Online Oversampling for Sparsely Labeled Imbalanced and Non-Stationary Data Streams

1st Łukasz Korycki
Department of Computer Science
Virginia Commonwealth University
Richmond VA, USA
koryckil@vcu.edu

2nd Bartosz Krawczyk
Department of Computer Science
Virginia Commonwealth University
Richmond VA, USA
bkrawczyk@vcu.edu

***Abstract*—Learning from imbalanced data and data stream mining are among most popular areas in contemporary machine learning. There is a strong interplay between these domains, as data streams are frequently characterized by skewed distributions. However, most of existing works focus on binary problems, omitting significantly more challenging multi-class imbalanced data. In this paper, we propose a novel framework for learning from multi-class imbalanced data streams that simultaneously tackles three major problems in this area: (i) changing imbalance ratios among multiple classes; (ii) concept drift; and (iii) limited access to ground truth. We use active learning combined with streaming-based oversampling that uses both information about current class ratios and classifier errors on each class to create new instances in a meaningful way. Conducted experimental study shows that our single-classifier framework is capable of outperforming state-of-the-art ensembles dedicated to multi-class imbalanced data streams in both fully supervised and sparsely labeled learning scenarios.**

***Index Terms*—active learning, data stream mining, imbalance, dynamic ratio, concept drift, oversampling, online adaptation.**

I. Introduction

Contemporary data sources continuously generate ever-growing amounts of information at high speed. This phenomenon is known as data streams and can be defined as a sequence $< \mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_n, ... >$, in which each element $\mathbf{X}_j$ is a set of instances (or a single instance in a case of online learning) independent and randomly generated using a stationary probability distribution.

Data streams evolve over time, changing their properties, due to a phenomenon known as concept drift [7]. These characteristics stimulated development of new machine learning algorithms capable of handling such continuously arriving and drifting data. Additionally, due to the potentially unbounded size of data stream it is impossible to provide ground truth for each new instance [21]. Therefore, in real-life scenarios we must deal with scarcely labeled data streams, which will very likely lead to underfitting, and use wisely the highly limited access to class labels. One of the most popular solutions is active learning that select instances for labeling, while accounting for the presence of concept drift [15]. The problem with using active learning alone is that for the highly limited budgets all the selected instances may still not be enough to ensure that dynamic boundaries will be updated effectively and quickly enough.

Another learning difficulty that gathers increasing attention from the data stream mining community is class imbalance [12]. In the streaming context it poses new challenges, especially when combined with concept drift [20]. Here, the proportions between classes change dynamically, as well as class roles – minority may become a majority over time [12]. Although there exist a plethora of solutions for imbalanced problems, adapting them to data streams and concept drift is not straightforward. Ensemble approaches offer attractive way to handle both streaming and imbalanced nature of data and have been showed to obtain excellent performance in this domain [13].

While there exist some works on handling imbalanced data streams, they usually concentrate on binary problems [9], [11], [20]. However, from the context of data streams multi-class imbalanced problems are much more interesting, as they occur when new classes emerge, old ones disappear, or break into subconcepts [17]. They pose even bigger challenge, as relationships among classes are not longer well-defined and one cannot decompose them into binary subproblems without losing valuable information [12]. There exist but few algorithms dedicated to multi-class imbalanced data streams, but they either focus on changing class ratios without drifts [18], [19], or on handling concept drift with static class ratios [14], [5]. However, in real-life problems these phenomena occur simultaneously. Additionally, existing solutions assume unrestricted access to class labels.

We propose a novel learning framework for multi-class data streams that addresses all of the mentioned challenges: (i) changing imbalance ratios among multiple classes; (ii) concept drift; and (iii) limited access to ground truth. We use active learning for selecting most valuable instances for labeling and then use them to perform the multi-class oversampling. However, we guide our selection and instance generation procedures with a hybrid criterion that takes into account both current class ratios and the classifier error on each class independently. This allows us to effectively tackle both concept drift and class imbalance. At the same time, the fact that we use the oversampling module, generating additional instances, may be a solution to the underfitting after drifts regardless of a class imbalance, so it potentially addresses the problem of the limited budget simultaneously. Experimental study shows

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

that our approach, based only on a single classifier, can provide sufficient solutions to the described problems, improving upon both standard active learning and more sophisticated ensembles, dedicated to learning from multi-class imbalanced data streams or simply performing well in general cases.

II. Proposed algorithms

A. Deceptive majority and budget constraints

The existing algorithms for multi-class imbalanced data streams with dynamic class ratios do not take into account two crucial aspects of learning from streaming sources. The first one is that while they adapt to changes in class proportions they do not provide any explicit mechanism to handle concept drifts that are common in streams and may occur concurrently with class ratio changes. Why is it important?

Let us consider a case in which we have a majority class c_1 consisting of 80% of all instances and minority c_2 with 20% of incoming objects. Now, during some period of time not only the ratios swap but also class concepts completely change, which means that both c_1 and c_2 should now be represented by new models. If an algorithm is based only on class ratios, it oversamples only the minority class c_1, improving its adaptation, while for the entirely new majority class c_2, which also requires significant updates, we will rely only on incoming instances. One may say that it is fine since we balance learning and if c_2 turned into majority it does not require additional instances. It is reasonable until we take into consideration the second crucial facet of learning from data streams – labeling budget constraints.

If a number of instances that can be used for updating is significantly limited, then not only minority instances may suffer from underfitting and require some amount of oversampling, but also we may encounter a high error for the majority classes. We call it a *deceptive majority*. Taking this fact into account may not only help with modeling the majorities more accurately, but also improve the minorities by excluding more precisely the subspaces to which they not belong. We assume that in such scenarios potential underfitting is more likely to occur and impede learning more than overfitting.

B. Framework

In this paper, we present our algorithms that are able to tackle all the mentioned problems emerging during learning from imbalanced data streams. We designed an online wrapper framework given in Algorithm 1.

Active learning. Our approach is, in fact a combination of active learning and oversampling techniques. The former one (`QueryStrategy`) is used to limit a number of labeling requests for incoming instances $\mathbf{x}$ by asking only for valuable ones given some criteria. If a current budget spending $\hat{b}$ does not exceeds an available budget B and the method decides that a new object should be labeled, we acquire a true class c for the instance and use it to update a classifier L. The available budget B is a fraction of instances that can be labeled. Since streams are infinite by definition, we approximate the current spending b with $\hat{b}$ as a ratio of labeled instances to all already acquired. A few online strategies have been already proposed that can be used in our framework [21]. They are usually based on uncertainty, like RandVar or selective sampling, however, to the best of our knowledge, there are no active learning strategies for multi-class imbalanced streams with dynamic ratios. We do not focus on this module in our work. Instead we are going to show that problems with limited budgets (underfitting) and class imbalance, while relying on the active learning alone, can be effectively handled by adding an additional module responsible for oversampling.

Algorithm 1: The framework for learning from imbalanced data streams on a budget.

Data: labeling budget B, `QueryStrategy`, `OversamplingStrategy`, budget spending $\hat{b}$, generated instances $\mathbf{S}$, metrics $\mathbf{M}$
Result: classifier L at every iteration
Initialization: $\hat{b} \leftarrow 0$, $\mathbf{S} \leftarrow []$, $\mathbf{M} \leftarrow []$
repeat
  receive incoming instance $\mathbf{x}$;
  if $\hat{b} < B$ and `QueryStrategy` $(\mathbf{x}) = true$ **then**
    request the true label c of instance $\mathbf{x}$;
    update labeling expenses $\hat{b}$;
    update classifier L with $(\mathbf{x}, c)$;
    update class metrics $\mathbf{M}[c]$
    $\mathbf{S} \leftarrow$ `OversamplingStrategy` $(\mathbf{x}, \mathbf{M}[c])$;
    for $i \leftarrow 1$ **to** $len(\mathbf{S})$ **do**
      update classifier L with $(\mathbf{S}[i], c)$;
until *stream ends*;

Oversampling. While the active learning approach is a step forward to handle realistic streaming scenarios, it may still be insufficient under strict budget constraints, when a very limited number of instances is used. To handle this problem, we propose an adaptation enhancement in a form of synthetically generated instances. Since in this work we focus on skewed data distributions, we use oversampling techniques for this purpose. We generate additional instances $\mathbf{S}$ accordingly to a given `OversamplingStrategy`. It takes the labeled instance $\mathbf{x}$ as a prototype and class metrics $\mathbf{M}[c]$ for the class c the instance belongs to. The oversampling strategy may use different: (i) generation methods that defines how new instances are created; (ii) balancing strategies that determines how many instances are generated based on class metrics.

C. Generation methods

We specify two incremental generation methods that synthesize additional instances $\mathbf{S}$. They are rooted in the offline oversampling domain.

Single Exposition (`SE`) – it is a fully online approach that simply duplicates d times a given instance $\mathbf{x}$ that is exposed to the algorithm only once.

SMOTE (`SM`) – in this method we maintain a sliding window $\mathbf{W}$ of ω_{max} latest instances $\mathbf{w}$ that represent current concepts. Each class has its own window, so we keep ω_{max} already received instances for each of them. New instances are generated using the SMOTE algorithm [4]. For a labeled instance

$\mathbf{x}$ we find its k nearest neighbors (belonging to the same class c), generate synthetic instances using Algorithm 2 (the gap is calculated using the uniform distribution $\mathcal{U}(0,1)$) and then duplicate them d times.

Algorithm 2: SMOTE single instance generation.

Data: new instance $\mathbf{x}$, window instance $\mathbf{w}$
Result: generated instance $\mathbf{s}$

$gap \leftarrow \mathcal{U}(0,1)$;
for $i \leftarrow 1$ **to** $len(\mathbf{x})$ **do**
  $diff \leftarrow \mathbf{w}[i] - \mathbf{x}[i]$;
  $\mathbf{s}[i] \leftarrow \mathbf{x}[i] + gap * diff$;
return $\mathbf{s}$

D. Balancing strategies

This module is responsible for balancing the learning process. In general, we want to use a function $d(\mathbf{m_c}) = d_{max}\gamma(\mathbf{m_c})$, where d_{max} is a maximal number of duplications that can be created and $\gamma(\mathbf{m_c})$ is a balancing function transforming metrics $\mathbf{m_c}$ for a class c into a value $v \in \langle 0, 1\rangle$.

Dynamic Class Ratio (DCR). The most straightforward approach is to generate d instances in an negative relation to a ratio λ_c for a class c. While for binary problems we can apply the ratios directly to get:

$$\gamma(\mathbf{m_c}) = \gamma(\lambda_{0/1}) = 1 - \lambda_{0/1}, \tag{1}$$

it may not be reasonable to do the same for multi-class problems, since for example, for balanced cases we would unnecessarily oversample some of the classes. The easiest approach is to perform oversampling relatively to the majority class λ_{max} [19]. Then for each class we can define the ratio as $\lambda'_c = \lambda_c/\lambda_{max}$, so we will try to oversample up to the *largest* class:

$$\gamma(\mathbf{m_c}) = \gamma(\lambda'_c) = \gamma(\lambda_c, \lambda_{max}) = 1 - \lambda_c/\lambda_{max}. \tag{2}$$

Since we aim to solve not only multi-class but also dynamic ratio problems, we need to apply adaptation mechanism to the maintained class ratio values. We use the sliding window approach in our algorithms. Although the presented method is theoretically able to handle multi-class problems with dynamic ratios, it still does not take into account the problem of the *deceptive majority* (Sec. 2.1).

Dynamic Hybrid Ratio (DHR). A possible solution to the problem is adding a concept drift detector and using its indications to guide the class balancing. There are a few existing online drift detectors (e.g., DDM [6]), which indicate a change discretely (absent/present) based on registered errors. Since we want to control the number of duplications in a continuous way, we utilize a class-wise error calculated within a sliding window. Different performance metrics can be used for this purpose. In our case, we apply G-mean measure g_c that is calculated in one-vs-all manner for each class c separately [3].

For each incoming instance $\mathbf{x}$ we combine both metrics – the relative class ratio λ'_c and error $(1 - g_c)$ – using a simple weighted sum:

$$\gamma(\mathbf{m_c}) = \gamma(\lambda'_c, g_c) = \alpha_\lambda(1 - \lambda'_c) + \alpha_g(1 - g_c), \tag{3}$$

where $\alpha_\lambda + \alpha_g = 1$. Assuming that both coefficients are equal, one can easily see that when one class is simply a stationary majority (for example, $\lambda'_c = 0.8$ and $g_c = 1.0$), the strategy will practically ignore this class regarding oversampling, however, if a majority class is drifting (class error is expected to be high, for example, $\lambda'_c = 0.8$ and $g_c = 0.1$), the strategy will maintain some level of oversampling for this class to help a model adapt to a new class concept.

III. Experimental study

In our experimental study of learning from multi-class imbalanced streams with evolving class ratios, concept changes and under strict budget constraints we evaluate:

1) if our combination of active learning and oversampling improves the former (E1),

2) if the hybrid balancing strategies outperform the simple ratio-based approaches, as well as if one generation method is better than another (E2),

3) if our single-classifier framework is competitive, in terms of classification performance and time consumption, to other solutions proposed for learning from multi-class imbalanced data streams – MOOB/MUOB [19], as well as to other state-of-the-art ensembles that are very efficient in general cases.

A. Data stream benchmarks

To evaluate the given questions we utilized a set of 13 real benchmarks widely used in the data stream mining domain. Most of them come from the UCI repository (Connect4, Covertype, EEG, Gas, Poker) and Kaggle competitions (Crimes, Olympic, Tags). The rest of them are very popular in related publications. They are summarized in Tab. I.

TABLE I: Summary of the used data streams.

Name	Instances	Att	Cls	Dyn	SC	Drifts	Δ
Activity	10 853	43	8	✓	4	1	0.89
Activity-Raw	1 048 570	3	6	✓	4	1	0.99
Connect4	67 557	42	3	-	3	2	0.84
Covertype	581 012	54	7	✓	4	1	0.97
Crimes	878 049	3	39	-	4	2	0.98
DJ30	138 166	8	30	-	4	2	0.99
EEG	14 980	14	2	✓	2	-	-
Electricity	45 312	8	2	✓	2	1	0.98
Gas	13 910	128	6	✓	3	1	0.65
Olympic	271 116	7	4	-	3	2	0.95
Poker	829 201	10	10	✓	4	2	0.98
Sensor	2 219 804	5	57	✓	4	2	0.99
Tags	164 860	4	11	-	4	2	0.98

We decided to split our evaluation into two parts. The first one is based on those **real data streams** that exhibit significant variability of class ratios over time (Dynamic). Since all of the presented data streams are supposed to consist of concept

drifts, we can safely assume that, in at least some cases the class ratio dynamics occurs simultaneously with the concept changes. We selected 8 of such data streams (Fig. 1, up to 4 classes are shown to preserve clarity).

To make sure that we evaluate our algorithms exactly in the described cases, we generated additional 12 **semi-synthetic data streams**, based on the real ones, using two fully controlled modifications. Firstly, we assigned classes in the streams to supersets (superclasses C_i), creating usually highly imbalanced majority and minority concepts. Secondly, we changed the assignments at some points to simulate class ratio and concept drifts. For example, if in Activity-Raw we assigned *Walking* and *Jogging* objects (about 70% of all instances) to a superclass C_1, and *Standing* objects (less than 5%) to C_2, then during a drift we reverse the relation, so all *Walking* and *Jogging* objects are C_2 now and all *Standing* instances become C_1. As a result, we simulate critical class ratio changes (C_1 is about 5% and C_2 is 70% after the change), as well as concept drifts, since both superclasses are represented by different distributions of objects before and after a change. The concept transitions were generated analogously to the formula from MOA, using the sigmoid function:

$$f(t) = 1/(1 + e^{-s(t-t_0)}), \tag{4}$$

where s controls the duration of change and t_0 is a peak of it. We created drifts of moderate lengths. All necessary details regarding the generation process can be found in a repository [1].

In Tab. I we enclose a number of such changes (Drifts) and the proportion of objects that change concepts due to drifts (Δ). Both the concept drifts and class ratio changes are severe in almost all cases, therefore the generated data streams represent the most difficult scenarios we can encounter. If we also take into consideration the fact that, most likely, not all of the class ratio changes in the real streams occur with concept drifts at the same time, it is reasonable to say that, when it comes to handling the described dynamics, the generated streams are more challenging on average than the selected real ones. Since obtained class ratios are dynamic, we enclosed their values over time as an appendix in the repository.

B. Set-up

Below we present the set-up of our experiments. They can be easily reproduced, using the environment available on the given website.

Algorithms. To investigate if our combination of oversampling and active learning improves the latter (E1), we collected results for random `AL-R` (Random), `AL-RV` (RandVar) and `AL-S` (Sampling) without instance generation. We evaluated all combinations of our strategies `SE-DCR`, `SE-DHR`, `SM-DCR`, `SM-DHR` to check if there are substantial differences between generation and balancing strategies (E2). As an active learning strategy for our framework we picked theoretically universal `AL-RV` and Adaptive Hoeffding Tree (AHT) [1] as a base learner, which is a state-of-the-art classifier in the streaming data domain. Finally, we juxtaposed results for our strategies with already published ensembles for multi-class dynamic ratio streams – `MOOB` and `MUOB` (E3). In addition, we compared them with other well-known ensembles: Online Bagging (`OZABAG`) [16], Leveraging Bagging (`LB`) [2], Adaptive Random Forest (`ARF`) [8], Online Boosting using ADWIN (`OB-ADW`) [16] and Dynamic Weighted Majority (`DWM`) [10]. The ensembles used different versions of the Hoeffding Tree, depending on their default settings. All of them were connected with the `AL-RV` strategy for working on a budget.

[1] github.com/mlrep/imb-drift-20

Budgets. The algorithms were evaluated on different budgets, with a particular focus on realistic low ones, $B \in \{100\%, 50\%, 20\%, 10\%, 5\%, 1\%, 0.5\%, 0.1\%\}$.

Configurations. We varied the size of windows for `SE` and `SM`, depending on a budget, to make them reactive to the limited number of labeled instances $\omega_{max} \in \{1000, 500, 200, 100, 50, 10, 10, 10\}$. We set a fixed maximum number of duplications $d_{max} = 100$, a fixed number of nearest neighbors $k = 10$ for `SM`, as well as, equal coefficients for the `DHR` strategies: $\alpha_\lambda = \alpha_g = 0.5$. For the size of ensembles we chose 10 base learners (we have not observed significant improvement for larger ensembles). All the Hoeffding Trees used default settings. For `AL-RV` we selected default $\theta = 0.01$ as its variable threshold step.

Metrics. We collected classification efficacy and computing performance for all classifiers. For the former, we used the generalized multi-class form of G-mean, which is given as $G_n = \sqrt[n]{R_1 \cdot R_2 \cdot ... \cdot R_n}$, where R_i is a class-wise recall and n is a number of classes [3]. It was calculated using the prequential evaluation method. Bonferroni-Dunn ranking test with significance level $\alpha = 0.05$ was used to compare examined algorithms over multiple datasets. For the performance of computations we registered update and classification time per instance separately.

C. Results and discussion

We present the average results for all algorithms and data streams under different budget constraints in Tab. II.

Improving active learning. The first observation is that our framework was able to enhance results over simple active learning strategies in all cases except for `SM-DCR` on $B = 100\%$. For the real data streams, we can observe that the `SM` strategies provide a steady increase of the improvements, compared with the best active learning strategy for a given setting, as budget constraints are being tightened from $B = 100\%$ down to $B = 0.5\%$ (Fig. 2). It starts from about 1.1 for `SM-DHR` and ends at more than 1.36 for the same strategy. Results for the `SE` approaches exhibit the same trend for budgets higher that $B = 1\%$, with a slightly lower values between approximately 1.09 and 1.29. Below the given budgets, `SM` and `SE` still provide some gain, however, they are no longer able to increase it. For the harder semi-synthetic streams the trend

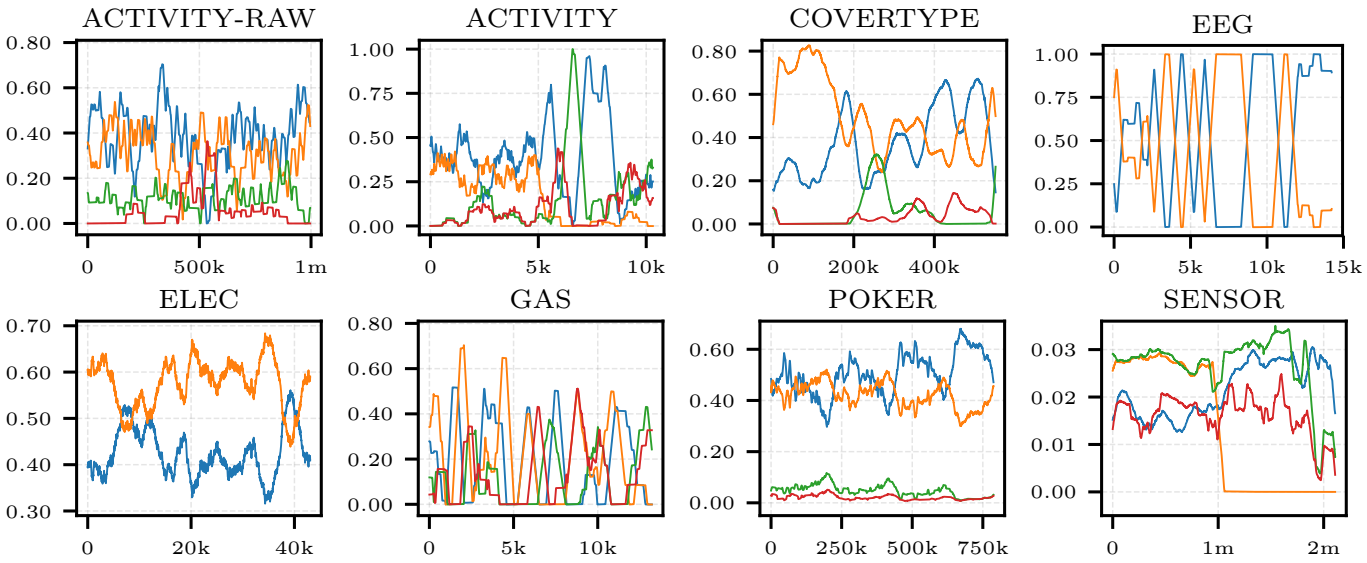

Fig. 1: Dynamic class ratios for the used multi-class real data streams, each color represents a different class.

TABLE II: Average G-mean values calculated over all real streams (top) and semi-synthetic streams (bottom) for different algorithms given a budget.

REAL	100%	50%	20%	10%	5%	1%	0.5%	0.1%
AL-R	0.6415	0.6014	0.5586	0.4919	0.3966	0.3405	0.3237	0.2849
AL-RV	0.6158	0.5981	0.5327	0.4637	0.4531	0.3699	0.3175	0.2768
AL-S	0.6183	0.6034	0.5493	0.4775	0.4362	0.3802	0.3233	0.2702
SE-DCR	0.6983	0.7103	0.6580	0.6187	0.5724	0.4667	0.3741	0.3393
SE-DHR	0.7179	0.7250	0.6820	0.6282	0.5839	0.4561	0.3816	0.3318
SM-DCR	0.7048	0.7121	0.6625	0.6087	0.5706	0.4850	0.4313	0.3421
SM-DHR	0.7397	0.7395	0.6901	0.6459	0.5985	**0.4980**	**0.4424**	**0.3534**
MOOB	0.6441	0.6518	0.5756	0.5248	0.4880	0.3953	0.3619	0.3194
MUOB	0.2290	0.2191	0.2223	0.2165	0.2090	0.1774	0.2065	0.1622
OZABAG	0.6140	0.6150	0.5262	0.4852	0.4528	0.3984	0.3174	0.2757
LB	0.7404	0.7419	0.6946	0.6436	**0.6064**	0.4694	0.3461	0.2685
ARF	**0.7597**	**0.7596**	**0.7001**	**0.6524**	0.5977	0.4421	0.3453	0.2012
OB-ADW	0.6432	0.6324	0.5936	0.5538	0.5088	0.3886	0.3202	0.2498
DWM	0.7222	0.7223	0.6690	0.6181	0.5751	0.4254	0.3497	0.2866

SYNTH	100%	50%	20%	10%	5%	1%	0.5%	0.1%
AL-R	0.7229	0.6677	0.5970	0.5668	0.5088	0.3685	0.2992	0.2071
AL-RV	0.6628	0.6617	0.6047	0.5727	0.5108	0.3840	0.3079	0.1672
AL-S	0.6884	0.6713	0.6090	0.5642	0.5429	0.3882	0.2868	0.1593
SE-DCR	0.7399	0.7458	0.6973	0.6898	0.6645	0.5626	0.5434	0.4062
SE-DHR	0.7636	0.7758	0.7363	0.7144	0.6720	0.5707	0.5391	0.4330
SM-DCR	0.7198	0.7104	0.6713	0.6357	0.5954	0.5280	0.4967	0.4245
SM-DHR	0.7862	0.7938	**0.7688**	**0.7439**	**0.7142**	**0.6359**	**0.5946**	**0.5019**
MOOB	0.7369	0.7456	0.6743	0.6451	0.5922	0.4821	0.4256	0.3002
MUOB	0.4471	0.4208	0.4124	0.3895	0.3716	0.3318	0.2562	0.2444
OZABAG	0.6287	0.6262	0.5605	0.4999	0.4213	0.3208	0.2802	0.2226
LB	0.7936	0.7913	0.7312	0.6651	0.6123	0.4995	0.4264	0.2895
ARF	**0.8183**	**0.8115**	0.7603	0.7138	0.6683	0.5430	0.4644	0.3086
OB-ADW	0.7787	0.7614	0.7479	0.7115	0.6752	0.5718	0.5034	0.3603
DWM	0.7215	0.7195	0.6564	0.6459	0.6005	0.4606	0.4103	0.2656

is even more clear (Fig. 2). The improvements for the SM strategies ranges from about 1.09 to more than 2.4 for DHR, and from 1.02 to almost 2.1 for SE using the same generation strategy. The most significant change can be observed after we limit the number of labeled instances below 5%, when the improvements become drastically higher. We suppose that the difference between results for the real streams and the semi-synthetic ones comes from the lower quality of the controlling metrics maintained by our algorithms (windowed class ratios, errors). It can be balanced by the difficulty of changes in a stream (like in the semi-synthetic ones and probably some of the real ones), when there is a higher chance that our approach will be effectively utilized.

Regardless of the quality of improvements, they are caused by the fact that the active learning strategies alone are not able to update base learners sufficiently while learning from extremely limited instances – single, sparsely labeled examples introduce inadequate changes to models in terms of reaction to skewed data distributions and severe concept drifts. Adding properly controlled oversampling helps with maintaining sufficiently balanced classifiers and prevents underfitting. The fact that we are able to increase the enhancements for lower budgets is particularly encouraging, since these are the most realistic scenarios [21]. Results of ranking tests for all budgets (Fig. 3 and 4) show the significance of the differences.

Hybrid over class ratio. When we look at different combinations of our generation and balancing strategies (Tab. II), we can conclude that methods based on DHR are generally better than those using DCR. One can also notice that the differences are more substantial for SM (up to about 0.04 for the real streams and up to almost 0.12 for the semi-synthetic ones) than for SE (up to 0.02-0.03 and 0.04, respectively), for which they are on the brink of significance when averaged over all examined budgets (Fig. 3 and 4). It may mean that improvements for the very simple generation strategy are harder to achieve.

The differences occur for both groups of data streams, however, they are definitely more significant for the semi-synthetic ones. The class ratio driven approaches are not the best solutions that we can find if with the class ratio changes come severe concept drifts. In such scenarios, especially when a budget is limited, majority classes also need to be sufficiently

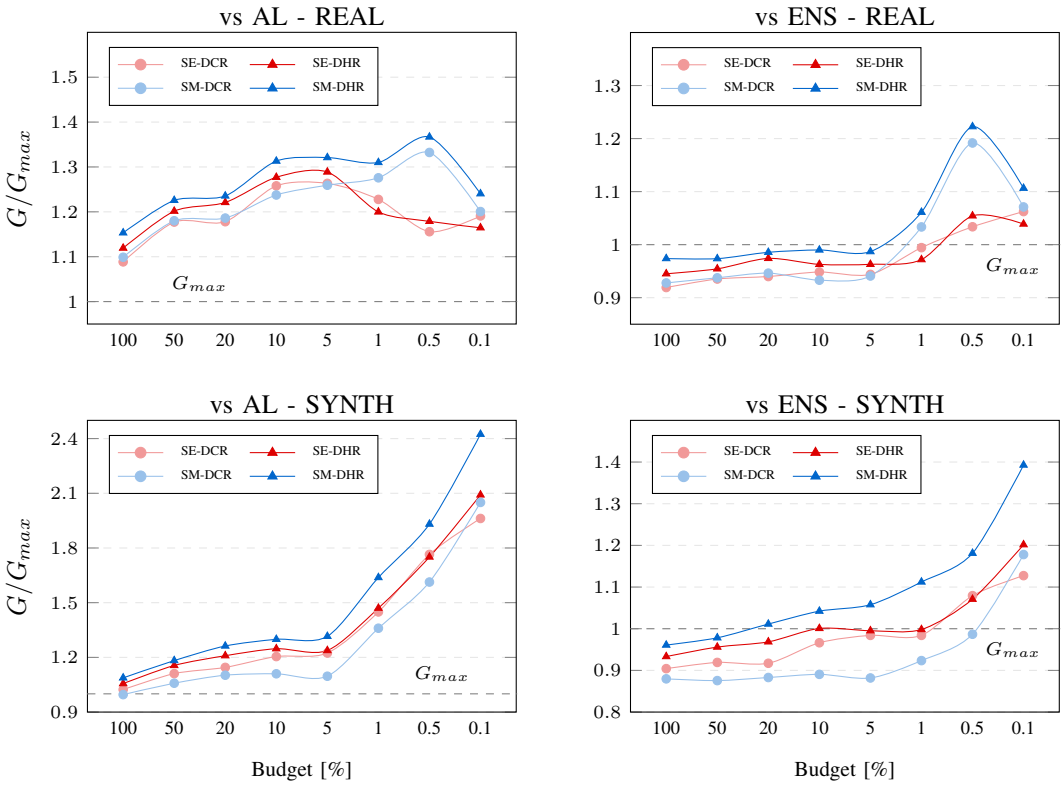

Fig. 2: Ratios of the average G-mean for our algorithms (G) to results for the best (G_{max}) active learning (left) and ensembles (right) on given budgets.

handled by boosting the adaptation process with additionally generated instances. The DHR strategies provide the additional objects, based on the drift indicator – an error for a class. One should also notice that even if we claim that the DHR approach is more useful when data streams are characterized by more severe simultaneous class ratio and concept changes, it almost never performs worse than the DCR strategy, regardless of the difficulty of drifts.

Generation methods. The observation that the gap between DCR and DHR is more clear for SM than for SE is correlated with the fact that the former works exquisitely well with DHR and disappointingly with DCR, especially for moderate budgets between $B = 20\%$ and $B = 5\%$. In particular, it can be seen for the semi-synthetic streams (Fig. 4). SE is more stable and combines better with DCR, however, at the same time it does not achieve as good results as SM with DHR. Eventually, we do not distinguish any generation method as significantly better than another.

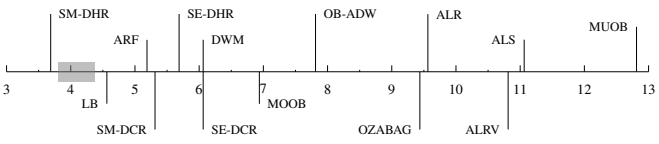

Fig. 3: The Bonferroni-Dunn test over all examined budgets for the real streams.

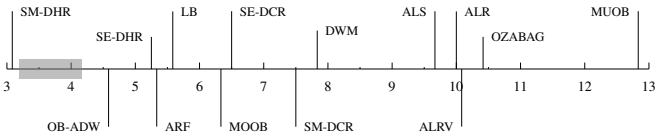

Fig. 4: The Bonferroni-Dunn test over all budgets for the semi-synthetic streams.

Comparison with ensembles. Most importantly, although our algorithms do not improve upon ensembles for high and very high budgets above $B = 20\%$, the relation between gain and budget is similar as for the active learning strategies. In most cases, except for the lowest budget for the real data stream, we can observe that with decreasing budget our chances for improvements increases (Fig. 2), which once again, is a very important property, since smaller numbers of labeled instances are more realistic.

Analogously to the results comparing our solutions with the active learning, we can observe that improvements upon the best ensembles on given budgets (usually LB or ARF) are much more clear for the results obtained from more challenging semi-synthetic streams. In particular, it can be noticed for our the most efficient combination – SM−DHR – which was better than any of the considered ensemble for the real streams on very low budgets below $B = 5\%$ (from about 1.05 to more than 1.2, Tab. II) and for the semi-synthetic streams

on low budgets below $B = 20\%$ (from 1.01 to nearly 1.4). The rest of our strategies were at least competitive on budgets lower than $B = 10\%$. One should notice that SM−DHR once again was resilient to very low budgets while compared to other algorithms. As a result, SM−DHR turned out to be the best algorithm overall (Fig. 3 and 4), outperforming all other classifiers. Also, SM−DCR was very competitive for the real streams (most likely due to the less severe concurrent class ratio and concept changes) and SE−DHR for the semi-synthetic ones.

Furthermore, it is worth mentioning that in nearly all cases with limited budget each of ours algorithms, except for SM−DCR working on the semi-synthetic streams, was better than MOOB and MUOB (Tab. II), which are considered state-of-the-art algorithms for the problem of learning from multi-class imbalanced data streams. In addition, one can notice that the DCR-based combinations, which use the similar balancing principle as MOOB, can also be better than the ensemble – SE−DCR exhibits higher quality for budgets lower than 50%, SM−DCR when less than 5% labeled instances are available. It is most likely caused by the fact that our strategies tend to generate much more additional instances than the bagging-based algorithms, so they less likely suffer from underfitting, like MUOB. Finally, the negative results for the undersampling ensemble prove that using this technique while working with highly limited budgets is not a reasonable approach.

The presented results show that our hybrid approach is adequate to the presented challenging scenarios and that currently available solutions can be meaningfully improved, especially under realistic budget constraints.

Time consumption. In Tab. III we enclose the average total running time per instance calculated over all data streams (real and semi-synthetic), as well as we distinguish proportions (bars in cells) of the time used for updates and classification. Interestingly, while ensembles spends more time on classification, our strategies use most of it for updates. Generally, there is also a pattern of dominating updates on higher budgets for all algorithms – it is probably caused by the nature of the base learner used (Hoeffding Trees), which for more labeled instances builds more complex structures that require more time-consuming updates.

For high budgets above 10% we can segregate the solutions into three groups – fast active learning methods and MUOB (about 0.005-0.02 ms), moderate SE along with all other ensembles (0.06-0.3 ms), and relatively very slow SM strategies (0.48-1.11 ms). The performance of the last one is caused mainly by the naive nearest neighbor search within sliding window, which can be improved using a dedicated data structure. On the other hand, one should also notice that the differences significantly changes as budgets get lower – and these are the scenarios to which we dedicate our methods.

It is worth noting that since all our strategies depends on the number of classes (class ratio and error), we observed that the total running time for the real data streams is higher on average than for the semi-synthetic streams (larger numbers of classes).

TABLE III: The average total running time [ms] per instance for all algorithms given a budget. Update time is blue, classification is red.

	100%	50%	20%	10%	5%	1%	0.5%	0.1%
AL-R	0.0203	0.0104	0.008	0.0099	0.0061	0.0052	0.0042	0.0041
AL-RV	0.0107	0.0124	0.0101	0.0064	0.0056	0.0051	0.0049	0.0041
AL-S	0.0135	0.0127	0.0128	0.0063	0.0059	0.0045	0.0043	0.0041
SE-DCR	0.1741	0.1596	0.0788	0.0473	0.0275	0.0093	0.0067	0.0049
SE-DHR	0.1614	0.1485	0.0807	0.0486	0.0306	0.0111	0.0083	0.0055
SM-DCR	1.1116	1.0767	0.4873	0.2579	0.1457	0.039	0.0225	0.0089
SM-DHR	0.9837	0.9403	0.4716	0.2699	0.166	0.051	0.027	0.0102
MOOB	0.1329	0.1482	0.0919	0.0689	0.0655	0.0556	0.054	0.0521
MUOB	0.0194	0.0199	0.0221	0.0197	0.0167	0.0182	0.0149	0.0114
OZABAG	0.0764	0.0827	0.0633	0.0562	0.0526	0.0471	0.0464	0.0446
LB	0.1187	0.1235	0.0822	0.0672	0.0588	0.0476	0.0446	0.0459
ARF	0.0722	0.0638	0.0372	0.0276	0.0225	0.0171	0.0164	0.015
OB-ADW	0.3092	0.2153	0.1193	0.0584	0.0467	0.0386	0.0355	0.0279
DWM	0.0833	0.0764	0.0524	0.0368	0.0284	0.0259	0.0263	0.0239

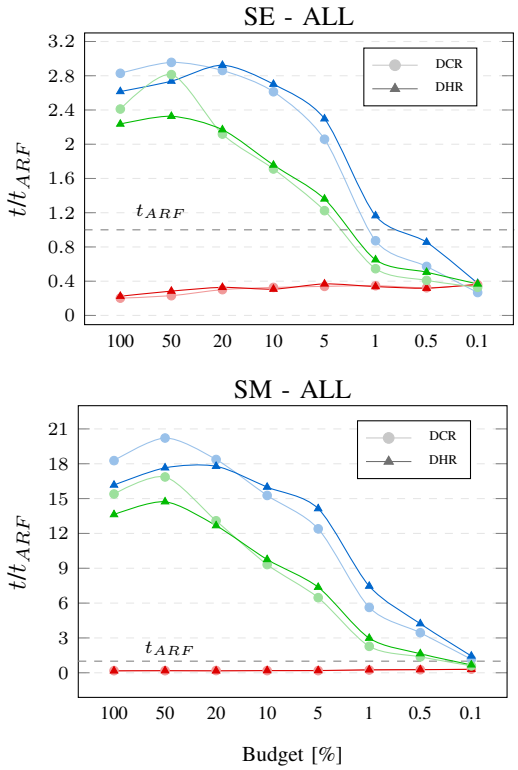

Fig. 5: Ratios of running time per instance for our algorithms (t) to the results for ARF (t_{ARF}). The update time is blue, classification is red and the total time is green. Ratios for DHR are in darker colors than for DCR.

In Fig. 5 we can observe how the ratios of the computation time for our strategies to measurements for the best ensemble – ARF (on average) – change with budget. We can clearly see that as budget decreases the ratios decrease, in favor of our methods. The SE approaches are competitive on the highest budgets and become even faster than ARF, if less than 5% labeled objects are available. The SM methods are more than 15

times slower than ARF on $B = 50\%$, however, they smoothly reduce the processing time (smaller windows, simpler AHT) and become competitive for budgets below 5%. Furthermore, for both generation methods we can see that the DCR strategies are slightly faster than DHR (probably because the latter tend to generate more instances). Finally, even if ratios for the update time remains unfavourable in most cases, the overall time reduces faster, since as the update time drops for all algorithms the ensemble classification time starts dominating not only in proportions (Tab. III) but also in absolute values, compared with our single-classifier framework.

IV. Conclusions and future works

To conclude with, in our work we presented a single-classifier framework addressing the problem of learning from multi-class imbalanced data streams with dynamic class ratios and concurrently drifting concepts. We analyzed our and referential solutions under wide range of budgets for labeling, including very strict constraints when even less than 1% of labeled instances are available. The experimental results most importantly show the following.

1) Combining active learning with oversampling improves the former by preventing underfitting and equilibrating adaptation between classes.

2) Hybrid ratio balancing strategies enhance simple approaches based on class ratio when dealing with concurrent ratio and concept changes.

3) Our single classifier framework using the best configuration (SM−DHR) is able to outperform existing ensemble solutions, which ignore the fact that concept changes may occur simultaneously also for dominating classes, making themselves susceptible to what we call the *deceptive majority*.

4) Our solutions are competitive also in terms of running time per instance.

Finally, we observe that the presented strategies exhibit their primacy over active learning and the ensembles especially when the number of labeled instances is critically low – it reflects in both classification quality and computing performance. We find it essential, since these are the most realistic scenarios one can encounter. Taking into account both metrics, we recommend using SE−DHR when relatively higher budgets are available (above 5%) and SM−DHR for the highly limited ones (below 5%). They provide the best quality-time improvement ratio for the given ranges of budgets.

In our future works, we will consider providing more in-depth analysis of used parameters (window sizes, numbers of duplications, hybrid ratio weights) in the context of different concept and ratio changes, including their severity. We may also investigate other than G-mean measures for balancing strategies.

References

[1] A. Bifet and R. Gavaldà. Adaptive Parameter-free Learning from Evolving Data Streams. Technical report, Universitat Politècnica de Catalunya, 2009.

[2] A. Bifet, G. Holmes, and B. Pfahringer. Leveraging bagging for evolving data streams. In J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 135–150, 2010.

[3] P. Branco, L. Torgo, and R. P. Ribeiro. Relevance-based evaluation metrics for multi-class imbalanced domains. In *Advances in Knowledge Discovery and Data Mining - 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part I*, pages 698–710, 2017.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.*, 16:321–357, 2002.

[5] S. Ding, B. Mirza, Z. Lin, J. Cao, X. Lai, T. V. Nguyen, and J. Sepulveda. Kernel based online learning for imbalance multiclass classification. *Neurocomputing*, 277:139–148, 2018.

[6] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence – SBIA 2004*, pages 286–295, 2004.

[7] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, 2014.

[8] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9):1469–1495, Oct 2017.

[9] B. Gulowaty and P. Ksieniewicz. SMOTE algorithm variations in balancing data streams. In *Intelligent Data Engineering and Automated Learning - IDEAL 2019 - 20th International Conference, Manchester, UK, November 14-16, 2019, Proceedings, Part II*, pages 305–312. Springer, 2019.

[10] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: a new ensemble method for tracking concept drift. In *Third IEEE International Conference on Data Mining*, pages 123–130, Nov 2003.

[11] Ł. Korycki, A. Cano, and B. Krawczyk. Active learning with abstaining classifiers for imbalanced drifting data streams. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2334–2343, 2019.

[12] B. Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in AI*, 5(4):221–232, 2016.

[13] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017.

[14] B. Mirza, Z. Lin, J. Cao, and X. Lai. Voting based weighted online sequential extreme learning machine for imbalance multi-class classification. In *2015 IEEE International Symposium on Circuits and Systems, ISCAS 2015, Lisbon, Portugal, May 24-27, 2015*, pages 565–568, 2015.

[15] S. Mohamad, A. Bouchachia, and M. Sayed Mouchaweh. A bi-criteria active learning algorithm for dynamic data streams. *IEEE Trans. Neural Netw. Learning Syst.*, 29(1):74–86, 2018.

[16] N. C. Oza. Online bagging and boosting. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2340–2345 Vol. 3, Oct 2005.

[17] Y. Sun, K. Tang, L. L. Minku, S. Wang, and X. Yao. Online ensemble learning of data streams with gradually evolved classes. *IEEE Trans. Knowl. Data Eng.*, 28(6):1532–1545, 2016.

[18] S. Wang, L. L. Minku, and X. Yao. A multi-objective ensemble method for online class imbalance learning. In *2014 International Joint Conference on Neural Networks, IJCNN 2014, Beijing, China, July 6-11, 2014*, pages 3311–3318, 2014.

[19] S. Wang, L. L. Minku, and X. Yao. Dealing with multiple classes in online class imbalance learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2118–2124, 2016.

[20] S. Wang, L. L. Minku, and X. Yao. A systematic study of online class imbalance learning with concept drift. *IEEE Trans. Neural Netw. Learning Syst.*, 29(10):4802–4821, 2018.

[21] I. Zliobaite, A. Bifet, B. Pfahringer, and G. Holmes. Active learning with drifting streaming data. *IEEE Trans. Neural Netw. Learning Syst.*, 25(1):27–39, 2014.