Prequential AUC for Classifier Evaluation and Drift Detection in Evolving Data Streams

Dariusz Brzezinski and Jerzy Stefanowski

Institute of Computing Science, Poznan University of Technology,
ul. Piotrowo 2, 60–965 Poznan, Poland
{dariusz.brzezinski,jerzy.stefanowski}@cs.put.poznan.pl

Abstract. Detecting and adapting to concept drift makes learning data stream classifiers a difficult task. It becomes even more complex when the distribution of classes in the stream becomes imbalanced. Currently, proper assessment of classifiers for such data is still a challenge, as existing evaluation measures either do not take into account class imbalance or are unable to indicate class ratio changes in time. In this paper, we advocate the use of the area under the ROC curve (AUC) in imbalanced data stream settings and propose an incremental algorithm that uses a sorted tree structure with a sliding window to compute AUC using constant time and memory. Additionally, we experimentally verify that this algorithm is capable of correctly evaluating classifiers on imbalanced streams and can be used as a basis for detecting sudden changes in class definitions and imbalance ratio.

Keywords: AUC, data stream, class imbalance, concept drift

1 Introduction

Many modern information system, e.g. concerning sensor networks, recommender systems, or traffic monitoring, record and process huge amounts of data. However, the massive size and complexity of the collected datasets make the discovery of patterns hidden in the data a difficult task. Such limitations are particularly visible when mining data in the form of transient *data streams*. Stream processing imposes hard requirements concerning limited amount of memory and small processing time, as well as the need of reacting to *concept drifts*, i.e., changes in distributions and definitions of target classes over time. For supervised classification, these requirements mean that newly proposed classifiers should not only accurately predict class labels of incoming examples, but also adapt to concept drifts while satisfying computational restrictions.

Classification becomes even more difficult if the data complexities also include *class imbalance*. It is an obstacle even for learning from static data, as classifiers are biased toward the majority classes and tend to misclassify minority class examples. However, it has been also shown that class imbalance ratio is usually not the only factor that impedes learning. Experimental studies [1, 2] suggest that when additional data complexities occur together with class imbalance, the

deterioration of classification performance is amplified and affects mostly the minority class. In this paper, we focus our attention on the complexity resulting from the combination of class imbalance, stream processing, and concept-drift.

Although for static imbalanced data several specialized learning techniques have recently been introduced [3, 4], similar research in the context of data streams is limited to a few papers [5–8]. However, these studies show that evolving and imbalanced data streams are particularly demanding learning scenarios, and the problem of effectively evaluating a classifier is vitally important for such data.

Currently, the performance of data stream classifiers is commonly measured with predictive accuracy (or respective error), which is usually calculated in a cumulative way over all incoming examples or at selected points in time when examples are processed in blocks. However, when values of these measures are averaged over an entire stream, they loose information about the classifier's reactions to drifts. Even recent proposals including a *prequential* way of calculating accuracy [9] or using the Kappa statistic [10, 11] are not sufficient as they are unable to depict changes in class distribution, which could appear in different moments of evolving data streams. Moreover, when the ratio of positive to negative instances changes in a test set, a classifier chosen using these metrics may no longer perform sufficiently good, or even acceptably [12].

For static imbalanced problems, a popular alternative to accuracy is the area under the ROC (Receiver Operator Characteristic) curve (AUC). An important property of AUC is that it is invariant to changes in class distribution. Moreover, for scoring classifiers it has a very useful statistical interpretation as the expectation that a randomly drawn positive example receives a higher score than a random negative example. Thus, it measures the ranking ability of classifiers, which is especially desirable if one wants to dynamically change the classification threshold in response to changing class or cost distributions [12]. Finally, several authors have shown that AUC is more preferable for model evaluation than total accuracy [13].

However, in order to calculate AUC, one needs to sort a given dataset and iterate through each example. Because the sorted order of examples defines the resulting value, adding an example to the dataset forces the procedure to be repeated. Therefore, AUC cannot be directly computed on data streams, as this would require $O(n)$ time and memory at each time point, where n is the current length of the data stream (if previously sorted scores are preserved, one only needs to insert a new score and linearly scan through the examples to calculate AUC). Up till now, the use of AUC for data streams has been limited to estimations on periodical holdout sets [8, 6] or entire streams [5, 7], making it either potentially biased or computationally infeasible.

In this paper, we propose a new approach for calculating AUC incrementally with limited time and memory requirements. The proposed algorithm incorporates a sorted tree structure with a sliding window as a forgetting mechanism, making it both computationally feasible and appropriate for concept-drifting streams. According to our best knowledge, such an approach has not been con-

sidered in the literature. Furthermore, we argue that, compared to standard accuracy, the analysis of changes of prequential AUC over time could provide more information about the performance of classifiers with respect to different types of drifts, in particular for streams with evolving class imbalance ratio. To verify this hypothesis, we carry out experiments with several synthetic and real datasets representing scenarios involving different types of drift, including sudden changes in the class imbalance ratio.

The remainder of the paper is organized as follows. Section 2 presents related work. In Section 3, we introduce an algorithm for calculating prequential AUC and investigate its properties, while Section 4 shows how prequential AUC can be used for concept drift detection. In Section 5, we present experimental results on real and synthetic datasets, which demonstrate the properties of the proposed algorithms. Finally, in Section 6 we draw conclusions and discuss future research.

2 Evaluating Data Stream Classifiers

In data stream mining, predictive abilities of a classifier are evaluated by using a holdout test set, chunks of examples, or incrementally after each example [14]. More recently, Gama et al. [9] proposed prequential accuracy with forgetting as a means of evaluating data stream classifiers and enhancing drift detection methods. They have shown that computing accuracy only over the most recent examples, instead of the entire stream, is more appropriate for continuous assessment and drift detection in evolving data streams. Nevertheless, prequential accuracy inherits the weaknesses of traditional accuracy, that is, variance with respect to class distribution and promoting majority class predictions.

For imbalanced data streams, Bifet and Frank [10] proposed the use of the Kappa statistic with a sliding window. Furthermore, this metric has been recently extended to take into account temporal dependence [11]. However, the Kappa statistic requires a baseline classifier, which is dependent of the current class imbalance ratio. Furthermore, in contrast to accuracy, the Kappa statistic is a relative measure without a probabilistic interpretation, meaning that its value alone does not directly state whether a classifier will predict accurately enough in a given setting, only that it performs better than general baselines.

The AUC measure has also been used for imbalanced data streams, however, in a limited way. Some researchers chose to calculate AUC using entire streams [5, 7], while others used periodical holdout sets [8, 6]. Nevertheless, it was noticed that periodical holdout sets may not fully capture the temporal dimension of the data, whereas evaluation using entire streams is neither feasible for large datasets nor suitable for drift detection. It is also worth mentioning that an algorithm for computing AUC incrementally has also been proposed [15], yet one which calculates AUC from all available examples and is not applicable to evolving data streams. Although the cited works show that AUC is recognized as a measure which should be used to evaluate classifiers for imbalanced data streams, up till now it has been computed the same way as for static data. In the following sections, we propose a simple and efficient algorithm for calculating

AUC incrementally with forgetting, and investigate its properties with respect to classifier evaluation and drift detection in evolving data streams.

3 Prequential AUC

Our main interest in this paper is to evaluate data stream classifiers for evolving imbalanced data streams. For this purpose, we advocate the use of the area under the receiver operator characteristic curve (AUC). Therefore, we will consider scoring classifiers, i.e., classifiers that for each predicted class label additionally return a numeric value (score) indicating the extent to which an instance is predicted to be positive or negative. Furthermore, we will limit our analysis to binary classification. It is worth mentioning, that most classifiers can produce scores, and those that only predict class labels can be converted to scoring classifiers [12].

We propose to compute AUC incrementally using a forgetting mechanism that employs a sorted window of classification scores of the most recent examples. It is worth noting that, since the calculation of AUC requires sorting examples with respect to their classification scores, it cannot be computed on an entire stream or using fading factors without using additional memory. To efficiently maintain a sorted set of scores, we propose to use a *red-black tree* [16], which is capable of adding and removing elements in logarithmic time without any additional memory. Furthermore, a window of scores is required to identify the age of each score. With these two structures, for each incoming example a new score is inserted into the window (line 16) as well as the tree (line 11) and, if the window of examples has been exceeded, the oldest score is removed (lines 5 and 16). After the window has been updated, AUC is calculated by summing the number of positive examples occurring before each negative example (lines 20–24) and normalizing that value by all possible pairs pn (line 25), where p is the number of positives and n is the number of negatives in the window. This method of calculating AUC is equivalent to summing the area of trapezoids for each pair of sequential points in the ROC curve [12], but is more suitable for our purposes as it requires very little computation given a sorted collection of scores. Algorithm 1 lists the pseudo-code for calculating prequential AUC.

Let us now analyze the complexity of the proposed approach. For a window of size d, the time complexity of adding and removing a score to the red-black tree is $O(2 \log d)$. Additionally, the computation of AUC requires iterating through all the scores in the tree, which is an $O(d)$ operation. In summary, the computation of prequential AUC has a complexity of $O(d + 2 \log d)$ per example and since d is a user-defined constant this resolves to a complexity of $O(1)$. It is worth noticing that if AUC only needs to be sampled every k examples (a common scenario while plotting metrics in time) lines from 19 to 25 can be executed only once per k examples. In terms of space complexity, the algorithm requires $O(2d)$ memory for the red-black tree and window, which also resolves to $O(1)$.

In contrast to error-rate performance metrics, such as accuracy [9, 14] or the Kappa statistic [10, 11], the proposed measure is invariant of the class distribu-

Algorithm 1 Prequential AUC

Input: $\mathcal{S}$: stream of examples, d: window size
Output: $\hat{\theta}$: prequential AUC after each example

$W \leftarrow \emptyset$; $n \leftarrow 0$; $p \leftarrow 0$; $idx \leftarrow 0$;
for all scored examples $\mathbf{x}^t \in \mathcal{S}$ **do**
  `// Remove oldest score from the window`
  if $idx \geq d$ **then**
    $scoreTree.remove(W[idx \textbf{ mod } d])$;
    if $isPositive(W[idx \textbf{ mod } d])$ **then**
      $p \leftarrow p - 1$;
    else
      $n \leftarrow n - 1$;
  `// Add new score to the window`
  $scoreTree.add(\mathbf{x}^t)$;
  if $isPositive(\mathbf{x}^t)$ **then**
    $p \leftarrow p + 1$;
  else
    $n \leftarrow n + 1$;
  $W[idx \textbf{ mod } d] \leftarrow \mathbf{x}^t$;
  $idx \leftarrow idx + 1$;
  `// Calculate AUC`
  $AUC \leftarrow 0$; $c \leftarrow 0$;
  for all consecutive scored examples $s \in scoreTree$ **do**
    if $isPositive(s)$ **then**
      $c \leftarrow c + 1$;
    else
      $AUC \leftarrow AUC + c$;
  $\hat{\theta} \leftarrow \frac{AUC}{pn}$;

tion. Furthermore, unlike accuracy it does not promote majority class predictions. Additionally, in contrast to the Kappa statistic, AUC is a non-relative, $[0, 1]$ normalized metric with a direct statistical interpretation. As opposed to previous applications of AUC to data streams [5–8], the proposed algorithm can be executed after each example using constant time and memory. Finally, compared to the method presented in [15], the proposed algorithm provides a forgetting mechanism and uses a sorting structure, making it suitable for evolving data streams and allowing for efficient sampling.

4 Drift Detection Using AUC

Prequential AUC assesses the ranking abilities of a classifier and is invariant of the class distribution. These properties differentiate it from common evaluation metrics for data stream classifiers and could be applied in an additional context. In particular, for streams with high class imbalance ratios simple metrics, such as accuracy, will suggest good performance (as they are biased toward recognizing the majority class) and may poorly exhibit concept drifts. Therefore, we propose

to investigate AUC not only as an evaluation measure, but also as basis for drift detection in imbalanced streams, where it should better indicate changes concerning the minority class.

For this purpose, we propose to modify the Page-Hinkley (PH) test [9], however, generally other drift detection methods could also have been adapted. The PH test considers a variable m^t, which measures the accumulated difference between the observed values e (originally error estimates) and their mean till the current moment, decreased by a user-defined magnitude of allowed changes δ: $m^t = \sum_{i=1}^{t} (e^t - \bar{e}^t - \delta)$. After each observation e^t, the test checks whether the difference between the current m^t and the smallest value up to this moment $\min(m^i, i = 1, \ldots, t)$ is greater than a given threshold λ. If the difference exceeds λ, a drift is signaled. In this paper, we propose to use the area *over* the ROC curve ($1 - AUC$) as the observed value. Hence, according to the statistical interpretation of AUC, instead of error estimates, we monitor the estimate of the probability that a randomly chosen positive is ranked *after* a randomly chosen negative. This way, the PH test will trigger whenever a classifier begins to make severe ranking errors regardless of the class imbalance ratio.

The aim of using prequential AUC as an evaluation measure is to provide accurate classifier assessment and drift detection for evolving imbalanced streams. In the following section, we examine the characteristics of the proposed metric in scenarios involving different types of drifts and imbalance ratios.

5 Experiments

We performed two groups of experiments, one showcasing the properties of prequential AUC as an evaluation metric, and another assessing its effectiveness as a basis for drift detection. In the first group, we tested five different classifiers [14, 17]: Naive Bayes (NB), Very Fast Decision Tree with Naive Bayes leaves (VFDT), Dynamic Weighted Majority (DWM), Online Bagging with an ADWIN drift detector (Bag), and Online Accuracy Updated Ensemble (OAUE). Naive Bayes and VFDT were chosen as incremental algorithms without any forgetting mechanism, Online Bagging was chosen as an algorithm with a drift detector, while OAUE and DWM were selected as representatives of ensemble learners. For the second group of experiments, we only utilized VFDT with Naive Bayes leaves, similarly as was done in [9].

All the algorithms and evaluation methods were implemented in Java as part of the MOA framework [18]. The experiments were conducted on a machine equipped with a dual-core Intel i7-2640M CPU, 2.8Ghz processor and 16 GB of RAM. For all the ensemble methods (Bag, DWM, OAUE) we used 10 Very Fast Decision Trees as base learners, each with a grace period $n_{min} = 100$, split confidence $\delta = 0.01$, and tie-threshold $\tau = 0.05$ [14].

5.1 Datasets

For the first group of experiments, with prequential AUC as an evaluation metric, we used 2 real and 10 synthetic datasets[1]. The real datasets were Airlines (`Air`) and PAKDD'09 (`PAKDD`), representing a balanced and imbalanced dataset respectively. To create synthetic datasets we used three popular data stream generators from MOA: SEA (`SEA`), Hyperplane (`Hyp`), and Random RBF (`RBF`) [18]. More precisely, `SEA` was a dataset without drift, $\texttt{SEA}_x$ were datasets with a 1:x class ratio and three sudden drifts, and $\texttt{SEA}_{RC}$ contained three class ratio changes (1:1 $\rightarrow$ 1:100 $\rightarrow$ 1:10 $\rightarrow$ 1:1). Furthermore, `RBF` contained two very short changes (blips), whereas $\texttt{Hyp}_x$ were datasets with a 1:x class ratio and a slow incremental drift throughout the entire stream.

For assessing prequential AUC as a measure for monitoring drift, we created 7 synthetic datasets using the SEA (`SEA`), RBF (`RBF`), Random Tree (`RT`), and Agrawal (`Agr`) generators [18]. Each dataset tested for a single reaction (or lack of one): $\texttt{SEA}_{NoDrift}$ contained no changes, and should not trigger any drift detector; `RT` involved a sudden change after 30 k examples; $\texttt{Agr}_1$, $\texttt{Agr}_{10}$, $\texttt{Agr}_{100}$ also contained a sudden change after 30 k examples, but had a 1:1, 1:10, 1:100 class ratio, respectively; $\texttt{SEA}_{Ratio}$ included a sudden 1:1 $\rightarrow$ 1:100 ratio change after 10 k examples; $\texttt{RBF}_{Blips}$ contained two blips, which should not trigger the detector. The main characteristics of all the datasets are given in Table 1.

Table 1. Characteristic of datasets.

Dataset	#Inst	#Attrs	Class ratio	Noise	#Drifts	Drift type
`SEA`	100 k	3	1:1	10%	3	none
$\texttt{SEA}_x$	1 M	3	1:x	10%	3	sudden
$\texttt{Hyp}_x$	500 k	5	1:x	5%	1	incremental
`RBF`	1 M	20	1:1	0%	2	blips
$\texttt{SEA}_{RC}$	1 M	3	1:1/1:100/1:10	10%	4	virtual
`Air`	539 k	7	1:1	-	-	unknown
`PAKDD`	50 k	30	1:4	-	-	unknown
$\texttt{SEA}_{NoDrift}$	20 k	3	1:1	10%	1	none
$\texttt{Agr}_x$	40 k	9	1:x	1%	1	sudden
`RT`	40 k	10	1:1	0%	1	sudden
$\texttt{SEA}_{Ratio}$	40 k	3	1:1/1:100	10%	1	virtual
$\texttt{RBF}_{Blips}$	40 k	20	1:1	0%	2	blips

5.2 Results

All of the analyzed algorithms were tested in terms of accuracy and prequential AUC. In the first group of experiments, the results were obtained using the

[1] Source code, test scripts, generator parameters, and links to datasets available at: `http://www.cs.put.poznan.pl/dbrzezinski/software.php`

test-then-train procedure [14], with a sliding window of 1000 examples. Table 2 presents a comparison of average classification accuracy and prequential AUC.

Table 2. Average prequential accuracy (Acc.) and AUC (AUC).

	NB		VFDT		Bag		DWM		OAUE	
	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC
$\texttt{SEA}_N D$	0.86	0.90	0.89	0.89	0.89	0.90	0.89	0.90	0.89	0.90
$\texttt{SEA}_1$	0.84	0.88	0.85	0.87	0.89	0.88	0.89	0.88	0.89	0.88
$\texttt{SEA}_{10}$	0.84	0.74	0.87	0.73	0.89	0.74	0.89	0.74	0.89	0.74
$\texttt{SEA}_{100}$	0.89	0.54	0.89	0.54	0.90	0.54	0.90	0.54	0.90	0.54
$\texttt{Hyp}_1$	0.78	0.85	0.81	0.87	0.88	0.93	0.88	0.92	0.88	0.93
$\texttt{Hyp}_{10}$	0.88	0.80	0.89	0.74	0.91	0.81	0.91	0.76	0.91	0.82
$\texttt{Hyp}_{100}$	0.94	0.57	0.93	0.53	0.94	0.56	0.94	0.52	0.94	0.55
RBF	0.74	0.83	0.96	0.98	0.99	1.00	0.98	1.00	0.99	1.00
$\texttt{SEA}_{RC}$	0.86	0.77	0.89	0.77	0.90	0.77	0.89	0.77	0.90	0.77
Air	0.65	0.66	0.64	0.65	0.64	0.65	0.65	0.65	0.67	0.68
PAKKD	0.56	0.64	0.73	0.57	0.80	0.63	0.80	0.50	0.80	0.62

By comparing average values of the analyzed evaluation metrics, we can see that for datasets with a balanced class ratio (SEA, $\texttt{SEA}_1$, $\texttt{Hyp}_1$, RBF, Air) both measures have similar values. As we expected, for datasets with class imbalance ($\texttt{SEA}_{10}$, $\texttt{SEA}_{100}$, $\texttt{Hyp}_{10}$, $\texttt{Hyp}_{100}$, PAKKD, $\texttt{SEA}_{RC}$) accuracy does not demonstrate the difficulties the classifiers have with recognizing minority class examples. The differences between accuracy and AUC are even more visible on graphical plots depicting algorithm performance in time. Figures 1–5 present selected performance plots, which best characterize the differences between both metrics.

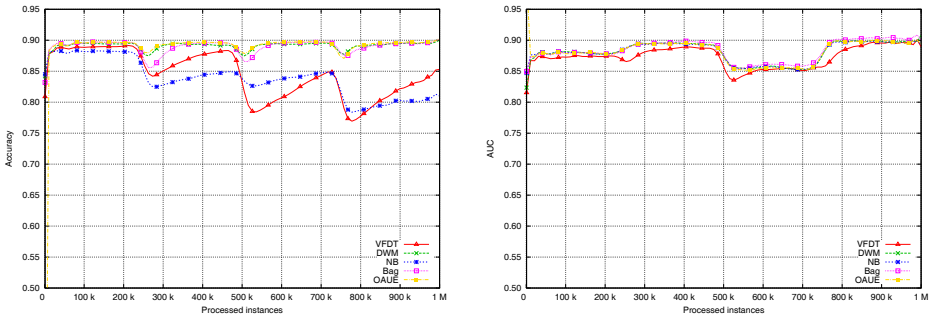

Fig. 1. Prequential accuracy (left) and AUC (right) on a data stream with sudden drifts and a balanced class ratio.

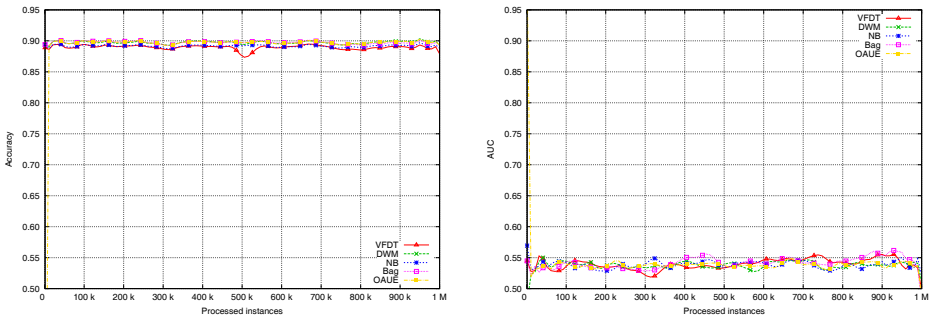

Fig. 2. Prequential accuracy (left) and AUC (right) on a data stream with sudden drifts and 1:100 class imbalance ratio.

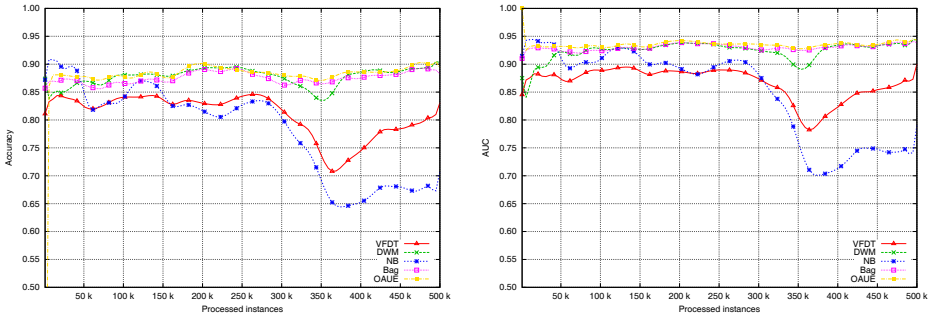

Fig. 3. Prequential accuracy (left) and AUC (right) on a data stream with incremental drift and a balanced class ratio.

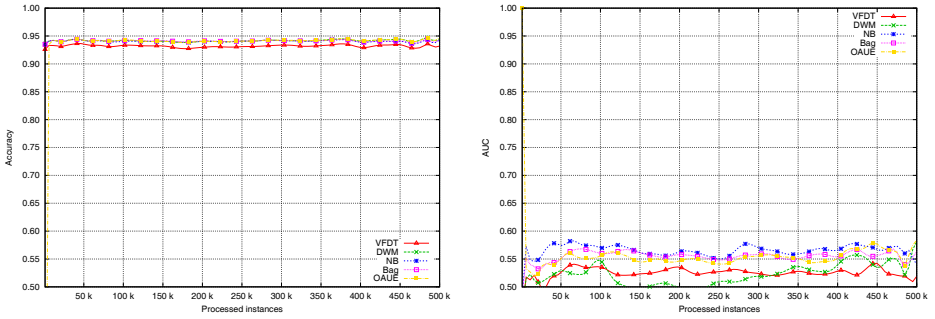

Fig. 4. Prequential accuracy (left) and AUC (right) on a data stream with incremental drift and 1:100 class imbalance ratio.

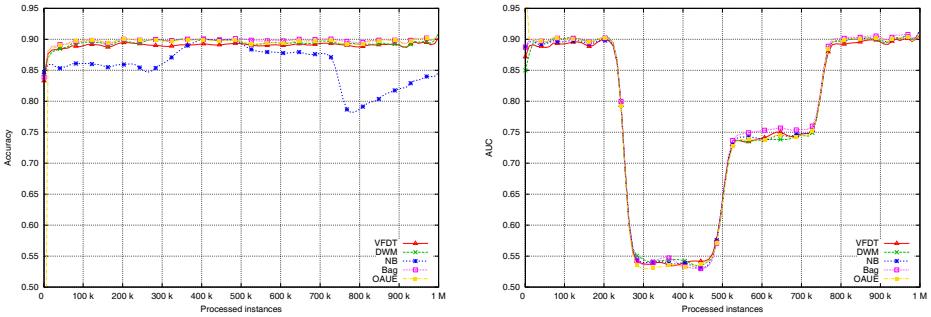

Fig. 5. Prequential accuracy (left) and AUC (right) for data with class ratio changes.

Comparing Figures 1 and 2, we can notice how the class imbalance ratio affects both prequential accuracy and AUC. The accuracy plot visibly flattens when class imbalance rises, but absolute values almost do not change. AUC on the other hand flattens but its value drastically changes, showing more clearly the classifiers' inability to recognize the minority class.

A similar situation is visible on Figures 3 and 4, where the classifiers were subject to an ongoing slow incremental drift. When classes are balanced, the plots are almost identical, both in terms of shape and absolute values. However, when the class ratio is equal 1:100, the accuracy plot flattens and its average value rises, while the AUC plot still clearly shows that classifiers are unstable and additionally its average value signals poor performance.

Finally, Figure 5 depicts classifier performance for a data stream with class ratio changes, which are sometimes called virtual drift. Apart from NB, all the tested classifiers kept the same accuracy after each drift making the changes invisible on the performance plot. However, on the AUC plot, ratio changes are clearly visible providing valuable information about the ongoing processes in the stream. In fact, the absolute values of AUC hint the severity of class imbalance in a given moment in time. This situation illustrates the advantages of prequential AUC as a measure for indicating class ratio changes.

The second group of experiments involved using the PH test to detect drifts based on changes in prequential accuracy and AUC. To compare both metrics, we used window sizes (1000–5000) and test parameters $\lambda = 100$, $\delta = 0.1$, as proposed in [9]. Table 3 presents the number of missed versus false detection counts, with average delay time for correct detections. The results refer to total counts and means over 10 runs of streams generated with different seeds.

Concerning datasets with balanced classes, both evaluation metrics provide similar drift detection rates and delays. However, for datasets with high class imbalance the PH test notes more missed detections for accuracy. This is probably due to the plot "flattening" caused by promoting majority class predictions. On the other hand, detectors which use AUC have less missed detections for highly imbalanced streams, but still suffer from a relatively high number of false alarms. This suggests that detectors using AUC should probably be parametrized dif-

Table 3. Number of missed and false detections (in the format missed:false) obtained using the PH test with accuracy (Acc) and AUC (AUC). Average delays of correct detections are given in parenthesis, where (-) means that the detector was not triggered or datasets did not contain any change. Subscripts in column names indicate the number of examples used for estimating errors.

	Acc_{1k}	Acc_{2k}	Acc_{3k}	Acc_{4k}	Acc_{5k}
$\texttt{SEA}_{NoDrift}$	0:0 (-)	0:0 (-)	0:0 (-)	0:0 (-)	0:0 (-)
$\texttt{Agr}_{1}$	0:2 (1040)	0:1 (1859)	0:0 (2843)	1:0 (4033)	5:0 (4603)
$\texttt{Agr}_{10}$	0:9 (1202)	0:3 (1228)	0:2 (1679)	0:2 (2190)	0:2 (2817)
$\texttt{Agr}_{100}$	2:12 (1610)	2:17 (2913)	2:10 (3136)	3:12 (3903)	3:10 (4612)
RT	6:0 (1843)	7:0 (2621)	8:0 (2933)	8:0 (3754)	8:0 (4695)
$\texttt{SEA}_{Ratio}$	10:0 (-)	10:0 (-)	10:0 (-)	10:0 (-)	10:0 (-)
$\texttt{RBF}_{Blips}$	0:2 (-)	0:1 (-)	0:0 (-)	0:0 (-)	0:0 (-)
	AUC_{1k}	AUC_{2k}	AUC_{3k}	AUC_{4k}	AUC_{5k}
$\texttt{SEA}_{NoDrift}$	0:0 (-)	0:0 (-)	0:0 (-)	0:0 (-)	0:0 (-)
$\texttt{Agr}_{1}$	2:2 (1042)	3:1 (1760)	4:1 (2726)	4:0 (3773)	7:0 (4640)
$\texttt{Agr}_{10}$	0:5 (868)	0:5 (1539)	0:1 (1506)	0:1 (1778)	1:1 (2197)
$\texttt{Agr}_{100}$	0:19 (1548)	0:18 (2461)	1:9 (2664)	1:11 (3563)	2:9 (4835)
RT	3:0 (1815)	5:0 (2407)	6:0 (3105)	6:0 (4121)	7:0 (4725)
$\texttt{SEA}_{Ratio}$	0:0 (1339)	0:0 (2249)	0:0 (3152)	0:0 (4057)	0:0 (4959)
$\texttt{RBF}_{Blips}$	0:3 (-)	0:1 (-)	0:0 (-)	0:0 (-)	0:0 (-)

ferently than those using accuracy. However, the most visible differences are for streams with class ratio changes. The PH test misses all virtual drifts when using accuracy as the base metric, but detects all the drifts when prequential AUC is used. This shows, that in imbalanced evolving environments the use of AUC as an evaluation measure could be of more value than standard accuracy.

6 Conclusions

In case of static data, AUC is a useful metric for evaluating classifiers both on balanced and imbalanced classes. However, up till now it has not been sufficiently popular in data stream mining, due to its costly calculation. In this paper, we introduced an efficient method for calculating AUC incrementally with forgetting on evolving data streams. The proposed algorithm, called prequential AUC, proved to be useful for visualizing classifier performance over time and as a basis for drift detection. In particular, experiments involving real and synthetic datasets have shown that prequential AUC is capable of correctly identifying poor classifier performance on imbalanced streams and detecting virtual drifts, i.e., changes in class ratio over time.

As our ongoing research, we are analyzing the possibility of using variations of AUC, such as scored AUC [12], to detect drifts more rapidly. Furthermore, we plan to analyze ROC curves plotted over time as a means of in-depth assessment of classifier performance on evolving data streams.

Acknowledgments. The authors' research was partially funded by the Polish National Science Center under Grant No. DEC-2013/11/B/ST6/00963.

References

1. Batista, G., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter **6(1)** (2004) 20–29
2. Japkowicz, N., Stephen, S.: The class imbalance problem: A systematic study. Intell. Data Anal. **6**(5) (2002) 429–449
3. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. **21**(9) (2009) 1263–1284
4. He, H., Ma, Y., eds.: Imbalanced Learning: Foundations, Algorithms, and Applications. Wiley-IEEE Press (2013)
5. Ditzler, G., Polikar, R.: Incremental learning of concept drift from streaming imbalanced data. IEEE Trans. Knowl. Data Eng. **25**(10) (2013) 2283–2301
6. Hoens, T.R., Chawla, N.V.: Learning in non-stationary environments with class imbalance. In: KDD, ACM (2012) 168–176
7. Lichtenwalter, R., Chawla, N.V.: Adaptive methods for classification in arbitrarily imbalanced and drifting data streams. In: PAKDD Workshops. Volume 5669 of Lecture Notes in Computer Science., Springer (2009) 53–75
8. Wang, B., Pineau, J.: Online ensemble learning for imbalanced data streams. CoRR **abs/1310.8004** (2013)
9. Gama, J., Sebastião, R., Rodrigues, P.P.: On evaluating stream learning algorithms. Machine Learning **90**(3) (2013) 317–346
10. Bifet, A., Frank, E.: Sentiment knowledge discovery in twitter streaming data. In: Discovery Science. Volume 6332 of Lecture Notes in Computer Science., Springer (2010) 1–15
11. Zliobaite, I., Bifet, A., Read, J., Pfahringer, B., Holmes, G.: Evaluation methods and decision theory for classification of streaming data with temporal dependence. Machine Learning (2014)
12. Wu, S., Flach, P.A., Ramirez, C.F.: An improved model selection heuristic for AUC. In: ECML. Volume 4701 of Lecture Notes in Computer Science., Springer (2007) 478–489
13. Huang, J., Ling, C.X.: Using AUC and accuracy in evaluating learning algorithms. IEEE Trans. Knowl. Data Eng. **17**(3) (2005) 299–310
14. Gama, J.: Knowledge Discovery from Data Streams. Chapman and Hall (2010)
15. Bouckaert, R.R.: Efficient AUC learning curve calculation. In: Australian Conference on Artificial Intelligence. Volume 4304 of Lecture Notes in Computer Science., Springer (2006) 181–191
16. Bayer, R.: Symmetric binary b-trees: Data structure and maintenance algorithms. Acta Inf. **1** (1972) 290–306
17. Brzezinski, D., Stefanowski, J.: Combining block-based and online methods in learning ensembles from concept drifting data streams. Information Sciences **265** (2014) 50–67
18. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive Online Analysis. J. Mach. Learn. Res. **11** (2010) 1601–1604