a drift can have very different impacts on minority and majority classes. A great performance variation caused by the drift in the minority class may not be observed in the majority class. More intractable issues exist in imbalanced cases that hinder the detector from capturing the drift. It would be necessary to discuss concept drift for each individual class.

In our framework for online class imbalance learning, the concept drift detector should not only identify when the drift occurs accurately in terms of each individual class, but also evaluate to what degree the performance of each class is affected. Based on the information from the class imbalance detector, the identified minority class will be given more focus by the concept drift detector. Correspondingly, the following questions should be answered by this module:

- Does concept drift happen to current data?
- Which class's performance is affected the most/least?

It helps the online learner to decide when and how to be adapted to the new concept. It can also warn the class imbalance detector of possible changes in class imbalance. A good concept drift detector should be capable of discovering different types of drifts.

C. Adaptive Online Learner

With the information of class imbalance and concept drift, the online learner needs to take the corresponding action adaptively to maintain its effectiveness on the new concept and minority classes without hurting the performance on the majority class. Different from the traditional online learning methods that aim for high overall accuracy, the online learner needs to handle class imbalance and concept drift simultaneously with the goal of a performance balance among classes. To handle class imbalance in data streams, new class imbalance techniques need to be developed, as the traditional ones are only applicable to a set of stationary data. At the same time, the online learner should be prepared to tackle various concept drift scenarios. To handle concept drift, one can choose to either update the current learner or build a new learner for the new concept. Besides, it would be a good idea to keep the track of the majority-class performance during the learning procedure, in order to avoid too much performance loss.

Generally speaking, the objective of online class imbalance learning can be described as "developing adaptive online learning methods that can identify minority-class data effectively and timely without sacrificing the performance on the majority class". Three important characteristics are desirable for a good solution: a) accuracy – predict the minority class accurately; b) efficiency – give timely response; c) adaptivity – handle nonstationary environments. To achieve the goal, the proposed framework formulates and resolves the learning tasks through three basic modules collaborating with each other. Each module contains very challenging and open research issues to be answered. It's worth mentioning that a change of true imbalance rates in the data stream is a specific type of concept drift, since it causes a change in the joint probability $P\left(x,y\right) = P\left(y\right) \cdot P\left(x|y\right)$ through the term $P\left(y\right)$, where y is the class label of given feature vector x. To simplify our problem and clarify learning tasks, we use class imbalance detector to focus on the change in prior probability $P\left(y\right)$ alone without considering any drift in classification boundaries, whereas concept drift in general may involve changes in the classification boundaries that will be tackled by concept drift detector.

According to the framework, we next define class imbalance in a dynamic environment and develop a class imbalance detection method to determine the current imbalanced status. Then, we propose resampling-based online learning methods to tackle the imbalance. In this paper, we temporarily assume that there are only class imbalance changes affecting $P\left(y\right)$ in data streams. Other types of concept drift will be considered in our future work.

IV. Class Imbalance Detector

In this section, to answer the questions in the module of class imbalance detector, we first define class imbalance in dynamically changing environment by proposing a new updating function for calculating class percentages. It can effectively reflect the current imbalance degree in data streams. We then propose a class imbalance detection method to decide when to invoke the online learner to deal with class imbalance.

A. Define Class Imbalance

Suppose a sequence of examples in the form of pairs (x_t, y_t), arriving one at a time. x_t is a p-dimensional vector belonging to an instance space X observed at time t, and y_t is the corresponding label belonging to the label set $Y = \{c_1, \ldots, c_k, \ldots\}$. Y is automatically extended if any new class joins in the stream. $|Y|$ denotes the number of classes that have appeared so far. Let $H\left(x\right)$ be the online learner, updating its hypothesis $H : X \rightarrow Y$ sequentially with the example at the current time step. When a target example x_{t+1} arrives, the task of online class imbalance learning is to predict its label y_{t+1}, aiming to minimize the cumulative prediction error and maximize the cumulative accuracy on minority classes during the learning process.

To define class imbalance, imbalance rate (IR) is an important indicator to describe how skewed the learning problem is. It is defined as the percentage of the minority class over the whole population representing the true underlying distribution. The smaller the imbalance rate, the more imbalanced the data set. In the offline mode, it can be well estimated based on a set of examples. In the online mode, because no prior knowledge about data is available and no data are stored during the processing, the estimation is not an easy task. Besides, the true IR can drift over time, as explained in section III-A.

In order to find out the true IR quickly, we need to set up a few variables updated at each time step for remembering data properties based on received examples. We use a set of counters recording the size percentage of each class in the data observed so far. Let $w_k^{(t)}$ denote the size percentage of class $c_k \in Y$ at time step t. When a new example x arrives at each time step t $(t = 1, 2, 3, ...)$, $w_k^{(t)}$ is usually calculated as follows:

$$w_k^{(t)} = \frac{(t-1) \cdot w_k^{(t-1)} + [(x, c_k)]}{t}, (k = 1, 2, \ldots, |Y|) \quad (1)$$

where $w_k^{(0)} = 0$ and $[(x, c_k)] = 1$ if the true class label of x is c_k, otherwise 0.

As more data are received, $\{w_k\}$ will get closer to the real class distribution gradually over time. This estimation can work quite well if the class imbalance status is static in the data stream. However, once the status changes, this method can take very long time to reflect the new status. This is because the term $[(x, c_k)]$ contributes to w_k less and less as the total number of examples grows. It would be necessary to emphasize examples from the new imbalance status and reduce the impact of examples representing the previous status. To capture the imbalance change quickly and estimate the imbalance degree accurately, we propose to apply a time decay factor η $(0 < \eta < 1)$ to w_k at each time step, by calculating $w_k^{(t)}$ as follows:

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

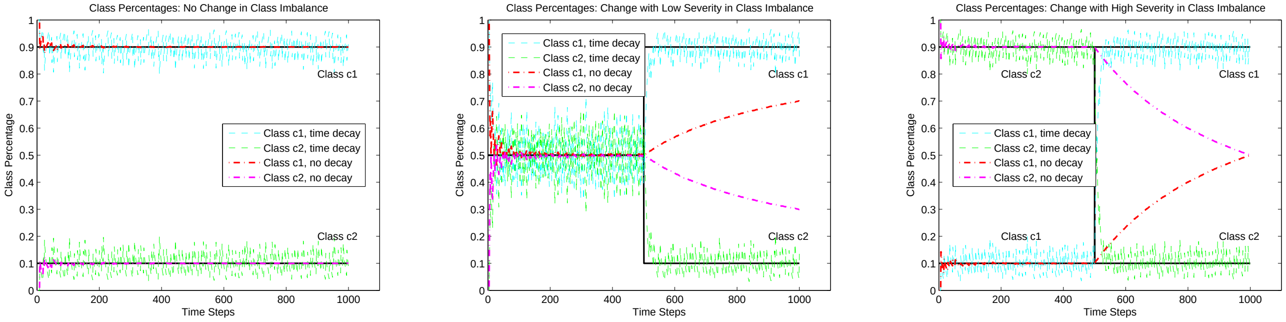
By doing so, older data affect the class percentage less along with time through an exponential smoothing, and w_k is adjusted more based on new data.

1) Data Sets and Experimental Settings: To examine whether our method for updating w_k is effective, we simulate three types of data streams with different severity of class imbalance change based on a real-world data set. The data set is from the PHM society, fault detection competition in 2009 [35]. The task is to detect faults in a running gearbox using accelerometer data and information about bearing geometry. The original data contain more than two types of faults, distributed in different parts of the gearbox. To simplify the problem, we just pick one type of faults that happens to the helical gear with 24 teeth – the gear with a chipped tooth. The data set is thus transformed into a two-class fault detection problem – the gear either in good condition (nonfaulty class) or having a chipped tooth (faulty class). We randomly choose 10 examples as a group from the nonfaulty and faulty classes 100 times without replacement (i.e. there are 1000 examples in total), to form the input data stream. Three data sequences are generated as follows:

- Case 1 (no change in class imbalance): every ten incoming data examples contain nine nonfaulty examples and one faulty example. The true between-class ratio (nonfaulty class to faulty class) in data is fixed to 9:1. Faulty class is regarded as the minority.
- Case 2 (low severity change in class imbalance): every ten incoming data examples contain five nonfaulty and five faulty examples during the first 500 time steps (i.e. data stream appears to be balanced with true between-class ratio 1:1). Afterwards, each group consists of nine nonfaulty examples and one faulty example (i.e. faulty class becomes the minority and the true between-class ratio is 9:1).
- Case 3 (high severity change in class imbalance): every ten incoming data examples contain one nonfaulty example and nine faulty examples during the first 500 time steps (i.e. faulty class appears to be the majority and the true between-class ratio is 1:9). Afterwards, each group consists of nine nonfaulty examples and one faulty example (i.e. faulty class turns into the minority and the true between-class ratio is 9:1).

2) Experimental Results: For each case, class percentages w_k are recorded at each time step, produced by the traditional updating method without time decay (equation 1) and our method with time decay (equation 2). They are shown in Fig. 2, where x-axis indicates the number of training examples seen so far, and y-axis indicates the size percentage of classes c_1 (nonfaulty class) and c_2 (faulty class). The black solid lines represent the true class ratio in data.

For case 1 without change in class imbalance, the true class ratio between c_1 and c_2 is 9:1. Both updating methods can give roughly correct estimation within a reasonable fluctuating range. The traditional method without time decay provides more stable and accurate results than our method as more examples are received. This is because the time decay factor reduces the role of old data, and the information in old data is still helpful in this case. For case 2 with a less severe change in class imbalance, before the change happens, both methods fluctuate at class percentage of 0.5, indicating a balanced class distribution. After the time step 500, when the class distribution becomes very skewed, our method can find true class percentages much more quickly than the other. When the traditional method is still slowly approaching to the black lines, our method has already reached the desired level. Similar results are obtained in case 3 with a severe change in class imbalance. The traditional method will take even longer time to find out the true imbalance rate after it changes. Our updating method responds to the change more quickly, because it is based on less information from the past and more information from recent data.



(a) No change

(b) Change with low severity

(c) Change with high severity

Fig. 2: Comparison of two updating methods of class percentages under the three scenarios with different changing severity of class imbalance status.

B. A Class Imbalance Detection Method

With better recognized class imbalance status, we propose an imbalance detection method to trigger the class imbalance technique of the online learner. Sometimes, imbalanced data does not cause poor performance, which also relies on other factors such as the complexity of data distributions [36]. In this case, it is not necessary to complicate the learning procedure by adding specific techniques for dealing with class imbalance. Based on this fact, such techniques should only be invoked when the classification accuracy on the minority class is very poor compared to the accuracy on the majority class. Thus, our imbalance detector considers both class imbalance status and single-class performance for deciding the current imbalance status. In addition to class percentages w_k, we maintain the current accuracy R_k of the online learner for each class, also known as recall, defined by n_k^+/n_k where n_k^+ denotes the number of correctly classified examples with true label c_k, and n_k denotes the total number of examples with true label c_k received so far. Similarly to w_k, if the current example x has true label c_k, R_k will be updated by $R_k^{(t)} = \eta' R_k^{(t-1)} + (1-\eta')\,[x \leftarrow c_k]$, where η' $(0 < \eta' < 1)$ is a time decay factor for emphasizing the learner's performance at the current moment, and $[x \leftarrow c_k]$ is equal to 1 if x is correctly classified and 0 otherwise. As a key output of class imbalance detector, R_k provides the information of which class receives the worst/best performance from the learner. It can help the online learner to decide which class needs more attention.

The main idea behind this method is, if there are any two classes having w_k difference greater than a threshold δ_1 $(0 < \delta_1 < 1)$ and R_k difference greater than a threshold δ_2 $(0 < \delta_2 < 1)$, then the small class is regarded as the minority and the large class is regarded as the majority. After going through all pairs, the remaining classes are treated as normal. The procedure leads to three label sets – minority-class set Y_{min}, majority-class set Y_{maj} and normal-class set Y_{nom}. It is necessary to have a normal-class set, considering that some easy classes may not need any specific treatment from the online learner. The three label sets are updated at each time step to check whether class imbalance happens. Table I describes the proposed algorithm. It is applicable to a data stream with arbitrary number of classes.

TABLE I: Class imbalance detection algorithm

```
Input: observed class labels $Y = \{c_1, \ldots, c_k, \ldots\}$ in the ascending order based on the current class percentage $w_k$; updated recall $R_k$ of each class; size threshold $\delta_1$; performance threshold $\delta_2$.

Initialize: label sets $Y_{min} = \{\}$, $Y_{maj} = \{\}$ and $Y_{nom} = \{\}$.

for $i = 1$ to $|Y| - 1$ do
    for $j = i + 1$ to $|Y|$ do
        if $w_j - w_i > \delta_1$ and $R_j - R_i > \delta_2$
            $Y_{min} \leftarrow Y_{min} \cup \{c_i\}$
            $Y_{maj} \leftarrow Y_{maj} \cup \{c_j\}$
        end if
    end for
end for

for $k = 1$ to $|Y|$ do
    if $c_k \in Y_{min}$ and $c_k \in Y_{maj}$
        $Y_{maj} \leftarrow Y_{maj} \backslash \{c_k\}$
    end if
    if $c_k \notin Y_{min}$ and $c_k \notin Y_{maj}$
        $Y_{nom} \leftarrow Y_{nom} \cup c_k$
    end if
end for

Output label sets $Y_{min}$, $Y_{maj}$ and $Y_{nom}$.
```

The algorithm is composed of two independent iterations. The first iteration recognizes all potential minority and majority classes. Then all the rest of the classes are inserted into the normal label set in the second iteration. It is worth pointing out that, after the first iteration, one class can appear in both of the minority and majority label sets. This is because the algorithm evaluates the relative imbalance of class pairs – one class can be the majority compared to a very small class, which can also be the minority compared to a very large class. When it happens, we remove the overlapping class from the majority set and treat it as the minority, to guarantee that no minority classes would be ignored. The final output of the algorithm should include three disjoint label sets, whose union is equal to Y. As a special case when there are only two classes in the current data stream, two types of results are possible: 1) there is one minority class and one majority class, and the normal-class set is empty; 2) both classes are in the normal-class set, which means that the current data stream is not thought of as "imbalanced". The effectiveness of this method will be examined through the resampling-based online learner proposed in the next section.

V. Adaptive Online Learner

In this section, we propose two resampling-based ensemble methods, oversampling-based Online Bagging (OOB) and undersampling-based Online Bagging (UOB), to handle imbalanced data streams and make real-time predictions. They process each example strictly online without the need for storage and any prior knowledge about data. When class imbalance is detected by our class imbalance detection method, oversampling or undersampling embedded in Online Bagging [5] will be triggered to either increase the chance of training minority class examples or reduce the chance of training majority class examples. Online Bagging (OB) is a popular ensemble approach that successfully extends the well-known offline ensemble algorithm Bagging [37] 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 $Poisson\,(1)$.

A. Resampling-Based Online Bagging

During the online processing, if the new example (x, c_k) belongs to one of the minority classes ($c_k \in Y_{min}$), OOB will tune the parameter λ of Poisson distribution to $1/w_k$, which indirectly increases the number of copies of the current example K for training. OOB adjusts the class distribution by increasing the number of minority-class examples. In other words, it combines oversampling with Online Bagging. If the new training example (x, c_k) belongs to one of the majority classes ($c_k \in Y_{maj}$), UOB will set λ to $(1 - w_k)$. Training examples from the majority class will be undersampled accordingly through smaller K. The training procedures of OOB and UOB are described in Tables II- III respectively. We can see that the sampling rate for OOB and UOB is automatically decided by w_k through λ. Smaller classes deserve more attention from the learner.

B. Comparative Study

In this section, we compare resampling-based Online Bagging with the original version on the same data streams we generated in Section IV. The experiments here aim to find out whether the class imbalance detector can help to identify the occurrence of class imbalance status in time and whether the proposed online learning algorithms are effective in dealing with class imbalance.

TABLE II: Oversampling-based Online Bagging

```
Input: label sets $Y_{min}$, $Y_{maj}$ and $Y_{nom}$, an ensemble with $M$
base learners, and current training example $(x, c_k)$.

for each base learner $f_m$ $(m = 1, 2, \ldots, M)$ do
    if $c_k \in Y_{min}$
        set $K \sim Poisson\,(1/w_k)$
    else
        set $K \sim Poisson\,(1)$
    end if
    update $f_m$ $K$ times
end for
```

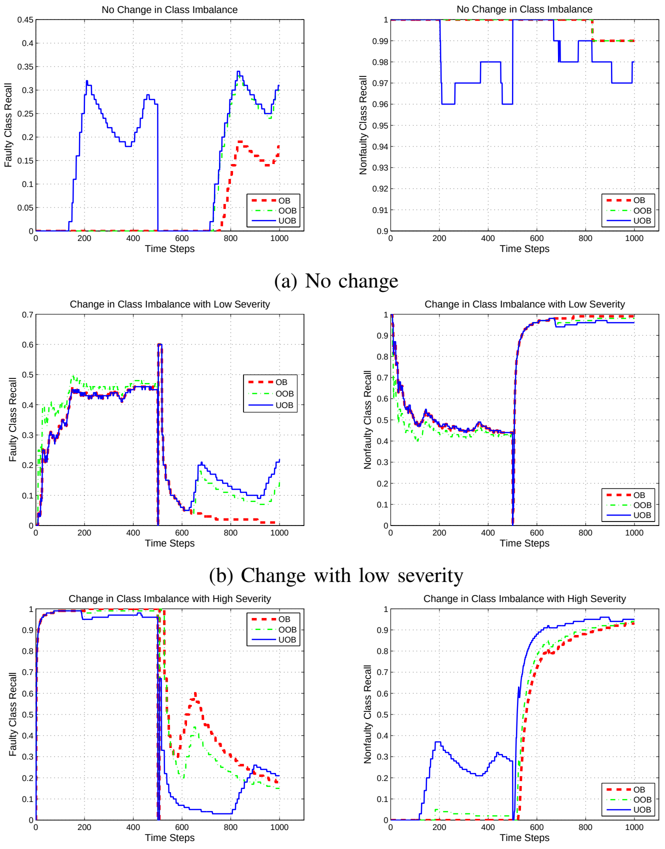
TABLE III: Undersampling-based Online Bagging

```
Input: label sets $Y_{min}$, $Y_{maj}$ and $Y_{nom}$, an ensemble with $M$
base learners, and current training example $(x, c_k)$.

for each base learner $f_m$ $(m = 1, 2, \ldots, M)$ do
    if $c_k \in Y_{maj}$
        set $K \sim Poisson\,(1 - w_k)$
    else
        set $K \sim Poisson\,(1)$
    end if
    update $f_m$ $K$ times
end for
```

Each ensemble model is formed by ten neural networks as the base learner. The performance is evaluated through the prequential test [38], 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 "current" performance at each time step. Because class imbalance change can happen right in the middle of the data stream, we separate prequential performance into two stages for each model by resetting the performance results to 0 at the 500th time step. This ensures that the performance observed after change is not affected by the performance before change, allowing us to analyse the behaviour of the models before and after the change adequately.

For the parameters in class imbalance detector, we set size threshold $\delta_1 = 0.6$ and performance threshold $\delta_2 = 0.4$ according to some preliminary experiments. Higher threshold values diminish the effect of resampling; lower threshold values can lead to unnecessary imbalance alerts and harm the majority-class performance and system stability, when the changes are not so frequent. The time decay factors η and η' for updating w_k and R_k are set to 0.9. The output of class label sets is sent to the online learner at each time step, to decide whether resampling needs to be applied at the current moment. Fig. 3 shows 2-stage prequential recall curves of faulty class (c_2) and nonfaulty class (c_1) produced by OB, OOB and UOB on the 3 data streams with different types of class imbalance change. Each point on the curve is the average of 30 independent runs. We can see how accurately the online learner can recognize data for each class.



(a) No change

(b) Change with low severity

(c) Change with high severity

Fig. 3: Prequential recall curves of faulty class c_2 (left) and nonfaulty class c_1 (right) produced by OB, OOB and UOB on the 3 data streams with different types of class imbalance change.

In the data set with no change in class imbalance, the faulty class is always the minority. OB produced very low recall for this class (Fig. 3(a) left). It remained zero, which means that none of the examples from this class is recognized, until after 700 time steps. Because resampling was triggered by our class imbalance detector at a very early stage, OOB and UOB showed different behaviors. During the first stage of prequential curve (1-500 time steps), OOB did not seem to have helped classify the minority-class data; UOB improved it greatly from 0% to 30%. During the second stage of prequential curve (501-1000 time steps), both OOB and UOB showed better recall. The reason for OOB not being very helpful in the first stage could be that the number of minority-class examples is too small to make an effect on the learner. Besides, oversampling is believed to be a more conservative and stable method than undersampling [17]. For the nonfaulty-class recall in this case (Fig. 3(a) right), UOB has a larger performance drop than OOB, due to the trade-off between two classes.

When the class imbalance status drifts from balanced to imbalanced (change with low severity), different observations were obtained, especially during the first stage. Generally, during the first stage, recall values from both classes got closer to each other as the data stream is balanced. During the second stage, when faulty class becomes the minority, its recall reduced and nonfaulty-class recall increased quickly for all methods. When size difference and recall difference hit the threshold, resampling was triggered by the class imbalance detector, and OOB and UOB started improving minority-class recall (Fig. 3(b) left). Note that several examples of the faulty class have already been used for training during the first half of the learning. That may be the reason why OOB and UOB obtained more similar performance behavior to each other on the minority class in the second half of the learning, in the same way as in the data set with no change.

In the data set with a severe change in class imbalance, i.e. faulty class turns from majority into minority, the faulty class recall was very high during the first stage, then dropped dramatically (Fig. 3(c) left). Correspondingly, nonfaulty-class recall was very low at first, then started increasing as more nonfaulty examples arrived during the second stage (Fig. 3(c) right). The class imbalance detector labelled the nonfaulty class as the minority during the first stage, and labelled the faulty class as the minority during the second stage. Therefore, UOB and OOB produced better nonfaulty-class recall than OB at first. When the status changed abruptly at the 500th time step, UOB and OOB suffered from a larger drop of faulty-class recall than OB, because resampling is still running to emphasize the nonfaulty class. When the faulty class was labelled as the minority, UOB started recovering its recall loss again. For this scenario, OOB and UOB performed better in minority-class recall than OB, but needed longer time to respond to the severe change in class imbalance status. Moreover, OOB recovered faster than UOB from the change, even though in the longer term UOB produced better recalls. A possible reason to that may be also linked to the number of examples that have been already received from the faulty class. At the moment of the change, UOB will have been trained with much less examples of the faulty class than OOB due to the undersampling of the faulty class. As OOB has been trained with more examples of the faulty class, it is better prepared for when the faulty class becomes the minority. This scenario is harder to be tackled than the previous two.

Generally speaking, the class imbalance detector can capture the class imbalance status correctly even when there is a status change in the data stream, and our resampling-based Online Bagging methods show better performance on the current minority class than the original Online Bagging accordingly. Particularly, undersampling-based method is shown to be a better choice than oversampling-based one. It is more aggressive at finding minority-class examples. When a severe change in class imbalance happens, however, undersampling-based method may suffer from a significant performance drop at the beginning, then takes longer time to recover its performance, possibly because it has been trained with fewer examples of the new minority class than other methods such as OB and

TABLE IV: Means and standard deviations of G-mean from OB, OOB and UOB at time steps 500 and 1000 on the three data sets. P-values of the Wilcoxon Sign Rank tests between UOB and OB/OOB 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 twelve comparisons performed. UOB's G-mean is shown in bold italics when it is significantly better than both OB and OOB.

	No change in class imbalance		Imbalance change with low severity		Imbalance change with high severity	
	Time step 500	Time step 1000	Time step 500	Time step 1000	Time step 500	Time step 1000
OB	0.000±0.000 ***(0.00000)***	0.420±0.040 ***(0.00000)***	0.443±0.009 (0.73430)	0.086±0.072 ***(0.00000)***	0.000±0.000 ***(0.00000)***	0.408±0.026 ***(0.00011)***
OOB	0.000±0.000 ***(0.00000)***	0.545±0.003 (0.21030)	0.443±0.009 (0.59990)	0.384±0.029 ***(0.00000)***	0.122±0.048 ***(0.00000)***	0.372±0.019 ***(0.00000)***
UOB	***0.507±0.022***	0.550±0.020	0.444±0.010	***0.462±0.042***	***0.517±0.020***	***0.443±0.028***

OOB. It can be fixed by further improving our class imbalance detector and adaptive online learner, which will be incuded in our future work.

To compare the overall performance of the three online learners, we present their prequential G-mean [39] and standard deviation at specific time steps 500 and 1000 in Table IV. G-mean is a better overall performance measure than overall accuracy for imbalanced data. It is defined as the geometric mean of recalls of all classes and shown to be insensitive to the imbalance degree [8]. Wilcoxon Sign Rank tests with Holm-Bonferroni corrections at the overall level of significance of 0.05 were performed to compare UOB and OB/OOB. The Holm-Bonferroni corrections consider the twelve comparisons performed to counteract the problem of multiple comparisons. UOB's G-mean is shown in bold when it is significantly better than both OB and OOB according to the statistical tests.

The results in the table tally with our observations from the prequential curves of recall. UOB achieves the best overall performance at the end of two stages when class imbalance exists. OOB also performs better than OB, after seeing more minority-class examples.

VI. Conclusions

As the first work that formally addresses online class imbalance learning, this paper formulates the problem by proposing a learning framework, which clarifies underlying issues and decomposes the learning tasks into three modules: a class imbalance detector to capture the current class imbalance status; a concept drift detector to find out whether the concept drift occurs; an adaptive online learner to learn the data stream and take corresponding actions when class imbalance and concept drift happen.

For the class imbalance detector, we propose a time decay function to report the current distribution of class labels. Its performance is tested on three types of data streams with different changing severity in class imbalance. It is shown to produce more accurate class percentages within a much shorter period of time than the traditional updating method. Based on the up-to-date class imbalance rate and performance of the online learner, we propose an imbalance detection algorithm to determine whether the current data stream should be regarded as "imbalanced" and which classes should belong to the minority and majority.

For the adaptive online learner, we propose two resampling-based ensemble methods that combine random oversampling and undersampling with Online Bagging to handle class imbalance by adjusting the parameter in Poisson distribution. They process data online strictly, without the need of any prior knowledge about data and storage of old examples. The resampling technique will be triggered once a class imbalance status is reported by the class imbalance detector at any time step. The performance of the two methods in collaboration with the class imbalance detector is examined through two-stage prequential test of recall, showing the current accuracy for each class along with time. Overall performance measure G-mean is also compared at the final moments before and after the imbalance status changes. The results show that both over- and undersampling can improve the accuracy on the current minority class identified by the detector significantly. Particularly, undersampling-based Online Bagging is more aggressive at both recall and G-mean measures. When there is a severe change, it suffers from longer time to retrieve its effectiveness on the new minority class.

In the near future, we would like to further improve our class imbalance detector and online learners to better deal with various types of class imbalance change, including abrupt/gradual changes and the cases with the presence of noisy data. How the parameters in the algorithms affect their performance, such as size and performance threshold values, would also be explored under different scenarios. All the methods proposed in this paper are applicable to data streams with arbitrary number of classes, but we have only discussed two-class cases so far. We would like extend our study to multi-class problems. In addition, there are still many interesting questions that haven't been answered within the framework. For example, our next step is to look into the module of concept drift detector, aiming to develop effective and efficient algorithms that can find out different drifts with changing classification boundaries in imbalanced data streams.

Acknowledgment

This work was supported by the European funded project (FP7 Grant No. 270428) "iSense: making sense of nonsense" and the EPSRC funded project (Grant No. EP/J017515/1) "DAASE: Dynamic Adaptive Automated Software Engineering".

REFERENCES

[1] M. Ciaramita, V. Murdock, and V. Plachouras, "Online learning from click data for sponsored search," in *International World Wide Web Conference*, 2008, pp. 227–236.

[2] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 226–235.

[3] K. Nishida, S. Shimada, S. Ishikawa, and K. Yamauchi, "Detecting sudden concept drift with knowledge of human behavior," in *IEEE International Conference on Systems, Man and Cybernetics, 2008.*, 2008, pp. 3261–3267.

[4] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A case-based technique for tracking concept drift in spam filtering," *Knowledge-Based Systems*, vol. 18, no. 4-5, pp. 187–195, 2005.

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

[6] C. E. Monteleoni, "Online learning of non-stationary sequences," Massachusetts Institute of Technology, Tech. Rep., 2003.

[7] L. L. Minku, "Online ensemble learning in the presence of concept drift," Ph.D. dissertation, School of Computer Science, University of Birmingham, 2010.

[8] 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.

[9] J. Gao, B. Ding, J. Han, W. Fan, and P. S. Yu, "Classifying data streams with skewed class distributions and concept drifts," *IEEE Internet Computing*, vol. 12, no. 6, pp. 37–49, 2008.

[10] Y. Wang, Y. Zhang, and Y. Wang, "Mining data streams with skewed distribution by static classifier ensemble," *Opportunities and Challenges for Next-Generation Applied Intelligence, Studies in Computational Intelligence*, vol. 214, pp. 65–71, 2009.

[11] G. Ditzler, M. D. Muhlbaier, and R. Polikar, "Incremental learning of new classes in unbalanced datasets: Learn++.udnc," *Multiple Classifier Systems, Lecture Notes in Computer Science*, vol. 5997, pp. 33–42, 2010.

[12] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, 2012 (DOI: 10.1109/TKDE.2012.136).

[13] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 730–742, May 2010.

[14] G. M. Weiss, "Mining with rarity: a unifying framework," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 7–19, 2004.

[15] S. Wang, "Ensemble diversity for class imbalance learning," Ph.D. dissertation, School of Computer Science, The University of Birmingham, 2011.

[16] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 341–378, 2002.

[17] A. Estabrooks, T. Jo, and N. Japkowicz, "A multiple resampling method for learning from imbalanced data sets," in *Computational Intelligence 20*, vol. 20, no. 1, 2004, pp. 18–36.

[18] N. Japkowicz, C. Myers, and M. A. Gluck, "A novelty detection approach to classification," in *IJCAI*, 1995, pp. 518–523.

[19] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, 2006, pp. 63– 77.

[20] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1–39, 2010.

[21] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *Knowledge Discovery in Databases: PKDD 2003*, vol. 2838, 2003, pp. 107–119.

[22] S. Wang, H. Chen, and X. Yao, "Negative correlation learning for classification ensembles," in *International Joint Conference on Neural Networks, WCCI.* IEEE Press, 2010, pp. 2893–2900.

[23] S. Wang and X. Yao, "The effectiveness of a new negative correlation learning algorithm for classification ensembles," in *IEEE International Conference on Data Mining Workshops*, 2010, pp. 1013–1020.

[24] ——, "Relationships between diversity of classification ensembles and single-class performance measures," *IEEE Transactions on Knowledge and Data Engineering*, 2011 (DOI: 10.1109/TKDE.2011.207).

[25] ——, "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.

[26] J. Gao, W. Fan, J. Han, and P. S. Yu, "A general framework for mining concept-drifting data streams with skewed distributions," in *Proceedings of SIAM ICDM*, 2007.

[27] S. Chen and H. He, "Sera: Selectively recursive approach towards nonstationary imbalanced stream data mining," in *International Joint Conference on Neural Networks*, 2009, pp. 522–529.

[28] 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.

[29] 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.

[30] R. N. Lichtenwalter and N. V. Chawla, "Adaptive methods for classification in arbitrarily imbalanced and drifting data streams," *New Frontiers in Applied Data Mining, Lecture Notes in Computer Science*, vol. 5669, pp. 53–75, 2010.

[31] 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.

[32] 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.

[33] M. Baena-Garcia, J. del Campo-Avila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *The Forth ECML PKDD International Workshop on Knowledge Discovery From Data Streams (IWKDDS'06)*, 2006.

[34] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," *Advances in Artificial Intelligence*, vol. 3171, pp. 66–112, 2004.

[35] 2009 PHM challenge competition data set. The Prognostics and Health Management Society (PHM Society). [Online]. Available: http://www.phmsociety.org/references/datasets

[36] T. Jo and N. Japkowicz, "Class imbalances versus small disjuncts," in *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, 2004, pp. 40–49.

[37] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

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

[39] 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.