Learning to Classify Data Streams with Imbalanced Class Distributions

Ryan N. Lichtenwalter[1] and Nitesh V. Chawla[1]

The University of Notre Dame, Notre Dame, IN 46556, USA

Abstract. Streaming data is pervasive in a multitude of data mining applications. One fundamental problem in the task of mining streaming data is distributional drift over time. Streams may also exhibit high and varying degrees of class imbalance, which can further complicate the task. In scenarios like these, class imbalance is particularly difficult to overcome and has not been as thoroughly studied. In this paper, we consider the issue of high class imbalacne in conjunction with data streams. We propose a method called Boundary Definition, which relies on building the classifiers by stressing on the boundary cases as the streams arrive. We employ a sequential validation framework, which we believe is the most meaningful option in the context of streaming imbalanced data.

Key words: Sequential learning, stream mining, imbalanced data, skew

1 Introduction

Data stream mining is a prolific area of research. In recent years, many papers have been published either directly related to stream mining or addressing challenges in a particular stream mining application. Concept drift is a problem inherent in stream mining that continues to limit performance. Classifier ensembles have been applied in several incarnations to reduce variance, but they cannot combat bias. More recently the problem of class imbalance has been considered in the context of existing stream mining research. The intersection of these difficulties makes for a uniquely interesting and demanding research topic that can contribute to practically every data-driven or data-related field. As data streams become increasingly ubiquitous and prolific, the importance of solving their unique and interrelated challenges grows.

Research in stream mining primarily falls into one of two families: incremental processing [1–4] or batch processing [5, 6]. Figure 1 illustrates. In the former approach, each processing step handles one instance. Labels for an instance are assumed to be available immediately after classifying the instance. Instance processing methods have an inherent advantage in terms of their adaptiveness due to this assumption. At time t, instance i_t is received and classified with some model trained on instances i_1 through i_{t-1}. At time $t+1$, a label arrives for instance i_t and unlabeled instance i_{t+1} becomes available for classification. In the latter approach, each batch is a collection of data instances that arrives during a

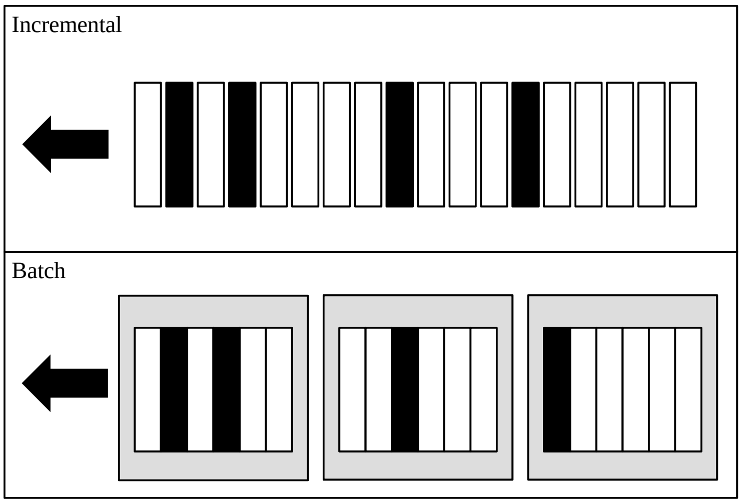

Fig. 1. An illustration of the difference in the way incremental and batch processing methods expect and handle incoming streams of data. With the arrival of each subsequent unit, class labels become available for the previous unit.

particular period. At time t, batch b_t is received and classified with some model trained on batches b_1 through b_{t-1}. At time $t+1$, labels arrive for batch b_t and unlabeled instances in batch b_{t+1} become available for classification. While the problem spaces for these two approaches do intersect, often the instance processing assumption is unrealistic. In many scenarios, large sets of new data and class labels arrive with low temporal granularity such as weeks, months, or years. The principles presented in this paper operate in the batch processing space.

1.1 Contributions

We focus on the intersection of the imbalanced data and data streams. Imbalanced data is a persistent problem in general data mining contexts. The problem becomes especially difficult in streams. Consider data with 12,000 instances and a class ratio of 100:1. This leaves 120 positive class examples. This may be sufficient to define the class boundaries and leaves plenty of data after application of a good resampling approach. In a streaming context this data may appear over the course of a year. If we assume a uniform temporal distribution of the positive class and want a good model for monthly predictions, we have only 10 positive class examples each month on which to train. This may be insufficient. In reality, some months may exhibit no positive class occurrences at all while others have hundreds. Figure 2 illustrates a simplistic example.

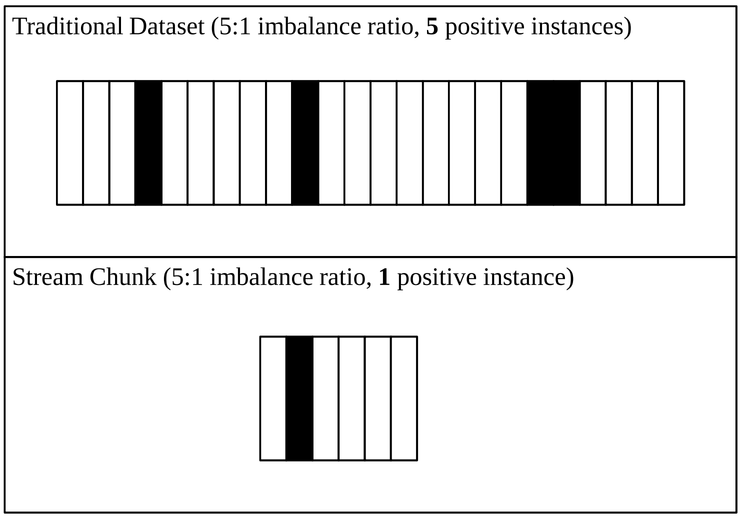
Traditional Dataset (5:1 imbalance ratio, **5** positive instances)
Stream Chunk (5:1 imbalance ratio, **1** positive instance)

Fig. 2. An illustration of the manifestation of the class imbalance problem in the stream mining domain.

We call our approach Boundary Definition as it focuses on improving the boundary discrimination between the minority (positive or rare) and majority (negative) classes across the stream.

2 Boundary Definition

We present two findings about how to resample in stream mining to achieve better performance under heavy class imbalance. The first, in line with other findings such as in [7], is that the performance improvement in propagating rare-class instances arises from defining the class boundary better rather than simply providing more positive class instances on which to train. In addition to propagating rare-class instances, we propagate instances in the negative class that the current model misclassifies.

Our method works as follows. We allow for the replication of the positive or rare class instances across the streams, but we focus more on the misclassified majority or negative class instances among all the negative or majority class instances. That is, after each stream we evaluate the predictions on both positive and negative class cases. We keep the positive class instance examples to ensure the desired percent positive. To achieve this we undersample the negative class instances. The negative class instances are first randomly undersampled from the set of correctly classified negative class isntances. Once we run out of that pool, then we undersample from the misclassified negative class instances. This is done to ensure that the class proportion remains at the desired level. The end effect of this method is that it effectively reduces the false positive rate, which is desired.

Algorithm 1 shows the pseudo-code.

Algorithm 1 Misclassified Propagation

```
Require: unlabeled batch of instances $\boldsymbol{b}$
    model $m$
    real class labels $l$
    desired percent positive $p$
    set of positive examples $\boldsymbol{P}$
    set of misclassified negative examples $\boldsymbol{N}$
Ensure: resampled batch $\boldsymbol{b}$
    updated set of positive examples $\boldsymbol{P}$
    updated set of misclassified negative examples $\boldsymbol{N}$

 1: for $instance \in \boldsymbol{b}$ do
 2:   $label \leftarrow$PREDICT$(m, instance)$
 3:   if $l_{instance} = +$ then
 4:     $\boldsymbol{pos} \leftarrow \boldsymbol{pos} \cup instance$
 5:   else
 6:     $\boldsymbol{neg} \leftarrow \boldsymbol{neg} \cup instance$
 7:     if $l_{instance} \neq label$ then
 8:       $\boldsymbol{mneg} \leftarrow \boldsymbol{mneg} \cup instance$
 9:     end if
10:   end if
11: end for
12: while $|\boldsymbol{pos}| < p * (|\boldsymbol{pos}| + |\boldsymbol{neg}| + |\boldsymbol{mneg}|)$ do
13:   if $|\boldsymbol{neg}| > 0$ then
14:     REMOVE-RANDOM$(\boldsymbol{neg})$
15:   else
16:     REMOVE-RANDOM$(\boldsymbol{mneg})$
17:   end if
18: end while
19: $\boldsymbol{b} \leftarrow \boldsymbol{pos} \cup \boldsymbol{neg} \cup \boldsymbol{mneg}$
20: return $\boldsymbol{b}, \boldsymbol{P}, \boldsymbol{N}$
```

3 Evaluation Methods

There are two fundamental methods of evaluating classification performance in data streams under the batch paradigm. The first is to evaluate the classifier over a single batch among the many batches of data. The second is to evaluate the classifier over all of the batches in the data stream and either examine them individually or use some notion of average performance. We highlight the core ideas underlying these two fundamental methods.

Evaluating the classifier over a single batch of data has been used in [5], where the last batch of data is used to render performance scores. Such an evaluation framework might somehow seek to report a concept of average classification accuracy by fulfilling some combination of these three goals:

1. Reporting results on a batch of data representative of the stream as a whole.
2. Reporting results on a batch of data for which classifier performance is representative of some notion of typical or average performance.
3. Reporting results on a batch of data sufficiently late in the stream so that the classifier has captured the concept.

The first and second may seem at first to be the same, but in fact they are not. A simple example proves this. Suppose a heavily drifting binary classification problem C with batches b_1 to b_n and class labels $\in \{0, 1\}$. Further suppose a dumb model M that always adapts to drift by classifying all instances of b_i with the predominant label from b_{i-1}. Now suppose that batches b_1 to b_{n-1} alternate such that instances $\in \{b_i | i = 2m, m \in \mathbb{Z}\}$ have label 0, instances $\in \{b_i | i = 2m + 1, m \in \mathbb{Z}\}$ have label 1, and b_n has a uniform distribution of the two classes. When M is evaluated according to the first goal, b_n is used and M is reported to obtain 50 percent classification accuracy. When M is evaluated according to the second goal, any one of b_1 through b_{n-1} is used and M is reported to obtain 0 percent classification accuracy.

The example is contrived, but nonetheless illustrates that the first two goals are not always aligned in terms of the results they output. Unfortunately, the first goal can output virtually meaningless results and the second goal may be difficult to achieve without evaluating model performance on all batches anyhow. The result is that it may be very difficult to decide on which batch one should evaluate and report performance, especially in the context of the third goal.

The third goal itself may or may not be desirable. Since one of the principle objectives in many approaches to data stream mining is to overcome various forms of drift, waiting until the model has achieved a stable state with respect to its classification performance in the stream may inflate the performance metric. Even in data streams with static distributions and concepts, using a single batch such as the last one does not include information about how quickly the target concept was learned. Two models M_1 and M_2 that reach performance level P somewhere in n batches of data certainly are not equally effective if M_1 reaches P after $\frac{n}{2}$ batches and M_2 reaches P after $\frac{n}{4}$ batches. The second model, in most circumstances should be considered clearly superior, but evaluation on the last batch will consider them the same.

All of these problems may be avoided by employing the second fundamental method. Using all batches does not suffer from the problem of finding representative data. Indeed, in data streams where concepts fluctuate through time, the existence of any meaningful sense of representative data is dubious since a single batch fundamentally lacks the ability to represent the temporal dimension. Using all batches also guarantees that not only the peak performance is captured in the evaluation, but also the speed at which that performance is reached. For these reasons, we advocate, and unless otherwise stated, employ the evaluation method that averages classification performance metrics over all batches in the data stream.

4 Datasets

We operate on different categories of data sets with drastically different properties. Chief among these in number are UCI data sets [8]. They are not inherently sequential and exhibit no concept drift, but we consider these data sets because of their class imbalance. We render them as data streams by randomizing the order of instances and processing them in batches. Most notably, we use `thyroid`, `covtype`, `optdigits`, and `letter-recognition` for direct comparison with [5].

We also present several generated and real world data sets that vary greatly in their degrees of imbalance and distributional drift. The `can` data set captures properties of a crunching can over time and exhibits imbalance at a ratio of 52:1 and extreme distributional drift in the form of blips of positive class instances. `compustat` includes both class imbalance and concept drift, as the training and testing sets represent several years of changing data. The `football` data set comprises features derived from 2003-2008 college football statistics available on ESPN and the class is whether the home team wins. Features drift over time due to underlying changes, the most significant of which are clock rule changes that affect game time. We use a version of the `kddcup2008` data set, which contains instances of breast cancer and exhibits extreme imbalance. `text` is a text-recognition data set with a large number of features. Finally, we include an ordered variation of the Wharton `wrds` data set.

Characteristics of all of these data sets are available in Table 1. Lack of public availability of source data is a problem with many works focusing on streaming data. It is therefore difficult to repeat or verify experiments. For this reason, all data sets for experiments in this paper are publicly available.

5 Results

We first examine the degree to which the boundary definition (BD) method can improve classification metrics on static data sets. Because this method seeks only to improve performance in the domain of extreme imbalance, we compare it to the other existing work [5], from which it borrows the idea of propagating minority class instances. In Tables 2(a), 2(b), and 2(c) we directly compare the performance for several data sets reported in [5]. We duplicated their methods (SE) for constructing the data sets and evaluating performance as closely as possible. For `letter` we created a separate data for each individual letter with that letter as the positive class. For `optdigits` we followed the same process for each number. We select class 2 as the negative class and class 4 as the positive class in `covtype`. From `thyroid` we create two data sets with class 3 as the negative class: one with 1 as the positive class and the other with 2 as the positive class. After constructing the data sets from the UCI source data, we randomized

Table 1. Data Set Characteristics

	Instances		Features		Properties	
Name	All	Positive	Nominal	Numeric	Imbalance	Ordered
`adult`	48,842	11,687	8	6	3.2:1	No
`boundary`	3,505	123	175	0	27.5:1	No
`breast-w`	569	212	0	30	1.7:1	No
`cam`	18,916	942	132	0	19.1:1	No
`can`	443,872	8,360	0	9	52.1:1	Yes
`compustat`	13,657	520	0	20	25.3:1	Yes
`covtype`	38,500	2,747	0	10	13.0:1	No
`estate`	5,322	636	0	12	7.4:1	No
`football`	4,288	1,597	2	11	1.7:1	Yes
`fourclass`	862	307	0	2	1.8:1	No
`german`	1,000	300	0	24	2.3:1	No
`ism`	11,180	260	0	6	42.0:1	No
`kddcup2008`	102,294	623	2	123	163.2:1	No
`letter`	20,000	789	0	16	24.3:1	No
`oil`	937	41	0	49	21.9:1	No
`ozone-1h`	2,536	73	0	72	33.7:1	Yes
`ozone-8h`	2,534	160	0	72	14.8:1	Yes
`page`	5,473	560	0	10	8.8:1	No
`pendigits`	10,992	1,142	0	16	8.6:1	No
`phoneme`	5,400	1,584	0	5	2.4:1	No
`phoss`	11,411	613	0	480	17.6:1	No
`pima`	768	268	0	8	1.9:1	No
`satimage`	6,430	625	0	36	9.3:1	No
`segment`	2,310	330	0	19	6.0:1	No
`splice`	1,000	483	0	60	1.1:1	No
`STAGGER`	12,000	5,303	3	0	1.3:1	Yes
`svmguide1`	3,089	1,089	0	4	1.8:1	No
`text`	11,162	709	0	11,465	14.7:1	Yes
`thyroid`	7,200	166	15	6	42.4:1	No
`wrds`	99,200	48,832	2	39	1.0:1	Yes

the order of the instances and repeated all framework experiments ten times. The baseline method (BL) is to train a model on batch t with no resampling and use it to test on batch $t+1$. For instructive purposes, we also examine the performance of positive class propagation when only the misclassified instances are propagated (SE-). We report several widely accepted imbalanced classification metrics including the area under the receiver operating characteristic curve (AUC-ROC) as implemented in WEKA [9], precision, recall, F_1-measure, and the area under the precision-recall curve (AUC-PR) as implemented in [10]. We observe that AUC-PR is a more discriminating measure than AUC-ROC in the domain of imbalance. Bold font indicates the highest value, but not necessarily statistical significance.

Table 2. Direct comparison of BD and SE.

(a) Precision and Recall

	Precision				Recall			
	BL	SE-	SE	BD	BL	SE-	SE	BD
`covtype`	0.980	0.603	0.770	**0.993**	0.958	**0.999**	0.998	**0.999**
`letter`	**0.836**	0.398	0.519	0.780	0.719	**0.963**	0.955	0.934
`optdigits`	0.869	0.765	0.834	**0.915**	0.827	**0.954**	0.952	0.952
`thyroid1`	**0.914**	0.527	0.728	0.897	0.907	**1.000**	**1.000**	**1.000**
`thyroid2`	0.940	0.617	0.857	**0.957**	0.982	**1.000**	**1.000**	**1.000**

(b) F_1-measure and PRBEP

	F_1-measure				PRBEP			
	BL	SE-	SE	BD	BL	SE-	SE	BD
`covtype`	0.969	0.748	0.869	**0.996**	0.968	0.880	0.953	**0.997**
`letter`	0.771	0.558	0.667	**0.847**	0.739	0.775	0.826	**0.891**
`optdigits`	0.846	0.847	0.887	**0.932**	0.833	0.896	0.915	**0.936**
`thyroid1`	0.910	0.686	0.836	**0.945**	0.899	0.876	0.912	**0.937**
`thyroid2`	0.960	0.762	0.922	**0.978**	0.941	0.860	0.922	**0.975**

(c) AUC-ROC and AUC-PR

	AUC-ROC				AUC-PR			
	BL	SE-	SE	BD	BL	SE-	SE	BD
`covtype`	0.975	0.999	**1.000**	**1.000**	0.944	0.840	0.951	**1.000**
`letter`	0.891	0.987	0.989	**0.993**	0.675	0.785	0.847	**0.941**
`optdigits`	0.909	0.990	0.990	**0.993**	0.768	0.934	0.949	**0.972**
`thyroid1`	0.951	0.998	**0.999**	**0.999**	0.854	0.897	0.913	**0.934**
`thyroid2`	0.991	0.995	0.998	**0.999**	0.934	0.862	0.920	**0.975**

From Table 2(a) we see immediately that propagating misclassified negative class instances improves precision. The method creates a more complete boundary for training. In all cases, it is much closer to baseline precision, while sacrificing very little in recall. In some cases, it even improves precision beyond the baseline, which uses no resampling. Table 2(b) illustrates the trade-off performance on precision and recall and we see here that BD outperforms SE in terms of F_1-measure by at least 5 percent and as much as 26 percent. Finally, BD always performs at least as well as SE in AUC-ROC while outperforming it for every data set in the more discriminating AUC-PR measure.

For a broader view of the performance of BD with respect to the baseline and with respect to SE, we provide comprehensive coverage of 21 data sets from the UCI repository. To obtain these results, we simply execute each framework ten times on each data set. The data sets have precisely the properties described in 1. Each cell intersecting a method and a data set represents the rank of the method on that data set for the metric in the heading. A rank of 1 means the method performs best, and a rank of three means it performs worst. At the bottom, we provide the mean rank and the overall rank. The overall rank is calculated using the Nemenyi procedure as described in [11] with $\alpha = 0.05$. Statistical significance is indicated between two ranks when a value of one or greater separates them. This comes into play for the F_1-measure column where BD performs significantly better than BL, but SE does not perform statistically significantly better than BL, and BD does not perform statistically significantly better than SE.

Table 3. Performance on Static Data Sets

	Precision			Recall			F_1-measure			AUC-ROC			AUC-PR		
Name	BL	SE	BD	BL	SE	BD	BL	SE	BD	BL	SE	BD	BL	SE	BD
`adult`	1	3	2	3	1	2	2	3	1	3	2	1	3	2	1
`boundary`	3	2	1	3	1	2	3	2	1	3	2	1	3	2	1
`breast-w`	2	3	1	3	2	1	3	2	1	3	2	1	3	2	1
`cam`	1	2	3	3	1	2	3	1	2	3	1	2	3	1	2
`covtype`	2	3	1	3	1	2	3	2	1	3	2	1	3	2	1
`estate`	3	2	1	3	1	2	3	1	2	3	1	2	3	1	2
`fourclass`	3	2	1	3	1	2	3	2	1	3	2	1	3	2	1
`german`	1	2	3	3	1	2	3	1	2	3	1	2	3	2	1
`ism`	1	2	3	3	1	2	1	2	3	3	2	1	2	3	1
`letter`	1	3	2	3	2	1	1	3	2	3	2	1	3	2	1
`oil`	1	3	2	3	1	2	1	2	3	3	1	2	3	2	1
`page`	1	2	3	3	1	2	1	3	2	3	2	1	3	2	1
`pendigits`	1	3	2	3	1	2	2	3	1	3	2	1	3	2	1
`phoneme`	2	3	1	3	1	2	3	2	1	3	2	1	3	2	1
`phoss`	1	2	3	3	1	2	3	1	2	3	1	2	3	1	2
`pima`	1	3	2	3	1	2	3	1	2	3	2	1	3	2	1
`satimage`	1	3	2	3	1	2	3	2	1	3	2	1	3	2	1
`segment`	1	3	2	3	1	2	2	3	1	3	2	1	2	3	1
`splice`	3	2	1	3	2	1	3	2	1	3	2	1	3	2	1
`svmguide1`	1	3	2	3	1	2	3	1	2	3	1	2	3	1	2
`thyroid`	1	3	2	3	1	2	1	3	2	3	2	1	3	2	1
MEAN	1.5	2.6	1.9	3.0	1.1	1.9	2.4	2.0	1.6	3.0	1.7	1.3	2.9	1.9	1.2
OVERALL	1	2.5	1.5	3	1	2	2.5	2	1.5	2.5	2	1.5	2.5	2	1.5

We can conclude that both BL and BD achieve better precision than SE, which loses precision with respect to BL because it tends to push the border beyond more negative class instances. Although BL usually achieves higher precision than BD, the results are not significant with $\alpha = 0.05$. Although we see that SE outranks BD in terms of recall, a quick look at both Table 2(a) and much of the data underlying the ranks show that the benefit of SE over BD with respect to BL is often negligible given the precision gains. Although individual F_1-measure results suggest that BD often outperforms SE, we cannot say this with $\alpha = 0.05$, nor can we say with confidence that SE outperforms BL. Finally, on both AUC-ROC and AUC-PR, despite frequent wins by BD, the results are not statistically significant at $\alpha = 0.05$. For AUC-PR, at $\alpha = 0.10$ we can say that SE performs worse than BD.

6 Related Work

Researchers have extensively studied class imbalance. The most popular approaches for handling imbalance are various forms of resampling to achieve a more balanced distribution, including random undersampling of the negative class and SMOTE [12]. A more recent approach is an active learning system capable of efficiently querying large data sets to find informative examples [13]. [5] addresses class imbalance in the streaming scenarios. Streaming data is generally tackled by batch approaches. Many batch approaches employ some form of ensemble technique. Some use simple average [5], while others employ weighting [3, 6]. With various levels of sophistication, some systems maintain a sliding window of examples or models. [14] and [15] dynamically adjust window size and the former provides strict performance guarantees. We observe that a contiguous window, even an optimally sized one, may fail to contain recurring concepts if they were previously forgotten by the system. This concept of periodicity is briefly discussed in [1].

7 Conclusions

We implemented a streaming classification method that focuses on the boundary conditions to improve the separation between the positive and negative class instances in a streaming scenario. The problem of class imbalance is highly accentuated in the streaming data, as the streams present the challenges of small sample size, as well as the changes in class distribution based on the nature of the stream. This requires the method to be sensitive to the instances that are "hard to classify". We achieve this by especially paying attention to the misclassified negative class instances. Our results show that this enables a higher precision rate, albeit lower recall than the competitive methods. BD is able to achieve a higher F_1-measure, albeit not statistically significant. We are focusing on integrating the BD method with other sampling mechanisms to also improve the recall rate so that there is an overall improvement enabled for the F_1-measure.

References

1. Becker, H., Arias, M.: Real-time ranking with concept drift using expert advice. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM Press (2007) 86–94
2. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM (2001) 97–106
3. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: An ensemble method for drifting concepts. Journal of Machine Learning Research **8** (2007) 2755–2790
4. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning **23**(1) (April 1996) 69–101
5. Gao, J., Fan, W., Han, J., Yu, P.S.: A general framework for mining concept-drifting data streams with skewed distributions. In: SDM '07: Proceedings of the SIAM International Conference on Data Mining. (2007)
6. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM (2003) 226–235
7. Han, H., Wang, W.Y., Mao, B.H.: Borderline-smote: A new over-sampling method in imbalanced data sets learning. Advances in Intelligent Computing (2005) 878–887
8. Asuncion, A., Newman, D.: Uci machine learning repository (2007)
9. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Second edn. Morgan Kaufmann, San Francisco, California, USA (2005)
10. Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves. In: ICML '06: Proceedings of the 23rd international conference on Machine learning, New York, NY, USA, ACM (2006) 233–240
11. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research **7** (2006) 1–30
12. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, P.W.: Smote: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research **16** (May 2002) 341–378
13. Ertekin, S., Huang, J., Bottou, L., Giles, L.: Learning on the border: Active learning in imbalanced data classification. In: CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, New York, NY, USA, ACM (2007) 127–136
14. Bifet, A., Gavaldá, R.: Learning from time-changing data with adaptive windowing. In: SIAM International Conference on Data Mining (SDM07). (2006)
15. Klinkenberg, R.: Using labeled and unlabeled data to learn drifting concepts. In: Workshop notes of the IJCAI-01 Workshop on Learning from Temporal and Spatial Data. (2001) 16–24