AUC Estimation and Concept Drift Detection for Imbalanced Data Streams with Multiple Classes

1st Shuo Wang
School of Computer Science
University of Birmingham
Birmingham, UK
s.wang@cs.bham.ac.uk

2nd Leandro L. Minku
School of Computer Science
University of Birmingham
Birmingham, UK
l.l.minku@cs.bham.ac.uk

***Abstract*—Online class imbalance learning deals with data streams having very skewed class distributions. When learning from data streams, concept drift is one of the major challenges that deteriorate the classification performance. Although several approaches have been recently proposed to overcome concept drift in imbalanced data, they are all limited to two-class cases. Multi-class imbalance imposes additional challenges in concept drift detection and performance evaluation, such as a more severe imbalanced distribution and the limited choice of performance measures. This paper extends AUC for evaluating classifiers on multi-class imbalanced data in online learning scenarios. The proposed metrics, PMAUC, WAUC and EWAUC, are studied through comprehensive experiments, focusing on their characteristics on time-changing data streams and whether and how they can be used to detect concept drift. The AUC-based metrics show effectiveness in detecting concept drift in a variety of artificial data streams and a real-world data application with multiple classes. In particular, EWAUC is shown to be both effective and efficient.**

***Index Terms*—Class imbalance learning, Online learning, Concept drift detection**

I. Introduction

Online class imbalance learning has been drawing growing attention from both academia and industry. It deals with data streams with imbalanced class distributions in an online fashion, and commonly exists in real-world applications, such as spam detection, fraudulent bank transactions, and social media analysis.

Online learning processes each data example for training once on arrival without storage and reprocessing [1], and provides real-time predictions. Because it deals with data sequences rather than a static data set, dynamic changes may occur in the data distribution, a.k.a. concept drift. Concept drift detection is a major challenge in the area of data stream learning, as concept drift compromises the classification performance over time and can even cause catastrophic failure [2].

There are three fundamental forms of concept drift [3]: 1) a change in class prior probabilities $P(y)$; 2) a change in class-conditional probability density function $p(\boldsymbol{x} \mid y)$; 3) a change in posterior probability $P(y \mid \boldsymbol{x})$. Only the third form causes classification boundary changes, potentially leading to the most severe performance reduction. It is called a "real" concept drift [4], which is the focus of this paper. In the following sections, concept drift detection implicitly refers to detecting a real concept drift.

Class imbalance learning has been widely discussed in offline machine learning. It refers to a type of classification problems, where some classes of data (minority) are heavily under-represented compared to other classes (majority). This skewed distribution between classes can cause learning bias towards the majority class and thus poor generalization [5]. It has been extended to online learning scenarios in recent years, but is limited to data with two classes [6].

Imbalanced data problems with more than two classes (i.e. multi-class imbalance) are widely seen. For example, in topic recognition of social media analysis, social media messages involve various topics, such as book, sports, music, politics, etc. Multi-class imbalance suffers more learning difficulties, because it increases data complexity and aggravates the imbalanced distribution [7] [8]. However, none of the existing methods are designed for non-stationary data with multi-class imbalance.

AUC is the most popular measure for evaluating classifiers on two-class data sets. Its online variant PAUC [9] has been shown to be an effective indicator of concept drift in two-class data streams [9] [6]. This paper is thus motivated to explore multi-class AUC for both performance estimation and concept drift detection in imbalanced data streams. We look into three research questions: *Q1) How to generalize AUC to multi-class online scenarios? Q2) Once a multi-class AUC is developed for streaming data, how well does it reflect the current performance of online classifiers, including performance variations resulting from concept drift? Q3) When used as a part of a concept drift detection method, how well does it detect concept drift and improve classification performance in multi-class imbalanced data streams?* To the best of our knowledge, an AUC extension that is suitable for multi-class data streams remains unexplored. This is the very first work of concept drift detection for multi-class imbalanced problems.

For Q1 (Sections III and IV), we define three new metrics, PMAUC, WAUC and EWAUC. For Q2 (Section V), we study their properties by visualizing their values on stationary and drifting data streams in comparison with G-mean [10]. For Q3 (Section VI), we discuss whether these AUC estimates are useful for concept drift detection and benefit the classification

performance overtime, through a variety of data streams with different settings.

II. Background

In this section, we introduce the existing active drift detection approaches for class imbalanced data, review the current research progress in multi-class imbalance learning, and discuss the existing multi-class extensions for AUC. The research gap in multi-class imbalanced data streams is recognized.

A. Active Drift Handling Approaches

An active drift handling approach consists of a change detector aiming to sense and report the drift accurately and timely, and an adaptation mechanism aiming to maintain the performance of the classifier by reacting to the detected drift. The change detector inspects certain features extracted from the incoming data stream (indicators), such as the sample mean/variance or the predictive performance. A thresholding mechanism or a hypothesis test adopted by the change detector is constantly applied during the data arrival to determine if any change occurs based on the chosen indicator [6]. Once a change has been detected, the adaptation mechanism is triggered, so that the classifier can adapt to the change.

A few active approaches for class imbalanced data streams have been proposed in recent years focusing on binary problems. A detailed comparison and analysis of these approaches can be found in [6]. Two of the significant findings were: 1) the active approaches outperformed the passive ones; 2) PAUC-PH [9], one of the active approaches, brought significant classification improvement. It uses PAUC, an AUC-based metric, as an indicator monitored by the Page-Hinkley (PH) test [11], and retrain the classifier once a drift alarm is triggered.

B. Multi-Class Imbalance Problems

Multi-class imbalance learning has attracted great attention in recent years. The existing solutions can be classified into two categories: class decomposition approaches and ad-hoc learning algorithms. A class decomposition approach converts a multi-class problem into a set of two-class subproblems. There are two commonly used decomposition schemes: one-versus-one approaches (OVO) and one-versus-all approaches (OVA). When one class is chosen to be the positive in a two-class subproblem, OVO picks a different class to be the negative (e.g. Zhang et al.'s DRCW-ASEG [12], Cerf et al.'s OVE framework [13]), and OVA merges all the remaining classes as the new negative (e.g. Yang et.al.'s IEFS framework [14], Chen et al.'s M^3-SVM [15]). Ad-hoc learning algorithms deal with multi-class data directly, such as multi-class versions of extreme learning machines [16] and Multi-class Roughly Balanced Bagging [17].

To evaluate the classification performance on multi-class imbalance data, G-mean [18] is the most popular "overall" performance metric, which can reflect the accuracy over all classes. It is preferable to the traditional accuracy in class imbalance learning, because it is insensitive to the class distribution. G-mean can be easily used in multi-class cases, which is defined as the geometric mean of recall of all classes [19]. One issue of G-mean is that it depends on the choice of the classification threshold, which is not a problem for AUC. The original AUC, however, is only suitable for offline two-class data scenarios. Further details on AUC and its extensions to other scenarios are given in the next section.

C. AUC and Its Extensions

As the most commonly used measure of classification performance, especially in class imbalance learning, the Area Under the ROC Curve (AUC) summarizes all possible trade-offs between the true and false positive rates of a binary classifier by varying the decision threshold. It has the attractive property of being insensitive to class prevalence and misclassification costs. A few attempts have been made for AUC's multi-class extension, referred to as Volume Under the ROC hyper-Surface (VUS). The existing findings show that a full extension of the two-class ROC analysis is theoretically possible [20], but the calculation of VUS remains computationally infeasible [21].

One approach to high-dimensional ROC surfaces is two-class approximation, decomposing the whole data set into several two-class subsets and applying the two-class methodology directly. Two forms of two-class approximation exist, similar to the data decomposition schemes in Section II-B: one-versus-all (OVA) and one-versus-one (OVO). The former, first used by Provost and Domingos, averages AUC scores between each class and all remaining classes, weighted by the class's frequency [10]. The advantage of this approach is that the computational complexity only increases linearly with the number of classes C, and the results are easy to visualize. However, it becomes sensitive to changes in class priors. The latter, called MAUC in the literature, averages AUC scores between all class pairs [22]. It keeps the property of the traditional AUC, but it has higher complexity involving $C^2 - C$ pairwise comparisons.

In spite of the wide use of AUC and the research effort into its multi-class extensions, the calculation and the high computational complexity restrict their application to offline learning. To address the limitation, Brzezinski and Stefanowski recently proposed Prequential AUC (PAUC), which makes use of a sliding window as a forgetting mechanism for assessing classifiers on evolving data streams and incorporates the red-black tree data structure to improve computational efficiency [23]. It was shown to be statistically consistent and comparably discriminating on stationary data, and a good indicator of concept drifts, especially abrupt concept drifts, on non-stationary data [6]. It is only suitable for two-class data streams.

III. Q1: Prequential Multi-Class AUC

We define Prequential Multi-Class AUC (PMAUC) in this section. It combines PAUC and MAUC to obtain the AUC estimate for multi-class data streams based on OVO. The sliding window allows us to compute AUC incrementally after every new example arrives, based on the most recent data. The pairwise computation based on any two classes in data allows

us to use binary approaches, while holding the insensitivity to class prior changes.

For any input data x, assume that the classifier can provide estimates of the probability of x belonging to each class $\hat{p}(i \mid x)$ for $i = 1, \ldots, C$. The classifier outputs a set of $\{\hat{p}(1 \mid x), \hat{p}(2 \mid x), \ldots, \hat{p}(C \mid x)\}$ with their summation equal to 1. For any pair of classes i and j, we follow the notations in [22], and define $\hat{A}(i \mid j)$ as the probability that a randomly drawn member of class j will have a lower estimated probability of belonging to class i than a randomly drawn member of class i. For instance, given a set of data examples with two possible class labels 1 and 2, $\hat{A}(1 \mid 2)$ is how likely an example x of class 2 will be predicted with $\hat{p}(1 \mid x) < \hat{p}(1 \mid x')$, where x' is a randomly drawn example of class 1. Thus, a higher $\hat{A}(i \mid j)$ implies more accurate prediction.

Based on the definition of the ROC curve, AUC between classes i and j can be estimated by $\hat{A}(i,j) = \left(\hat{A}(i \mid j) + \hat{A}(j \mid i)\right)/2$. Note that $\hat{A}(i \mid j)$ is not necessarily equal to $\hat{A}(j \mid i)$ in multi-class cases, although $\hat{A}(1 \mid 2) = \hat{A}(2 \mid 1)$ in binary cases. This will be explained through an example below.

Given the data examples in the current window, to calculate $\hat{A}(i \mid j)$ efficiently, $\hat{p}(i \mid x)$ will be sorted in descending order and maintained in a "red-black tree" [24]. A red-black tree is a self-balancing binary search tree that can complete insertion and deletion operations in $O(\log n)$ time without additional memory cost. The method of calculating $\hat{A}(i \mid j)$ is to count the number of pairs of class i and class j examples such that the former has a higher probability than the latter, and then to normalize this value by all possible pairs $n_i n_j$, where n_i is the number of class i examples and n_j is the number of class j examples in the window [22].

Table I gives an example of calculating $\hat{A}(i,j)$ in a three-class case. Suppose the window W has six examples $\{x_1, \ldots, x_6\}$. Their true class labels are given in the first row of the table, following the example notation. The numbers in each column are the predicted probabilities for the corresponding example.

TABLE I: A three-class data example with labels 1, 2 and 3.

$\hat{p}(i \mid x)$	x_1 (C1)	x_2 (C1)	x_3 (C2)	x_4 (C1)	x_5 (C2)	x_6 (C1)
C1	0.9	0.7	0.6	0.3	0.2	0.1
C2	0.04	0.25	0.2	0.15	0.1	0.7
C3	0.06	0.05	0.2	0.55	0.7	0.2

In this example with classes 1, 2 and 3, when calculating $\hat{A}(1 \mid 2)$, we treat class 1 as the positive and class 2 as the negative class, and sort the examples based on $\hat{p}(1 \mid x)$ in the descending order (the first row in the table). The sorted examples have the following classes: +,+,-,+,-,+. By adding the number of positive examples before each negative one, we obtain $\hat{A}(1 \mid 2) = (2+3)/(4 \times 2) = 0.625$. When calculating $\hat{A}(2 \mid 1)$, we treat class 2 as the positive and sort the examples based on $\hat{p}(2 \mid x)$ (the second row in the table), obtaining the following class sequence: -,-,+,-,+,-. Thus, $\hat{A}(2 \mid 1) = (0+0+1+2)/(2 \times 4) = 0.375$. As you can see, $\hat{A}(1 \mid 2)$ is not equal to $\hat{A}(2 \mid 1)$. $\hat{A}(1,2)$ is their average, equal to 0.5.

After obtaining all $[C(C-1)]$ pairs of $\hat{A}(i,j)$, the overall AUC estimate of the classifier is their average:

$$PMAUC = \frac{2}{C(C-1)} \sum_{i<j} \hat{A}(i,j)$$

The detailed calculation is presented in Table II. During online learning, when a new example arrives, the data in the current window and the $[C(C-1)]$ corresponding red-black trees will be updated for calculating PMAUC (lines 1-10). The time complexity of updating the trees is in $O\left(|C|^2 \log d\right)$, where d is the window size. The time complexity of calculating all $A(i \mid j)$ values (lines 11-30) is in $O\left(|C|^2 d\right)$, Therefore, the overall time complexity of computing PMAUC at each time step is in $O\left(|C|^2 d\right)$.

TABLE II: Calculating PMAUC.

Input: W: a window of examples; d: window size; $Y = \{1, 2, \ldots, C\}$: class label set; (x_t, y_t): the example received at the current moment t with true class label $y_t \in Y$; n_i: the number of class i examples in W; $tree(i \mid j)$: the red-black tree when treating class i as the positive and class j as the negative.

Output: $pmauc$: PMAUC value at time step t.

Initialize: $A(i \mid j) = 0$: pairwise AUC when treating class i as the positive and class j as the negative, where $i, j = 1, 2, \ldots, C$ and $i \neq j$.

1. n = $\sum_{i=1}^{C} n_i$
2. **if** $n = d$ %when window is full, remove the oldest and add the new.
3. **for all** trees involving class y_{t-d+1}
4. tree.remove(x_{t-d+1})
5. **end for**
6. $n_{y_{t-d+1}} = n_{y_{t-d+1}} - 1$
7. **for all** trees involving class y_t
8. tree.add(x_t)
9. **end for**
10. $n_{y_t} = n_{y_t} + 1$
11. **for any** 2 classes i and j in W
12. $s = 0$
13. **for all** sorted examples $(x, y) \in tree(i \mid j)$
14. **if** $y = i$
15. $s = s + 1$
16. **else**
17. $A(i \mid j) = A(i \mid j) + s$
18. **end for**
19. $A(i \mid j) = A(i \mid j) / (n_i n_j)$
20. $s = 0$
21. **for all** sorted examples $(x, y) \in tree(i \mid j)$
22. **if** $y = j$
23. $s = s + 1$
24. **else**
25. $A(j \mid i) = A(j \mid i) + s$
26. **end for**
27. $A(j \mid i) = A(j \mid i) / (n_i n_j)$
28. **end for**
29. $pmauc = \frac{\sum_{i \neq j} A(i|j)}{C(C-1)}$
30. **return** pmauc

*The code is made available at GitHub: https://github.com/shuo-wang/Data-Stream-Learning.

IV. Q1: Weighted Multi-Class AUC and Equal Weighted Multi-Class AUC

In this section, we extend Provost and Domingos' weighted AUC [10] [25] for data stream learning. We denote the extension by WAUC. WAUC applies the OVA scheme for

multi-class decomposition and the sliding window strategy for incremental update. Each class decomposition is weighted by the class's frequency based on the samples in the window, thus it is sensitive to class imbalance. For a fair comparison of AUC estimates between OVA and OVO schemes, we also propose and discuss equal weighted AUC, denoted by EWAUC. The only difference between WAUC and EWAUC is that, EWAUC is calculated with an equal weight $1/C$ assigned to each class decomposition. Both WAUC and EWAUC output the average AUC score of C class decompositions. The detailed calculation is given in Table III.

TABLE III: Calculating WAUC and EWAUC.

Input: W: a window of examples; d: window size; $Y = \{1, 2, \ldots, C\}$: class label set; (x_t, y_t): the example received at the current moment t with true class label $y_t \in Y$; n_i: the number of class i examples in W; $tree\,(i \mid \bar{i})$: the red-black tree when treating class i as the positive and the remaining classes as the negative.

Output: $wauc$ and $ewauc$: WAUC and EWAUC values at time step t.

Initialize:
$A\,(i \mid \bar{i}) = 0$: AUC when treating class i as the positive and the remaining classes as the negative, where $i, j = 1, 2, \ldots, C$.

1. n = $\sum_{i=1}^{C} n_i$
2. **if** $n = d$ %when window is full, remove the oldest and add the new.
3. **for each** class i
4. $tree\,(i \mid \bar{i})$.remove(x_{t-d+1})
5. **end for**
6. $n_{y_{t-d+1}} = n_{y_{t-d+1}} - 1$
7. **for each** class i
8. $tree\,(i \mid \bar{i})$.add(x_t)
9. **end for**
10. $n_{y_t} = n_{y_t} + 1$
11. **for each** class i
12. $s = 0$
13. **for all** sorted examples $(x, y) \in tree\,(i \mid j)$
14. **if** $y = i$
15. $s = s + 1$
16. **else**
17. $A\,(i \mid \bar{i}) = A\,(i \mid \bar{i}) + s$
18. **end for**
19. $A\,(i \mid \bar{i}) = A\,(i \mid \bar{i}) \,/\, (n_i\,(n - n_i))$
20. **end for**
21. $wauc = \sum_{i=0}^{C-1} (n_i/n)\, A\,(i \mid \bar{i})$
22. $ewauc = \sum_{i=0}^{C-1} (1/C)\, A\,(i \mid \bar{i})$
30. **return** wauc, ewauc

Because C red-black trees are maintained for the calculation, the overall time complexity of computing WAUC and EWAUC at each time step is in $O\,(|C|\,d)$, which is more efficient than computing PMAUC.

V. Q2: Properties of Multi-Class AUC Metrics in Imbalanced Data Streams

In this section, we look into the properties of the AUC estimates, i.e. PMAUC, WAUC and EWAUC, by tracking and visualizing their values over time on stationary and drifting imbalanced data streams, in comparison with G-mean. The visualization helps us to understand how these metrics behave over time on different types of data streams, which will further motivate our study on concept drift detection.

A. Experimental Settings

We generate four artificial data streams with 5000 examples (namely 5000 time steps). Each example is formed of two numeric attributes and a class label that can be one of the four class values $\{c1, c2, c3, c4\}$. Two classes, c1 and c2, are chosen to be the minority with an imbalance ratio (IR) equal to 3:7. Each class follows the multivariate Gaussian distribution, where the mean and covariance matrix are randomly generated. The means range within [0, 10].

The first data stream ('DataStatic') follows a static distribution without any changes.

The second data stream ('DataPrior') involves a class prior probability change after time step 2500, but no real concept drift occurs. In other words, the classes c1 and c2 are the minority before time step 2500, after which c3 and c4 become the minority with the same IR.

The third data stream ('DataDrift30%') involves a concept drift with 30% severity after time step 2500. There are no class prior changes, and IR remains 3:7. In this case, c1 is swapped with c2.

The fourth data stream ('DataDrift100%') involves a concept drift with 100% severity after time step 2500. There are no class prior changes, and IR remains 3:7. In this case, c1 is swapped with c2; c3 is swapped with c4.

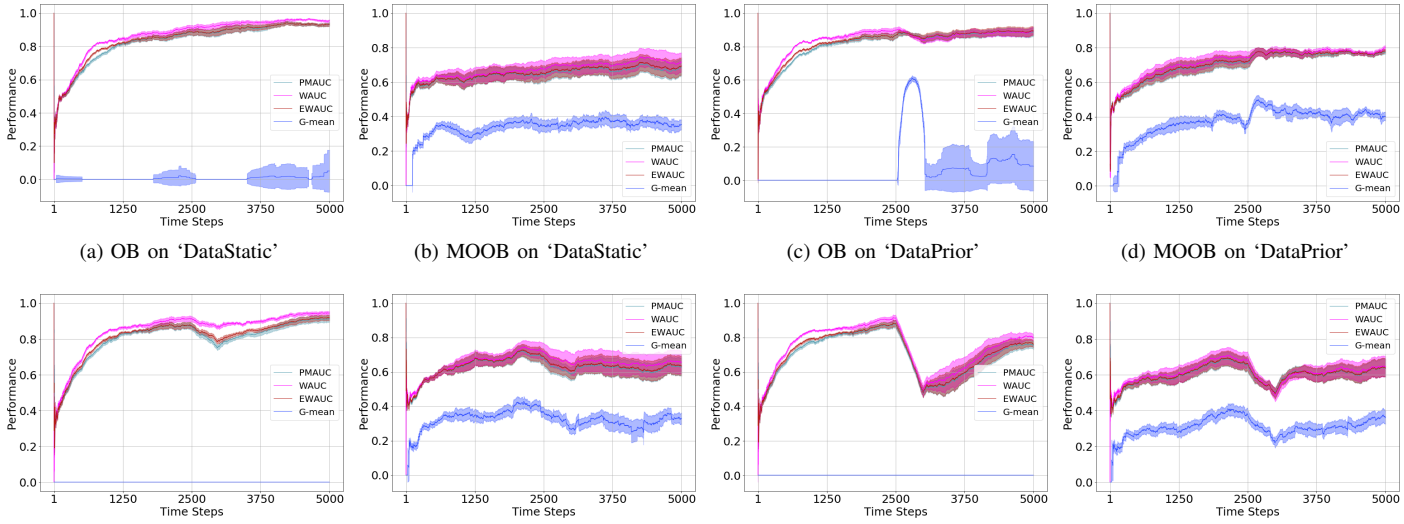
All the changes occur abruptly. 'DataDrift100%' has a more severe concept drift than 'DataDrift30%'. Severity refers to the percentage of the input space which has its target class changed after the drift is complete [2]. With this definition, because the two minority classes are affected by the concept drift in 'DataDrift30%' with 3:7 of IR, this concept drift has 30% of severity. Similarly, the concept drift in 'DataDrift100%' has 100% severity, because all the classes are replaced with a new concept.

We apply both traditional Online Bagging (OB) [26] and Multi-class Oversampling-based Online Bagging (MOOB) [8] to these data streams. OB does not tackle class imbalance, while MOOB uses adaptive random oversampling to identify minority-class examples more accurately. The purpose of including both is to gain insights into the impact of class imbalance techniques on these performance metrics. This will further help us to develop effective concept drift detection methods that are the most suitable to class imbalanced data. Both OB and MOOB are ensemble learning algorithms, training 11 multilayer perceptron (MLP) classifiers as one ensemble model. The calculation of PMAUC, WAUC, EWAUC and G-mean is based on the same window of examples for fair comparison. The window size is set to 500 in this section, which is large enough to include examples from all classes as required in [23].

B. Property Visualization and Analysis

Fig. 1 depicts how the aforementioned four performance metrics (i.e. PMAUC, WAUC, EWAUC and G-mean) produced by OB and MOOB behave over time on the four data streams. Each is presented with a mean curve and the standard deviation based on 100 runs.

By comparing across all plots, we can see that the three AUC-based metrics, i.e. PMAUC, WAUC and EWAUC, behave similarly. They are more optimistic than G-mean, in the sense that they always show a higher value, even when



(a) OB on 'DataStatic' (b) MOOB on 'DataStatic' (c) OB on 'DataPrior' (d) MOOB on 'DataPrior'

(e) OB on 'DataDrift30%' (f) MOOB on 'DataDrift30%' (g) OB on 'DataDrift100%' (h) MOOB on 'DataDrift100%'

Fig. 1: PMAUC, WAUC, EWAUC and G-mean behavior.

the model fails to recognize any examples from some of the minority classes. When OB is the learner, no class imbalance techniques are applied to rectify the imbalanced distribution, and G-mean remains zero most of the time. G-mean is the geometric mean of recall of each class, as introduced in Section II-B. A zero G-mean implies that at least one class receives zero classification accuracy. When MOOB is the learner, more minority-class examples are used for training, and G-mean is improved significantly. Based on the existing literature on class imbalance learning, a higher G-mean suggests better overall classification performance with more balanced accuracy among classes. In contrast with G-mean, the AUC-based metrics in the MOOB cases become lower than that in the corresponding OB cases. This observation tells us that the AUC values can be high when the learner performs poorly on minority classes. Therefore, G-mean can better reflect the performance on minority classes than AUC-based metrics. It also suggests the necessity of applying class imbalance techniques in order to better recognise examples from minority classes. The over optimism of AUC was also observed in Brzezinski's research in binary cases [23].

In Fig. 1(c)(d), when there is a class prior change at time step 2500, all the metrics are not negatively affected by the change. In Fig. 1(e)-(h), when there is a concept drift at time step 2500, the AUC-based metrics show a reduction in both OB and MOOB cases, indicating the performance loss caused by a concept drift. G-mean shows similar behaviour in the MOOB cases only. The impact of concept drift on G-mean cannot be observed in OB, due to the zero G-mean values. The result here suggests that, the AUC-based metrics can be over-optimistic for performance evaluation compared to G-mean, but present good sensitivity to concept drift *regardless of* whether a class imbalance technique is applied.

Let us now compare 'DataDrift30%' and 'DataDrift100%'. It is expected that the less severe concept drift (Fig. 1(e)(f)) causes a smaller performance reduction. In plot (e), WAUC does not react to concept drift as much as EWAUC and PMAUC. The reason could be that the majority class has a higher weight than the minority class in WAUC. In 'DataDrift30%', the concept drift only occurs to the minority classes. Therefore, WAUC does not show a performance drop as much as EWAUC and PMAUC. It could be problematic if the data stream becomes more class imbalanced. PMAUC and EWAUC behave closely to each other. The OVA and OVO schemes of calculating multi-class AUC do not show differences, as long as all classes are equally treated. This may suggest that EWAUC is a better AUC extension than PMAUC because of its lower computational cost.

VI. Q3: Concept Drift Detection

Concept drift may be actively detected by tracking the above performance metrics discussed in this paper. A significant drop in the metric could suggest a potential concept drift. The significance is constantly examined during learning through a statistical test, such as the popular Page-Hinkley (PH) test [11]. Existing active detectors are either designed for binary data or limited to balanced data. Therefore, in this section, we make the first step of exploring how to accurately and timely detect concept drift in multi-class imbalanced data streams.

A. Experimental Design

In the following experiment, we trace each of the four multi-class metrics, PMAUC, EWAUC, WAUC and G-mean, within the PH test, and discuss their detection performance on a set of artificial and real-world data problems. These four drift detectors are denoted by PMAUC-PH, EWAUC-PH, WAUC-PH and GM-PH. They are combined within MOOB online learner as follows. When the detector reports a concept drift, the MOOB learner is reset and starts learning from scratch; all

the window-based metrics are reset to 0. To avoid false drift alerts at the early stage of training, the detector is disabled until at least 30 examples have arrived. OB learner is not discussed in this section, because of its poor performance [6].

It is worth noting that detection threshold λ in the PH test is set to 50 throughout our experiment, for fair comparison and consistent results. When the performance reduction is greater than λ, a drift alert is issued. This is the most commonly used setting in practice [27] and open-source learning frameworks, such as Java MOA and Python scikit-multiflow, giving a generally good trade-off between false alarms and miss detections. Depending on the admissible false alarm rate, λ can be adjusted for each individual application.

1) Data Settings: We perform two groups of experiments on the artificial data with concept drift, followed by more discussions on a real-world problem. Using artificial data allows us to control when and how the concept drift happens, and compare the performance of the drift detectors.

For the first group of experiments on the artificial data (Section VI-B), we aim to find out the effectiveness of the drift detectors on different types of data, and answer: would the detectors be effective and remain effective, when the number of classes increases, when data becomes more imbalanced, and when concept drift is less severe? Which detector is most effective on different data settings? We use the same data generation method as in Section V to generate artificial data, but vary the number of classes in data (C), the severity of concept drift (s) and IR using the values shown in Table IV. There are 32 artificial data sets generated in total. The drift severity is controlled by how many classes are assigned with a new mean and covariance matrix. In this group, we fix the window size to 500 as in Section V.

TABLE IV: Artificial data settings.

Class number C = 4, 8, 16, 24			
IR	3:7	Severity s	1, 0.7, 0.5, 0.3
	1:9		1, 0.9, 0.5, 0.1

For the second group (Section VI-C), we aim to find out the impact of window size d on the performance of the drift detectors, and answer: could a smaller d lead to better sensitivity to concept drift with the risk of higher false alarm? Which detector is most effective under which window setting? We consider 4 different window settings: $d = 50, 200, 500, 1000$, and fix the data properties with IR = 3:7, C=4, and s=1.

After the in-depth analysis using the artificial data, we focus on a real-world problem (Section VI-D). It is a public gas sensor data set with potential concept drift. The task is to discriminate 6 types of gases based on data collected from 16 chemical sensors over 36 months. Eight features are extracted from each sensor, resulting in 128-dimensional feature space. Sensor drift has been recognized in this data set, which could be caused by external contamination and sensor environments [28].

2) Performance Evaluation: All the experimental analysis is based on 100 repetitions of training. We record and compare the True Detection Rate (TDR), the False Alarm Rate (FAR) and the Delay of Detection (DoD) of the concept drift detectors, when discussing the artificial cases [6]. TDR is the possibility of detecting the true concept drift. FAR is the possibility of reporting a concept drift that does not exist. DoD records the number of time steps required on average to detect a drift after the actual occurrence. They tell us how accurate and fast the detection is. A good detector should have a high TDR, a low FAR and a low DoD. A statistical test for comparing TDR, FAR and DoD from different algorithms is not applicable in our case, because their calculation relies on the outputs of all runs. Instead, we visualize and compare the TDR-FAR trade-off and average DoD through scatter plots. When studying the real-world case without knowing the ground truth, we record the total number of reported drift and observe the window-based performance metrics over time. These metrics are compared through the Wilcoxon signed-rank test at 95% confidence level.

B. Effectiveness of the Drift Detection Methods in Different Types of Data

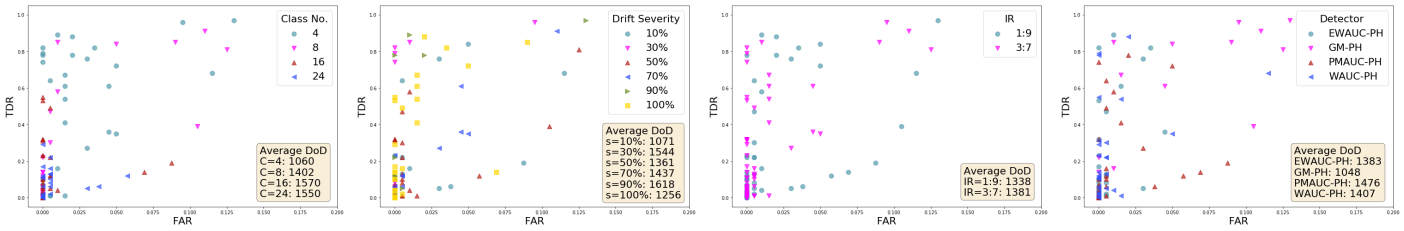
In this experiment, we discuss the performance of the 4 drift detectors on data streams with different number of classes, imbalance rate and drift severity. Fig. 2 presents the trade-off between TDR and FAR of the drift detectors on the 32 artificial data streams. Points at the top left corner are desirable. The average DoD of each group of points is also listed.

The first three plots show how the detectors are affected by the data properties. Each point is the average over all the detectors. The number of classes C presents a clear impact on the performance. Most of the $C = 4$ cases have a high TDR locating above 0.4 compared to the others; most of the $C = 16, 24$ cases locate at the bottom left corner, showing that the concept drift is less likely to be detected and thus less probability of producing a false alarm. The $C = 4$ cases also have the shortest DoD, indicating an earlier detection.

The drift severity s, surprisingly, does not present a clear impact as it ranges from 10% to 100%. A $s = 10\%$ case can have a high TDR and a low FAR, and a $s = 100\%$ can suffer from a low TDR.

The imbalance ratio does not show a significant impact on TDR, but the cases with IR=1:9 produce a higher FAR even when their TDR is low. In other words, the detected drift in more imbalanced data seems more likely to be a false alarm. It suggests that the classification performance on more imbalanced data is more fluctuant, causing the higher FAR.

In the final plot, we compare the 4 detectors across all data streams. In terms of TDR, EWAUC-PH and PMAUC-PH are competitive with GM-PH. GM-PH has good TDR (TDR $>$ 0.6), but high FAR. Among the three AUC-based detectors, PMAUC-PH tends to have more false alarms (FAR $>$ 0.05). In terms of DoD, GM-PH can detect the drift earlier than the three AUC-based detectors. Based on the observation in this experiment, EWAUC-PH is the best choice when there is limitation on false alarms; otherwise, GM-PH is also a good detector when detection promptness is necessary.



(a) Number of classes C (b) Drift severity s (c) IR (d) Detector comparison

Fig. 2: TDR, FAR and DoD on different types of data and detector. Each plot is created based on the 32 data streams.

C. Impact of Window Size on the Effectiveness of the Drift Detection Methods

The performance of each detector is presented in Table V. The results show that the window size is a key factor affecting how accurate and how soon the concept drift is detected. When the window size increases from 50 to 1000, it can be clearly observed that TDR and FAR are decreasing, and DoD is increasing for all the detectors. In other words, a small window size increases the probability of detecting the drift and reduces the delay; due to the TDR-FAR trade-off, it is more likely to be a false alarm. Therefore, it is important to find the best trade-off point for the problem and for the chosen drift detector. In this experiment, GM-PH can reach 100% TDR, but its FAR is greatly increased to 0.62; when the window size is increased to 200, although TDR is decreased to 0.87, its FAR is dropped into a more reasonable range.

TABLE V: Detection performance at the setting of "C = 4, Model = MOOB, s = 100%, IR = 3:7".

Detector	d	TDR	FAR	DoD
PMAUC-PH	50	0.65	0.04	1206
	200	0.53	0.03	1172
	500	0.41	0.015	1246
	1000	0.35	0	1662
EWAUC-PH	50	0.8	0.082	1057
	200	0.69	0.04	1065
	500	0.61	0.015	1201
	1000	0.52	0	1518
WAUC-PH	50	0.63	0.035	955
	200	0.58	0.03	960
	500	0.54	0.015	1104
	1000	0.52	0	1446
GM-PH	50	1	0.622	618
	200	0.87	0.055	813
	500	0.67	0.015	1121
	1000	0.45	0	1566

D. On Real-World Data

We apply the same online learner combined with the drift detector on the gas sensor data set. The chosen data batches cover 8 months of collected data with 5289 examples. The number of examples of each type of gases per month is shown in Table VI [28]. It presents a skewed class distribution varying over time.

Table VII shows the average number of detected drift over 100 runs, and compares all approaches against GM-PH based on 4 performance metrics through Wilcoxon signed-rank test.

TABLE VI: Number of examples of each class per month.

Month	Gas1	Gas2	Gas3	Gas4	Gas5	Gas6	Total
1	76	0	0	88	84	0	248
2	7	30	70	10	6	74	197
3	0	0	7	140	70	0	217
4	0	4	0	170	82	5	261
8	0	0	0	20	0	0	20
9	0	0	0	4	11	0	15
10	100	105	525	0	1	0	731
36	600	600	600	600	600	600	3600

Each cell in the table includes the mean and standard deviation of the average performance during the last month period over 100 runs. P-values from the significance test are presented in brackets for comparing each of the AUC-based detectors with GM-PH in each column. The purpose is to find out how the discussed detectors benefit classification over the new data concept period. The authors in [28] who collected the gas sensor data claimed and showed that the data of the last month suffered sensor drift the most.

We can see that GM-PH does not detect any drift, because G-mean remains 0 at most of the time steps. This suggests that, if the classifier performs poorly on one or more minority classes, it affects the effectiveness of GM-PH detecting concept drift. The 3 AUC-based detectors report 1 concept drift on average, and lead to a significant performance improvement in AUC types of metrics compared to GM-PH. For example, PMAUC is improved from 0.579 when using GM-PH to 0.651 when using EWAUC-PH. This reflects the positive role of using AUC-based metrics to detect concept drift in performance improvement. Among PMAUC-PH, WAUC-PH and EWAUC-PH, although they produce similar classification performance, EWAUC-PH is preferable, because EWAUC is both insensitive to class imbalance changes and computationally efficient.

VII. Conclusions

This paper studied the multi-class extension of AUC for online class imbalance learning. We focused on three research questions: Q1. define multi-class AUC metrics applicable to online class imbalance problems; Q2. analyze their properties as an evaluation measure and a concept drift indicator; Q3. investigate if and how they can be used to detect concept drift effectively and efficiently.

TABLE VII: The number of detected drift and average PMAUC, WAUC, EWAUC and G-mean on the last month of data.

Detector	Drift No.	PMAUC	EWAUC	WAUC	G-mean
PMAUC-PH	1.35	0.668±0.053 (0.000)	0.668±0.053 (0.000)	0.658±0.059 (0.000)	0.000±0.000 (0.322)
WAUC-PH	1.13	0.671±0.058 (0.000)	0.671±0.058 (0.000)	0.664±0.063 (0.000)	0.000±0.000 (NaN)
EWAUC-PH	1.17	0.671±0.059 (0.000)	0.670±0.059 (0.000)	0.662±0.065 (0.000)	0.000±0.001 (0.322)
GM-PH	0.00	0.589±0.102	0.589±0.102	0.588±0.100	0.000±0.000

*Columns 3-6: mean ± standard deviation (p-value). P-values smaller than 0.05 indicate a significant difference w.r.t. GM-PH.

For Q1, we defined PMAUC, WAUC and EWAUC that use OVO and OVA decomposition schemes. They are calculated through the sliding window strategy for incremental update and make use of the red-black tree data structure to improve computational efficiency. For Q2, we studied their properties compared to G-mean. We visualized their differences on stationary and drifting streams with different types of changes. We find that the AUC-based metrics can be optimistic as a performance measure, and G-mean can better reflect the performance on minority classes. PMAUC and EWAUC reflect the impact of concept drift well without being affected by other types of changes. For Q3, we discussed four active concept drift detectors that use PMAUC, EWAUC, WAUC and G-mean as the drift indicator respectively within PH-test. EWAUC-PH and GM-PH show better TDR. The former has a lower false alarm rate, and the latter has an earlier detection. The window size is a key factor affecting concept drift detection. Following the experiment on the artificial data, we applied the drift detectors on a real-world data set. The AUC-based detectors brought significant classification improvement.

In summary, we made the first step towards detecting concept drift in multi-class imbalanced data streams. The results are promising. In the near future, we would like to discuss their performance on more real-world data sets. Having an adaptive window size for AUC calculation would be useful.

REFERENCES

[1] N. C. Oza and S. Russell, "Experimental comparisons of online and batch versions of bagging and boosting," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 359–364.

[2] 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.

[3] M. G. Kelly, D. J. Hand, and N. M. Adams, "The impact of changing populations on classifier performance," in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999, pp. 367–371.

[4] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in non-stationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[5] 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.

[6] S. Wang, L. L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4802 – 4821, 2018.

[7] 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.

[8] S. Wang, L. L. Minku, and X. Yao, "Dealing with multiple classes in online class imbalance learning," in *The 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*, 2016, pp. 2118–2124.

[9] D. Brzezinski and J. Stefanowski, "Prequential auc for classifier evaluation and drift detection in evolving data streams," *New Frontiers in Mining Complex Patterns*, vol. 8983, pp. 87–101, 2015.

[10] F. Provost and P. Domingos, "Well-trained pets: Improving probability estimation trees," in *CDER Working Paper, STERN School of Business, NYU*, 2000, pp. 1–24.

[11] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1, pp. 100–115, 1954.

[12] Z.-L. Zhang, X.-G. Luo, S. González, S. García, and F. Herrera, "Drcw-aseg: One-versus-one distance-based relative competence weighting with adaptive synthetic example generation for multi-class imbalanced datasets," *Neurocomputing*, vol. 285, pp. 176 – 187, 2018.

[13] L. Cerf, D. Gay, N. Selmaoui-Folcher, B. Crémilleux, and J.-F. Boulicaut, "Parameter-free classification in multi-class imbalanced data sets," *Data & Knowledge Engineering*, vol. 87, pp. 109 – 129, 2013.

[14] J. Yang, J. Zhou, Z. Zhu, X. Ma, and Z. Ji, "Iterative ensemble feature selection for multiclass classification of imbalanced microarray data," *Journal of Biological Research-Thessaloniki*, vol. 23, no. 13, pp. 1–9, 2016.

[15] K. Chen, Bao-Liang Lu, and J. T. Kwok, "Efficient classification of multi-label and imbalanced data using min-max modular classifiers," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, 2006, pp. 1770–1775.

[16] H. Yu, C. Sun, X. Yang, W. Yang, J. Shen, and Y. Qi, "ODOC-ELM: Optimal decision outputs compensation-based extreme learning machine for classifying imbalanced data," *Knowledge-Based Systems*, vol. 92, pp. 55 – 70, 2016.

[17] M. Lango and J. Stefanowski, "Multi-class and feature selection extensions of roughly balanced bagging for imbalanced data," *Journal of Intelligent Information Systems*, vol. 50, no. 1, pp. 97–127, Feb 2018.

[18] M. Kubat, R. Holte, and S. Matwin, "Learning when negative examples abound," in *European Conference on Machine Learning*, vol. 1224, 1997, pp. 146–153.

[19] Y. Sun, M. S. Kamel, and Y. Wang, "Boosting for learning multiple classes with imbalanced class distribution," in *Sixth International Conference on Data Mining (ICDM'06)*, Dec 2006, pp. 592–602.

[20] C. Ferri, J. Hernández-Orallo, and M. A. Salido, "Volume under the roc surface for multi-class problems," in *Machine Learning: ECML 2003*, 2003, pp. 108–120.

[21] T. C. W. Landgrebe and R. P. W. Duin, "Efficient multiclass roc approximation by decomposition via confusion matrix perturbation analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 810–822, May 2008.

[22] D. J. Hand and R. J. Till, "A simple generalisation of the area under the roc curve for multiple class classification problems," *Machine Learning*, vol. 45, no. 2, pp. 171–186, 2001.

[23] D. Brzezinski and J. Stefanowski, "Prequential AUC: properties of the area under the roc curve for data streams with concept drift," *Knowledge and Information Systems*, vol. 52, no. 2, p. 531–562, 2017.

[24] R. Bayer, "Symmetric binary b-trees: Data structure and maintenance algorithms," *Acta Informatica*, vol. 1, no. 4, pp. 290–306, 1972.

[25] F. Provost and P. Domingos, "Tree induction for probability-based ranking," *Machine Learning*, vol. 52, no. 3, pp. 199–215, 2003.

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

[27] T. Djukic, G. Mackinnon, R. Mena, P. Krishnakumari, O. Cats, and M. Brackstone, "Seta deliverable 4.3: Initial evaluation of predictors for smart mobility," pp. 1–57, 2017.

[28] A. Vergara, S. Vembu, T. Ayhan, M. A. Ryan, M. L. Homer, and R. Huerta, "Chemical gas sensor drift compensation using classifier ensembles," *Sensors and Actuators B: Chemical*, vol. 166-167, pp. 320 – 329, 2012.