Classifying and Counting with Recurrent Contexts

Denis Reis, André Maletzke
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, São Paulo, Brazil
{denismr,andregustavo}@usp.br

Diego F. Silva
Departamento de Computação
Universidade Federal de São Carlos
São Carlos, São Paulo, Brazil
diegofs@ufscar.br

Gustavo E. A. P. A. Batista
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, São Paulo, Brazil
gbatista@icmc.usp.br

ABSTRACT

Many real-world applications in the batch and data stream settings with data shift pose restrictions to the access to class labels after the deployment of a classification or quantification model. However, a significant portion of the data stream literature assumes that actual labels are instantaneously available after issuing their corresponding classifications. In this paper, we explore a different set of assumptions without relying on the availability of class labels. We assume that, although the distribution of the data may change over time, it will switch between one of a handful of well-known distributions. Still, we allow the proportions of the classes to vary. In these conditions, we propose the first method that can accurately identify the correct context of data samples and simultaneously estimate the proportion of the positive class. This estimate can be further used to adjust a classification decision threshold and improve classification accuracy. Finally, the method is very efficient regarding time and memory requirements, fitting data stream applications.

CCS CONCEPTS

• **Computing methodologies → Machine learning approaches**;

KEYWORDS

Classification, Counting, Quantification, Concept Drift, Dataset Shift, Recurrent Contexts

ACM Reference Format:
Denis Reis, André Maletzke, Diego F. Silva, and Gustavo E. A. P. A. Batista. 2018. Classifying and Counting with Recurrent Contexts. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220059

1 INTRODUCTION

Classification and quantification (counting) are two different, but intimately related data mining tasks. Classification assigns a class label to an individual and unknown example. Quantification estimates the class distribution for an unlabeled sample of instances.

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220059

These tasks can benefit from each other. We can naively quantify a sample by classifying and counting each example (although approaches that take into consideration other pieces of information, such as the marginal errors, outperform this method [15]). Conversely, we can use the quantification output to adjust the decision threshold of classifiers and, therefore, improve classification accuracy in applications with varying class distributions.

In this paper, we are interested in non-stationary problems, both in batch and stream settings. Alongside current quantification literature, we assume that the class priors ($P(Y)$) dynamically change over time. Otherwise, we could just use the training set distribution to quantify the test data. Unlike other quantification papers, we assume that the data distribution ($P(X)$) will change. The batch and streaming mining literature reference these changes by several different names such as dataset shift, covariate shift, concept drift and so forth.

This setting is extremely challenging, not only because we are assuming a non-stationary environment, but also because we cannot rely on actual labels after the classifier is deployed. In other words, the difference in time between the arrival of a new example and the observation of its label is very large or even infinite. In the data stream jargon, we have a very large or extreme label verification latency [28]. Small or null-verification latencies are usually unrealistic assumptions and would make this problem uninteresting: instead of building classifiers/quantifiers, we could just wait for the correct labels to arrive and use them as a response to the classification and quantification problems.

Current literature has addressed the verification latency problem with three primary approaches: active learning [26], semi-supervised [21, 22] and unsupervised learning [1, 9, 11, 28] for both batch and stream settings. Active learning typically requires class labels when drifts occur and are particularly problematic for stream settings since it would demand an expert "over the phone" to label instances at any moment. Semi-supervised learning needs labels for part of the examples, and it may be infeasible for several classification and quantification applications. Unsupervised learning typically tracks drifts associating clusters and classes. Although this last approach requires no labels after deployment, it also suffers from the difficulties of designating groups to concepts and frequently relies on strong assumptions regarding the nature of the changes, such as incremental variations in $P(X)$ and nearly constant $P(Y)$.

With this work, we propose a novel and practical approach to the problem of learning from non-stationary data, with the absence of class information during deployment. We assume that the data can be (approximately) described by a set of recurring concepts that we call *contexts*, and that we have access to labeled data for these

contexts at training time. We also assume that although an extensive set of latent factors can cause concept drifts, there are smaller subset of observable variables related to most of them. Although we have access to these variables during training, we do not assume their availability afterward.

Our primary contribution is a method that, given an unlabeled sample at deployment, returns the most similar data seen at training. In other words, it returns what is the most likely known distribution (context) to be followed by the sample. Contrarily to other statistical tests for matching data distribution [10], our method is robust regarding extreme changes in $P(Y)$.

Given the unlabeled deployment sample, our method provides not only its context but also an estimate of the current class distribution $\hat{P}(Y)$. The utility of such estimate is two-fold. First, it is the direct answer to quantification queries. Second, we can use it to adjust the decision threshold of the classifiers, providing accurate results for applications where the class priors vary significantly.

We make a comprehensible evaluation of our proposal and show that:

(1) We can identify the correct context under varying class distributions with high accuracy;
(2) For quantification problems, our method is significantly more accurate than the state-of-the-art under changes in $P(X)$. Additionally, other quantification methods can benefit from our context identification approach;
(3) Our method is more accurate than the state-of-the-art in data stream classification with data drifts in $P(X)$ and $P(Y)$ and total absence of class labels;
(4) Our method is significantly more accurate than a single model that ignores contexts altogether, for both quantification and classification;
(5) Our proposal is efficient and prone to an incremental implementation, fitting stream settings.

This paper is organized as follows. Section 2 presents a motivation for this research; Section 3 reviews the literature regarding the tackled problem; Section 4 describes our proposal; Section 5 explains our experimental setup; Section 6 presents and discusses our findings; Section 7 discusses our proposal's efficiency; and Section 9 presents our final conclusions and prospects for future work.

2 MOTIVATION

In the last years, our research group has designed a sensor for flying insects [27]. The sensor uses infrared light to capture the movement of the insect wings and machine learning to classify the insect signals into species and sex.

This sensor can augment existing mosquito traps, creating a device that can attract, capture, and also count the insects. Hence, such trap can be a valuable tool for mosquito control and surveillance.

For control, classification is the primary task. Real-time classification allows us to create a trap that only captures insects of interest, such as a disease vector or agricultural pest. Other species can be released, reducing the environmental impact of the device.

For surveillance, quantification is the primary task. Counting the number of captured insects by species and sex over time gives us an estimate of the insect population in the trap area. Heat maps integrate the counts of multiple traps and provide an essential tool for teams to plan and execute larger-scale control interventions.

Data collected by the trap presents drifts. There are a large set of factors that influence the flying behavior of insects (and, therefore, $P(X)$) such as temperature, humidity, air pressure, age, availability of water and food and so forth. However, temperature is a prominent factor. Figure 1 illustrates temperature influence on the wing-beat frequency, one of the features we extract from the signals.

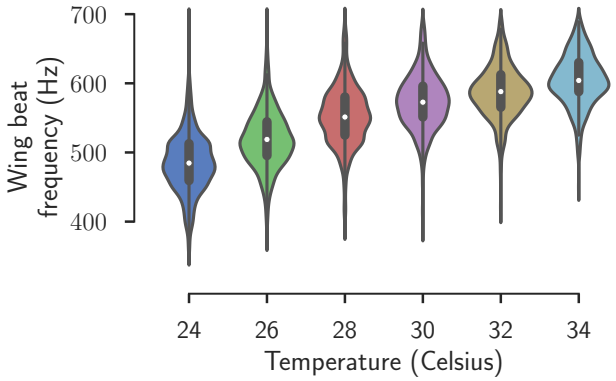

Figure 1: Influence of temperature on the wing-beat frequency of *Aedes aegypti* mosquitoes.

An aggravating factor is that the class distribution is unknown and highly variable. It depends on two primary factors: the local availability of the species of interest and the insect circadian rhythm. The circadian rhythm is a biological process that governs peaks of activity and resting. For many insect species, these peaks occur at dawn and dusk, as shown in Figure 2.

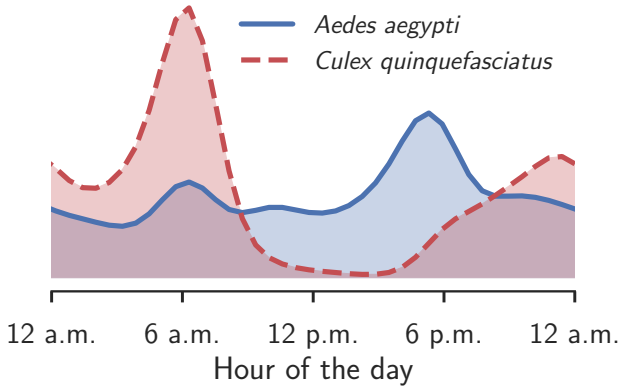

Figure 2: Histograms representing the circadian rhythm of *Aedes aegypti* and *Culex quinquefasciatus* mosquitoes.

For this application, we possess plenty of labeled *training* data. We gather these data in the laboratory using insectaries, which are containers with a sensor attached. Each insectary maintains

insects of a single species, leading to labeled data. We also control temperature, humidity, and other ambient conditions maintaining the insectaries in climatized chambers.

However, training data offers limited information regarding class distributions ($P(Y)$). In the laboratory, the class distribution depends on the number of insects in each insectary and the circadian rhythm of the insects. However, on the field, the class distribution will be given by the local availability of the insects as well as their circadian rhythms.

Additionally, laboratory conditions differ from those in the deployment field and as such, we cannot count on the same data availability. In the laboratory, some factors such as true labels are known only because we directly control them. Those become promptly unavailable, once we achieve deployment.

While we can measure many of the factors that influence the insects behavior through sensors, we would want to remove all non-essential ones, since they raise battery consumption, building costs, and project complexity. The ambient sensors are particularly troublesome since they must be installed externally to better register the ambient conditions. Therefore, they are also prone to damages, resulting in failures.

Notwithstanding, the possibility of controlling some factors (such as temperature, humidity, air pressure, and so on) that affect the wing beat data ($P(X)$) enables us to collect enough data to identify a reasonable portion of all recurrent behavioral patterns.

We note that other applications may fall into one of four similar situations that follow:

(1) The factors responsible for different contexts are unknown, too many, or too complex to be feasibly accountable. In this case, it may still be possible to recognize distinct and recurrent contexts over time (by applying a suitable statistical test, for instance);
(2) Even if the factors responsible for drifts are known, they can be missing in data obtained with a different setup;
(3) The sensors responsible for registering data related to the drifts are prune to failures. Hence a fallback mechanism is desired; and
(4) The sensors are unreliable. Hence the use of their readings can be potentially improved with an additional opinion, as in a data fusion scheme.

3 RELATED WORK

In this section, we review the most relevant related work in data drift classification and quantification.

3.1 Concept Drift, Dataset Shift, and Context Drift

In the literature of batch and stream settings, changes in data distribution attend by several names such as concept, dataset and covariate drift, and shift.

In data streams, concept drift refers to changes in the probability distribution of the data over time. As distributions change, outdated models can have their performance decreased unless they can detect and adapt to such changes. In classification, a conventional approach is to detect concept drifts by measuring the classification performance over time and looking for declines. However, such approach depends on the timely availability of actual labels.

Verification latency is the period between observing an event and acquiring its true label and label scarcity is the number of actual labels accessible after classification. While scarcity and latency rates entirely depend on the application domain, they are rarely inexistent in real-world applications: a period and cost necessary for an external classification should exist. Yet, most research papers assume the absence of latency [21, 22], scarcity [2, 19, 21, 29] and, more often, both [3–5, 13, 24].

The extreme verification latency, i.e., extreme scarcity, is the scenario where no true labels are ever available during the deployment time of a classification model [8]. Consequently, there are no labels available for drift detection or model adjustment. Nonetheless, labeled data is available during a training phase.

There is a small body of work on streaming classification assuming extreme verification latency. For instance, concept drifts has been moddelled as data clusters movements in the feature space [9, 11, 28], which limits these methods to deal with incremental drifts and nearly constant class distributions.

However, the co-occurrence of incremental drifts and extreme class distribution variation can mislead these methods. For instance, the insect sensor only registers the insects that pass in front of it. Hence, the chance of registering flights for a species depends on its presence and activity. The insect circadian rhythm governs its activity periods, making it vary throughout the day. The chance of registering events for a species diminishes when it is inactive. When that species becomes active again, it can have a different data distribution than its last active period. Therefore, it may be infeasible to infer an incremental drift due to the lack of data between consecutive active periods.

Another active research area is the use of statistical tests to detect changes in data distribution. To the best of our knowledge, Kifer et al. [18] are the first to propose directly applying hypothesis tests to detect concept drifts without true labels. In this case, a fixed sample of events represents a training set and a dynamic sample represents recently observed data. A hypothesis test is then applied to verify whether the distributions of these samples are similar. The authors focused entirely on the task of drift detection and as such did not evaluate the impact on other related tasks, such as classification.

Using a similar approach, Zliobaite [30] proposes a method that can identify detectable changes in a stream: a hypothesis test is performed to verify if two consecutive sliding windows follow different distributions. An attractive proposal of this paper is the application of the test over the classifier output scores, instead of the feature values. As most statistical tests are univariate, this idea simplifies the test application and the decision process of indicating when a drift occurred. Nonetheless, she does not address possible actions after a drift detection.

Reis at al. [26] introduce an incremental version of the Kolmogorov-Smirnov test that suits data stream applications. They evaluate the impact of concept drift detection in classification regarding accuracy and number of labels demanded to update the model. Their setting shares similarities with active learning since their method requires labeled data to retrain the models after a concept drift.

Despite presenting favorable results at detecting drifts, previous work lacks unsupervised mechanisms to tackle drifts after detection, ceding the very first assumption of extreme verification latency.

In the batch setting, dataset shift is the term used to denote changes in data distribution. In this case, the data used to induce a model has a different probability distribution than the data in the deployment. One example of a method that deals with dataset shift is the Versatile Decision Tree [1]. They can adapt to batches of data that are monotonic transformations of the original data used at training. Besides the strong assumption of monotonic transformation, they also require the proportion of classes to be known.

Reis at al. [10] revisited the problem with a new assumption that enables tackling concept drifts without requiring true labels, after deployment. Concept drifts are restricted to a limited number of known and recurrent concepts, referred as *contexts*. The researchers demonstrated that individual models for each context outperform a single model trained with data from all contexts, and introduced a method based on the Kolmogorov-Smirnov test to switch between those individual models. However, the proposed method is just slightly better than their proposed baseline, and the proposal is incapable of dealing with varying class distributions, as Kolmogorov-Smirnov is sensitive to changes in both $P(X)$ and $P(Y)$. This means that if the only difference between two sets of data is the proportion of the classes, the Kolmogorov-Smirnov statistic could still be high, indicating that the distributions are different.

In this paper, we preserve the assumption of recurrent and limited contexts. However, we also expand it to varying proportion of classes. The latter assumption is not only more realistic to data quantification problems, but also an integral part of our approach.

3.2 Quantification

Quantification is the task of estimating the class distribution of a given dataset. One straightforward way of achieving quantification is to count the classes output by a classifier. This simple approach, known as Classify And Count (CC) [14], reveals the first detachment between these tasks: an accurate quantifier is not necessarily an accurate classifier. The error committed by a binary quantifier is always the absolute difference between the number of false positives and false negatives. If they are the same, the quantifier obtains perfect estimation, regardless the accuracy of the underlying classifier.

The Adjusted Classify and Count (ACC) [12] is a more sophisticated and commonly used quantification method. It accounts for the errors committed by its underlying classifier to adjust an estimation previously obtained with CC. Its gaudy point is that if the estimations for the errors are precise, so will be the estimate for the class proportions. However, it becomes difficult to assure accurate estimates for those errors when the distribution of the data is non-stationary.

There are quantification approaches that differ from counting single classifications. For instance, HDy [16] is a method that relies on the statistical divergence between the scores that a classifier outputs for positive and negative events.

The HDy algorithm builds two normalized histograms, H_+ and H_-, for the scores obtained by the classifier on two validation sets with exclusively positive and exclusively negative instances, respectively. When presented with an unlabeled test set, the algorithm builds a histogram $H_?$ with the scores obtained by the same classifier. Then, it estimates the positive proportion rate as follows:

$$\text{HDy}(H_+, H_-, H_?) = \arg\min_{0 \le \alpha \le 1} \{\text{HD}\left(\alpha H_+ + (1-\alpha)H_-, H_?\right)\}$$

where HD is the Hellinger Distance [25], and each histogram, with B bins, is represented as a vector in the $\mathbb{R}^B$. Hellinger Distance is a function that quantifies the similarity between two probability distributions. Equation 1 defines Hellinger Distance for discrete distributions, as a histogram with B bins.

$$\text{HD}(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i}^{B} \left(\sqrt{p_i} - \sqrt{q_i}\right)^2} \tag{1}$$

Although HD is a metric, the interpolation between H_+ and H_- used inside HDy is inadequate to the Hellinger space. Withal, we empirically confirmed that $HD(A + xB, C)$ is not unimodal. The authors of the algorithm suggest finding HDy with a linear search through α, after discretizing the interval $[0, 1]$. On the other hand, we suggest using Ternary Search even if the results could be sub-optimal, to speed-up the algorithm. We back this suggestion with a preliminarily step in our experimental evaluation.

There are two clear advantages in representing the distribution of the training data as a linear interpolation between two histograms. As scalable vectors in $\mathbb{R}^B$ represent the distributions, when we need to represent a more significant number of events for one of the classes, we do not need to duplicate individual events. In this way, all events still have the same weight. Similarly, when we need to represent a smaller number of events for the other class, we do not need to sample individual observations, and therefore, we do not discard useful information. Notably, when scaling down a class, if the number of available observations in the training set is low, discarding any information can be critical and further skew from the actual distribution of the data.

Some other forms of comparing distributions do not have such advantages, as the discrete computation of the Kolmogorov-Smirnov test, since it counts individual observations.

We can extend this approach by replacing the Hellinger Distance with any other dissimilarity measure, as Minkowski or Kullback Leibler. Cha [6] presents a comprehensive survey on similarity measures that suit probability distributions. However, we delegate the impact of such replacement to future work.

In the next section, we present and detail our proposed methods.

4 PROPOSED METHODS

In this section, we introduce two methods to identify which context a data sample belongs to, while simultaneously estimating its positive class ratio. Our proposals are suitable to work with binary classification and quantification problems.

We assume that the sample belongs entirely to only one context at a time. Although the algorithm receives a sample as input, this chunk can represent a sliding window in a data stream.

4.1 SMR-HDy

Our first algorithm, Single Most Relevant HDy (SMR-HDy) is a simple approach to the problem. Moreover, it is the base for the second algorithm, which we describe later on.

Consider the set $C = \{1, 2, \ldots, |C|\}$ of contexts. For each context $i \in C$, we have available training and validation sets of events $T_i \in T$ and $V_i \in V$, respectively. Each event in these sets is associated with a class label $y \in \{+, -\}$. Let $V_i^y \subseteq V_i$ be the subset of V_i that contains all examples from V_i labeled as y.

From the training sets T_i, we induce the classification models M_i. The algorithm follows by computing normalized (unit area) histograms $H_i^y, y \in \{+, -\}$ of the output scores obtained by M_i on the validation set V_i^y.

Finally, given an unlabeled test set of events U, we compute the normalized histogram H_i^U of the scores obtained by the classifier M_i on the test set U. We then consider the most probable context for U as:

$$\hat{c}_s = \text{SMR-HDy}(U) = \arg\min_{i \in C} \text{HD}\left(\alpha_i H_i^+ + (1 - \alpha_i) H_i^-, H_i^U\right)$$

$$\text{where} \qquad \alpha_i = \text{HDy}(H_i^+, H_i^-, H_i^U)$$

In other words, $\hat{c}_s$ is the context that minimizes the divergence between the distributions of scores obtained by the classifier $M_{\hat{c}_s}$ on the validation set $V_{\hat{c}_s}$ and on the unlabeled sample U. This minimization is found when we estimate the positive class proportion as $\alpha_{\hat{c}_s}$. The rationale behind SMR-HDy is that the stored histograms for the validation sets represent expected *behaviors* for the scores. For instance, H_1^+ represents the behavior we expect to find for the scores by applying M_1 on positive events from context 1. Finally, the HDy algorithm provides an interpolation parameter (in our case, the estimation for the positive class ratio) that minimizes the divergence between a test histogram and a linear interpolation of two training histograms. Additionally, as a by-product, we obtain a measurement for the divergence that can be used to compare different contexts.

We note that, in SMR-HDy, we expect the behavior of the scores produced by each model on events that belong only to its corresponding context. This is a simple approach, and we are in fact discarding useful information: the expected behavior of the scores obtained by the classifier M_i on the validation set V_j when $i \neq j$. We overcome this limitation in our second algorithm, presented in the next section.

4.2 XO-HDy

Our second proposal, the Crossed Opinions HDy (XO-HDy) takes into account how the scores for events from one context are expected to behave when obtained by a classifier of a different context.

Consider $H_{i,j}^y$ the normalized histogram of scores obtained by the classifier M_i on the validation set V_j^y. Also, consider $\alpha_{i,j}$ as follows:

$$\alpha_{i,j} = \text{HDy}(H_{i,j}^+, H_{i,j}^-, H_i^U)$$

We then consider the most probable context for U as:

$$\hat{c}_x = \text{XO-HDy}(U) = \arg\min_{j \in C} \frac{1}{|C|} \sum_{i \in C} \text{HD}\left(\alpha_{i,j} H_{i,j}^+ + (1 - \alpha_{i,j}) H_{i,j}^-, H_i^U\right)$$

Figure 3 provides a visual illustration of the sources of data inside the computation of each Hellinger Distance in XO-HDy.

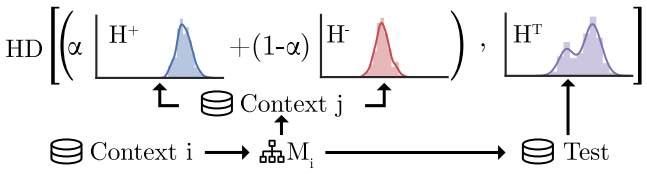

Figure 3: *Opinion* from model M_i regarding context j. If M_i applied on test data presents similar score behavior to any mixture of positive and negative scores of M_i applied on context j's validation data, then M_i votes for higher chance of actual context being j.

Our method provides $\alpha_{\hat{c}_x}$, an estimation of the positive class ratio, as a byproduct. However, once our method has identified the context, we can use any quantification algorithm in conjunction with the appropriate classifier, instead of HDy.

4.3 Classification Adjustment

Once we have both the inferred context $\hat{c}$ and an estimation $\hat{\alpha}$ for the positive class ratio, we can readjust the classifier induced upon $t_{\hat{c}}$ with a new threshold and reclassify the events in U. Precisely, we expect $\hat{\alpha}\%$ of the examples to belong to the positive class. Therefore, we set the classification threshold as the $(1 - \hat{\alpha})$-percentile of the obtained scores; that is, we consider the $\hat{\alpha}\%$ events with highest scores as positive.

5 EXPERIMENTAL SETUP

In this section, we describe the experimental setup and datasets adopted to evaluate our methods.

5.1 Algorithms and Evaluation

Although the proposed algorithms aim at identifying the correct context of a sample of unlabeled events, the reason behind doing so is the existence of underlying tasks which could potentially achieve better performance when considering the context information. Therefore, we not only measure the context identification accuracy, *i.e.*, how many times our proposed methods predict the correct context, but also the performance for the classification and quantification tasks when relying on such predictions.

We compare our algorithms' performances with two toplines, two baselines, and the Kolmogorov-Smirnov (KS) test. Their reasoning is as follows:

$\mathcal{T}_1$ (Topline 1): This algorithm always guesses the correct context. We expect it to perform as well or better than our proposals. We use this algorithm to obtain an upper limit for the underlying tasks (classification and quantification) when

we consider separate contexts and distinguish between them flawlessly;

$\mathcal{T}_2$ (Topline 2): In this case, the correct context is known and is used as an additional feature to describe each example. We induce only one classifier using all available training data, after including this additional feature. We believe that this is the way most practitioners would model their problems, in case they had access to the variables responsible for defining the contexts. If the first topline or the proposed approaches surpass this topline, we have evidence that it is better to induce individual models for each context instead of considering a single model with a context attribute in the example;

$\mathcal{B}_1$ (Baseline 1): Similarly to the second topline, this approach only induces one classifier with all available training data. However, this baseline disregards context information. If this baseline performs as the toplines, it indicates that there is no concept drift (or dataset shift) across different contexts. Otherwise we have evidence that the data distribution changes across contexts;

$\mathcal{B}_2$ (Baseline 2): This baseline guesses the context uniformly at random. Since in our experiments all contexts occur with equal chance, surpassing this baseline indicates that our proposals learned the problem.

KS (Kolmogorov-Smirnov): This is a non-parametric test to compare distributions and is a state-of-the-art approach to identify concept drifts [10, 18, 26, 30]. We compare the distributions of the scores output by each classifier and choose the one with the lowest statistic. As this method does not estimate the positive class ratio, we apply HDy after choosing the context.

Section 5.2 describes the six real datasets used in our experiments. We uniformly and randomly split each dataset into two disjoint halves. The first half is reserved for training and validation, while the second is used for testing. As all compared algorithms depend only on the scores output by classification models, we used 10-fold cross-validation to obtain the scores using the first half of the dataset. After obtaining all validation scores, we induced the classifiers in the first half (one for each context, plus one for $\mathcal{U}_2$ and one for $\mathcal{B}_1$) and obtained scores for all events in the second half.

For each context, in each dataset, we varied the positive class proportion from 0% to 100%, incrementing by 1% at a time. For each positive ratio, we created 10 random samples of W events. No event appears more than once in each sample, though it can appear on multiple samples. These were the samples used to test and measure the performance of the competing approaches. All-in-all, we evaluated $1000 \times |C|$ samples for each dataset, in which C is the set of contexts.

We also report the obtained quantification results for Adjusted Classify And Count (ACC) coupled with all competing approaches. For instance, ACC with XO-HDy uses the classifications from the classifier trained on the context predicted by XO-HDy, and ACC with $\mathcal{B}_\in$ uses the classifications from a single classifier without context information.

We measured the performance of the approaches regarding context identification accuracy (only for SMR-HDy and XO-HDy), the accuracy of the underlying classifier after choosing the context, and mean absolute error for the quantification task.

We merged all test samples into a single stream to analyze our approach in data streams scenarios. In this case, each context appears in the stream in the form of 10 non-consecutive sequences of instances. Each sequence is composed of the concatenation of 100 randomly positioned samples of size W, each sample having a single positive class ratio within this sequence. We compare XO-HDy against MINAS [9] regarding non-adjusted classification accuracy. We used the authors' freely available implementation of MINAS with default settings, except for the threshold for Novelty Detection. We raised this threshold so that MINAS detected no more than 50 events as novelty.

All classification models are Random Forests with 500 trees, using scikit-learn[1] with default parameters.

Since HDy is still a relatively new algorithm, there is no consensus regarding an optimal number B of bins for the histograms. The original authors of HDy did not suggest any specific value, though they ran all experiments with varying sizes from 10 to 100 and reported the median. We note that choosing B, although a non-trivial task, is not devoid of useful insights. Histograms with large values of B are negatively affected by two aspects. First, if W is not large enough, these histograms can become too sparse, each bin can have too low weight and, ultimately, the Hellinger Distance can face the curse of dimensionality. Second, a large number of bins has the strong assumption of high precision for the scores obtained with the classifiers. On the other hand, if B is too small, the Hellinger Distance can be unable to differentiate distributions. We opted for using $B = 11$ across all experiments, meaning that we allow for a precision of 0.1 for the scores (from 0 to 1.0). We delegate a better approach to address this parameter for future work.

Finally, to verify if we can safely adopt Ternary Search inside HDy, we simulated the same conditions as described above, and measured the difference between the positive class ratios found by HDy with linear search and with Ternary Search. For the linear search, we split the interval $[0, 1]$ into 1000 uniformly distributed parts. For the Ternary Search, we adopted an error of 10^{-5}.

For the sake of reproducibility and further usage of our ideas, we make available all the codes, datasets, and detailed results at our supplementary material repository[2].

The next section describes each dataset used in our experimental setup.

5.2 Datasets Description

This section describes each dataset used to benchmark our proposals. As the datasets have a limited number of events, we chose W (size of the test sample) so that there would be enough examples to vary the probability of the positive class from 0 to 1, and still have variability across samples in the extremes of this range. The datasets are:

Aedes-Culex The dataset *Aedes aegypti-Culex quinquefasciatus* [10] contains 24,000 events described by features that

[1] scikit-learn v0.19.0 available at http://scikit-learn.org/

[2] https://github.com/denismr/Classification-and-Counting-with-Recurrent-Contexts

register the passage of female *Aedes aegypti* and female *Culex quinquefasciatus* mosquitoes in front of a sensor. Besides wing beat frequency, there are other 25 numerical features obtained from the signal power spectrum. Six ranges of temperature define the contexts, and the objective is to distinguish between the two species. $W = 300$;

Aedes-Sex The dataset *Aedes aegypti*-sex [10], also containing 24,000 entries, is similar to the previous one. However, we want to discriminate between male and female *Aedes aegypti* mosquitoes. $W = 300$;

Arabic-Digit A modified version of the Arabic-Digit dataset [17, 20], which contains 8,800 entries described by a fixed number of MFCC values for the human speech of Arabic digits (among 10). The spoken digit defines the context, and the task is to predict the sex of the speaker. $W = 200$;

CMC CMC [20] is a dataset from a survey on contraceptive prevalence at Indonesia. It contains 1,473 entries described by eight features. The task is to predict whether a contraceptive is in use or not, and the concept is given by two ranges of age. $W = 100$. As MINAS requires purely numerical datasets, we excluded this dataset in its evaluation;

Handwritten-QG QG is a subset of the dataset Handwritten [10], which contains 4,010 entries described by 63 numerical features regarding the handwritten letters *g*, and *q*. The context is given by the author (among 10), and the objective is to predict the letter. $W = 80$;

Wine Wine Quality [7, 20] contains 6,498 entries described by 11 features. The task is to differentiate between two ranges of quality of wines. The context is given by the type of the wine (red or white). $W = 300$.

6 EXPERIMENTAL EVALUATION

In this section, we present and discuss our experimental results.

We open this section by displaying evidence that supports using Ternary Search inside the HDy algorithm. Table 1 presents the average absolute differences between the positive ratios found when applying discretized linear search and Ternary Search inside HDy. The most significant average difference occurs for dataset CMC, and this value represents a single instance every 2,645 instances. Interestingly, we note these differences are for estimated $\arg\min \mathrm{HD}(x)$, and when we map these values to $\mathrm{HD}(x)$, Ternary Search produces smaller distances for most of the cases (column $<$ HD).

We also assess the accuracy of the context identification. Table 2 presents the results for mean accuracy for all class distributions. Our methods performed consistently above chance level ($\mathcal{B}_2$), and XO-HDy consistently outperformed both SMR-HDy and KS. We note that XO-HDy achieved perfect context identification in three out of six datasets, almost perfect in a fourth, and the lowest identification accuracy was 88.73%.

SMR outperformed KS in all dataset but CMC. KS performs well mainly when the positive class ratio is similar to that in the training set, and poorly otherwise. Figure 4 illustrates the context identification accuracy for each positive class proportion on dataset Aedes-Culex. Other datasets have similar performance (shown in the supplemental material).

Table 1: Average absolute difference ($\overline{\Delta x}$) between the positive ratios found by discretized linear search and by Ternary Search, in the HDy algorighm. Column # represents the number of trials and column s presents the standard deviation. The last column presents the proportion of cases in which the Ternary Search led to a lower Hellinger Distance.

Dataset	#	$\overline{\Delta x}$	s	$<$ HD
Aedes-Culex	387840	1.78×10^{-4}	1.65×10^{-4}	70.49%
Aedes-Sex	387840	2.47×10^{-4}	1.46×10^{-4}	98.45%
Arabic-Digit	1454400	2.44×10^{-4}	1.49×10^{-4}	97.21%
CMC	32320	1.75×10^{-4}	1.65×10^{-4}	68.95%
Handwritten-QG	1454400	3.78×10^{-4}	6.50×10^{-3}	84.64%
Wine	32320	1.89×10^{-4}	1.64×10^{-4}	75.17%

Table 2: Mean Context identification accuracy comparison for a uniformly random test ($\mathcal{B}_2$), Kolmogorov-Smirnov test (KS) and our two proposals.

Dataset	$\mathcal{B}_2$	KS	SMR	XO-HDy
Aedes-Culex	16.95%	46.07%	61.57%	88.73%
Aedes-Sex	16.95%	39.65%	65.66%	99.77%
Arabic-Digit	10.50%	66.87%	99.59%	100.00%
CMC	51.34%	80.59%	71.19%	94.21%
Handwritten-QG	10.08%	80.78%	88.59%	100.00%
Wine	48.71%	90.00%	100.00%	100.00%

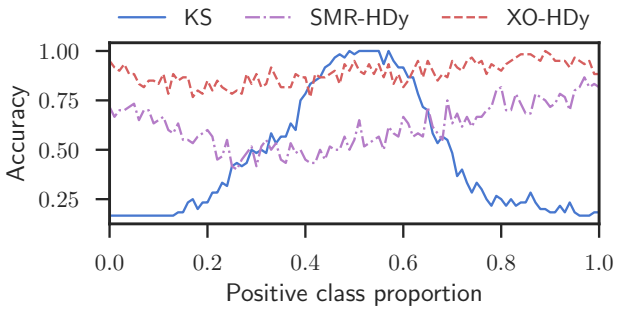

Figure 4: Context identification accuracy for all positive class proportions for dataset Aedes-Culex.

In the remainder of this section, we present the results for the underlying tasks, that is, quantification and classification.

6.1 Quantification

Table 3 presents the mean absolute error (MAE) between the predicted and actual positive ratios obtained with HDy. The mean values are computed across all positive distributions.

For quantification tasks, XO-HDy performed best among all methods. The fact that XO-HDy performed better than $\mathcal{B}_1$ highlights the importance of differentiating contexts. Similarly, the fact it also outperformed $\mathcal{B}_2$ indicates the relevance of using the *correct* context. In fact, $\mathcal{B}_2$ performed consistently worse than $\mathcal{B}_1$. This further supports the previous point, and more importantly, it also

Table 3: HDy absolute quantification error averaged over all positive proportions.

Dataset	$\mathcal{B}_1$	$\mathcal{B}_2$	KS	SMR	XO	$\mathcal{T}_1$	$\mathcal{T}_2$
Aedes-Culex	27.04% (20.29)	28.23% (26.32)	19.65% (20.97)	17.87% (24.96)	6.63% (14.66)	2.20% (1.78)	5.95% (4.18)
Aedes-Sex	4.14% (4.64)	9.38% (16.01)	7.36% (16.13)	0.62% (0.75)	0.44% (0.40)	0.44% (0.40)	1.00% (1.15)
Arabic-Digit	1.13% (0.92)	19.96% (19.36)	6.19% (12.00)	0.71% (0.64)	0.70% (0.63)	0.70% (0.63)	0.97% (0.80)
CMC	25.24% (14.98)	29.15% (26.24)	23.10% (22.90)	27.60% (28.70)	16.71% (18.21)	12.73% (10.77)	10.69% (8.60)
Handwritten-QG	1.96% (3.17)	22.87% (25.34)	1.47% (5.71)	0.82% (0.70)	0.80% (0.66)	0.80% (0.66)	1.90% (2.93)
Wine	13.04% (10.65)	26.14% (28.83)	10.08% (18.76)	3.94% (2.66)	3.94% (2.66)	3.94% (2.66)	13.28% (11.50)

clarifies that it is better to ignore context information rather than using it if the context information is unreliable.

As XO-HDy obtained perfect context identification in three datasets, and almost perfect in a fourth dataset, it performed as well as the topline $\mathcal{T}_1$ in these cases. However, XO-HDy performed worse then $\mathcal{T}_1$ for the other two datasets. Although a paired t-test between XO-HDy and $\mathcal{T}_1$ results in a p-value of 17.55%, we believe that this p-value would decrease as we increase our sample size, as we expect $\mathcal{T}_1$ to be an upper bound of the performance obtainable by XO-HDy. A paired t-test between XO-HDy and $\mathcal{T}_2$ resulted in a p-value of 72.14%, which does not indicate a statistically significant difference between the methods. An important observation, though, is that XO-HDy only underperformed $\mathcal{T}_2$ when it did not achieve almost perfect context identification. Finally, $\mathcal{T}_1$ only underperformed $\mathcal{T}_2$ in one dataset. This indicates that in most of our cases, separating the contexts is a better approach than including it as a feature.

In general, our results suggest that in many cases, we can confidently use XO-HDy in quantification tasks without context information, and expect the same results as those obtained if context information were available. In other cases, although XO-HDy performs worse than when context information is available, it still outperforms approaches that simply ignore contexts or randomly guess the current context (as done by $\mathcal{B}_1$ and $\mathcal{B}_2$).

Table 4 presents the quantification errors when using Adjusted Classify and Count (ACC) instead of HDy. We did not observe significant differences between HDy and ACC. Indeed, ACC outperforms HDy in specific datasets; however, it still benefits from context identification the same way HDy does and relies on HDy to this task.

6.2 Classification

Tables 5 and 6 present non-adjusted and adjusted classification accuracies, respectively. The observations regarding $\mathcal{B}_2$ are the same as the ones made for quantification. All p-values reported are according to the paired t-test.

We cannot statistically infer that XO-HDy outperforms $\mathcal{B}_1$ (p-value = 32.29%). However, when we compare the adjusted XO-HDy against the best results between non-adjusted and adjusted $\mathcal{B}_1$, XO-HDy outperforms $\mathcal{B}_1$ in every single dataset. Comparing adjusted XO against the best between adjusted and non-adjusted $\mathcal{T}_1$ and $\mathcal{T}_2$ lead to the same conclusions than the ones obtained from the quantification results.

Figure 5 illustrates the average accuracy obtained by both adjusted and non-adjusted XO-HDy in dataset Aedes-Culex, for every positive class ratio tested. From this figure, we can see that it is possible to adjust a classifier only when the quantification is within specific ranges of the positive class proportion, and improve classification accuracy even further. This pattern was common across all datasets. We believe that, while adjustment increases performance for samples that have class proportions different than the one found in the training set (which in this case is balanced), it may have the opposite effect otherwise. We delegate further investigation on this topic to future work.

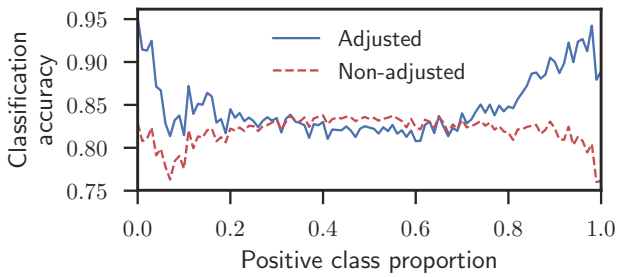

Figure 5: Comparison between adjusted and non-adjusted classification performance across different class proportions. Data obtained with method XO-HDy on dataset Aedes-Culex.

6.3 Data Stream Classification

Table 7 presents the accuracies obtained by XO-HDy and MINAS [9] in a streaming setting. Although we fed both with the same training data, MINAS was unable to discern the contexts and, judging by its inferior performance compared to B_1, its forgetting mechanism was harmful. XO-HDy consistently outperformed MINAS. Our results suggest that when contexts are known, switching between them is better than adapting the models after tracking incremental drifts.

7 TIME AND MEMORY COMPLEXITIES

We can split the computational cost into two parts: training and deployment. The training phase is performed once, and the deployment phase is recurrent for every data chunk.

Consider $\mathcal{T}$ the cost in time to train a classifier, $\mathcal{M}$ the cost in memory to store a classification model, and $\mathcal{L}$ the cost in time to issue a classification for one event. Notice that $\mathcal{T}$, $\mathcal{M}$, and $\mathcal{L}$ depends on the adopted classification algorithm. Assitionally, consider $\mathcal{V}$ the size of the validation set for each context and B the number of bins of the histograms.

Table 4: ACC absolute quantification error averaged over all positive proportions.

Dataset	$\mathcal{B}_1$	$\mathcal{B}_2$	KS	SMR	XO	$\mathcal{T}_1$	$\mathcal{T}_2$
Aedes-Culex	23.09% (18.25)	28.53% (26.16)	19.54% (20.57)	18.19% (24.83)	7.14% (14.59)	2.84% (2.21)	5.84% (3.72)
Aedes-Sex	3.24% (4.67)	8.44% (14.99)	7.46% (15.00)	0.65% (0.72)	0.52% (0.45)	0.52% (0.45)	0.84% (0.64)
Arabic-Digit	1.00% (0.69)	12.54% (14.01)	6.51% (11.93)	0.59% (0.53)	0.59% (0.53)	0.59% (0.53)	1.00% (0.69)
CMC	22.55% (17.43)	31.12% (27.32)	26.53% (26.30)	29.02% (29.19)	19.10% (20.51)	14.88% (13.72)	16.20% (12.10)
Handwritten-QG	1.18% (1.67)	11.15% (20.27)	1.04% (4.96)	0.61% (0.49)	0.60% (0.43)	0.60% (0.43)	1.18% (1.67)
Wine	13.56% (10.70)	18.95% (24.60)	7.37% (6.24)	5.74% (3.53)	5.74% (3.53)	5.74% (3.53)	13.22% (10.45)

Table 5: Classification accuracy averaged for all positive class proportions.

Dataset	$\mathcal{B}_1$	$\mathcal{B}_2$	KS	SMR	XO	$\mathcal{T}_1$	$\mathcal{T}_2$
Aedes-Culex	76.84% (14.42)	72.32% (19.87)	74.40% (13.66)	76.37% (15.66)	82.02% (9.77)	84.01% (4.96)	83.94% (5.39)
Aedes-Sex	96.14% (4.87)	89.25% (16.69)	91.66% (15.21)	98.58% (1.14)	98.79% (1.02)	98.79% (1.02)	98.51% (1.11)
Arabic-Digit	98.96% (1.01)	84.54% (14.22)	93.30% (12.07)	99.48% (0.64)	99.48% (0.64)	99.48% (0.64)	98.89% (0.98)
CMC	66.07% (6.74)	62.96% (10.66)	61.72% (9.94)	61.71% (9.63)	64.03% (7.74)	65.45% (5.20)	66.51% (5.47)
Handwritten-QG	99.20% (1.82)	88.04% (20.97)	99.26% (5.03)	99.71% (0.62)	99.71% (0.57)	99.71% (0.57)	99.20% (1.82)
Wine	78.30% (5.51)	69.42% (16.37)	77.75% (6.12)	78.18% (5.36)	78.18% (5.36)	78.18% (5.36)	78.62% (5.43)

Table 6: Mean classification accuracy for all positive class proportions with decision thresold ajusted according to quantifications results of Table 3.

Dataset	$\mathcal{B}_1$	$\mathcal{B}_2$	KS	SMR	XO	$\mathcal{T}_1$	$\mathcal{T}_2$
Aedes-Culex	70.87% (18.98)	68.95% (24.28)	73.51% (16.85)	75.97% (22.02)	84.70% (13.54)	88.07% (5.83)	87.26% (5.83)
Aedes-Sex	95.50% (4.66)	88.77% (16.54)	91.81% (16.18)	98.68% (1.15)	98.90% (0.94)	98.90% (0.94)	98.53% (1.37)
Arabic-Digit	98.73% (0.98)	78.80% (19.18)	93.68% (12.01)	99.23% (0.68)	99.24% (0.67)	99.24% (0.67)	98.88% (0.89)
CMC	68.15% (15.52)	62.92% (22.66)	64.14% (19.14)	61.79% (23.70)	68.70% (16.14)	72.32% (11.99)	73.31% (11.80)
Handwritten-QG	98.29% (3.15)	76.38% (25.71)	98.68% (5.77)	99.41% (0.85)	99.44% (0.83)	99.44% (0.83)	98.37% (2.94)
Wine	80.83% (9.37)	68.45% (25.39)	79.25% (16.38)	85.10% (6.18)	85.09% (6.18)	85.10% (6.18)	80.48% (9.76)

Table 7: Comparison of classification accuracy between MINAS [9] and XO-HDy in data stream setting.

Dataset	MINAS	XO-HDy
Aedes-Culex	58.30%	82.91%
Aedes-Sex	91.91%	98.68%
Arabic-Digit	56.97%	99.43%
Handwritten-QG	74.57%	99.61%
Wine	48.20%	77.49%

The cost in time to train the SMR-HDy is $O(|C|\mathcal{T}+|C|\mathcal{L}\mathcal{V})$, while the cost to keep the SMR-HDy in memory after the training phase is $O(|C|\mathcal{M} + |C|B)$. The training time cost of XO-HDy is $O(|C|\mathcal{T} + |C|^2\mathcal{L}\mathcal{V})$, while its post-training memory cost is $O(|C|\mathcal{M} + |C|^2B)$.

From a deployment perspective, the cost in time to use both SMR-HDy and XO-HDy depends on which search strategy we use for minimizing $\text{HD}(A + xB, C)$ inside the HDy algorithm. Although the original authors of the algorithm reckon that many strategies can be applied, they suggest using linear search. In this case, the domain of the function ($[0, 1]$) is discretized in P parts, and the cost in time of estimating its minimum is $O(PB)$. On the other hand, we advocate applying Ternary Search, even though the function to be minimized is not unimodal. Our experimental results show that the results are essentially the same as the ones obtained by the linear search, though the computational cost becomes a substantially lower $O(Bk_\epsilon)$, where k_ϵ grows logarithmically on the inverse of the error ϵ, which is a parameter of the Ternary Search. For instance, $k_\epsilon = 24$ when $\epsilon = 10^{-4}$, and $k_\epsilon = 30$ when $\epsilon = 10^{-5}$.

Given W, the number of events in the test set U. The computational time cost of running SMR-HDy depends on the time required to compute the test histograms H_i^X ($O(|C|\mathcal{L}W)$), and the time required to identify the current context ($O(|C|Bk_\epsilon)$). In total, it is $O(|C|\mathcal{L}W + |C|Bk_\epsilon)$.

Similarly, the cost of running XO-HDy is composed of the time required to compute the histograms ($O(|C|CW)$), and the time required to identify the current context ($O(|C|^2Bk_\epsilon)$). In total, it is $O(|C|\mathcal{L}W + |C|^2Bk_\epsilon)$.

We note that if the test set U comprises a sliding window in a data stream, every time we observe an event, the histograms are updated in $|C|\mathcal{L}$. In this case, it is unnecessary to recompute the histograms entirely. We note, however, that it increases the memory cost by $O(W|C|)$, given the need for a queue to remove old values from the histograms in the correct order.

Additionally, it is unnecessary to keep all W events from the test chunk U in memory, simultaneously. Each event can be processed and promptly discarded, one at a time.

8 LIMITATIONS

Our proposed methods have three significant limitations. The first is the fact that they are suitable only for binary classification problems. This same assumption is present in most papers on quantification. We plan to remove such limitation in future work, as we discuss in the next section.

The second limitation is the assumption of purity of context, *i.e.*, adjacent instances belong to the same context. Our experimental evaluation reflects this assumption as, except when comparing our approaches against MINAS, we measure performance on batches that contain data from only one context at a time. Indeed, we expect more than one context to coexist within a same sliding window, in a data stream, during the transition between two consecutive contexts.

However, two essential aspects soften this limitation. First, the period of coexistence is as long as the size of the sliding window. We expect the window to contain much fewer instances than the number of observed examples associated with the active context. Therefore, this assumption is not likely to affect the measured performance significantly. Second, for some applications, it is reasonable to assume that consecutive contexts share enough similarities so that applying either model would lead to acceptable performance. In that sense, even during transitions, the mixture of contexts would not pose an issue to the performance.

The third and final limitation is the assumption that we previously know all contexts. One suggestion to circumvent this limitation would be to set a threshold for the output of our approaches: values above the threshold would indicate novelty, and labels for recently observed events could be requested to form a new context.

9 CONCLUSIONS AND FUTURE WORK PROSPECTS

We presented XO-HDy, a method that accurately identifies the context of data samples, while simultaneously estimating their positive class ratios. We have empirically shown that assertively predicting a data's context lead to lower quantification errors and higher classification accuracy than: **(1)** ignoring the existence of different contexts altogether; and **(2)** inaccurately guessing the context. Additionally, our method is very efficient regarding memory and time consumption and does not impose restrictions on the number of contexts in the data.

XO-HDy is limited to binary classification. As future work prospects, we aim at extending the presented concepts to multi-class problems. One possible way is to aggregate the results from one-vs-rest [23]. Additionally, we will evolve our approaches so that we can identify novel contexts that were so far unknown to the model. A possibility is employing Transfer Learning techniques on the most similar known context.

ACKNOWLEDGMENTS

The authors would like to thank CAPES (grant PROEX-6909543/D), CNPq (grant 306631/2016-4), FAPESP (grant 2016/04986-6). This material is based upon work supported by the United States Agency for International Development under Grant No AID-OAA-F-16-00072.

REFERENCES

[1] Reem Al-Otaibi, Ricardo BC Prudêncio, Meelis Kull, and Peter Flach. 2015. Versatile Decision Trees for Learning Over Multiple Contexts. In *ECML/PKDD*. 184–199.
[2] Mohd Amir and Durga Toshniwal. 2010. Instance-Based Classification of Streaming Data Using Emerging Patterns. In *ICT*. 228–236.
[3] A. Bifet and R. Gavaldà. 2007. Learning from time-changing data with adaptive windowing. In *SDM*. 443–448.
[4] Albert Bifet and Ricard Gavaldà. 2009. Adaptive learning from evolving data streams. In *IDA*. 249–260.
[5] Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. 2010. Leveraging bagging for evolving data streams. In *ECML/PKDD*. 135–150.
[6] Sung-Hyuk Cha. 2007. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences* 1, 4 (2007), 300–307.
[7] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. 2009. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems* 47, 4 (2009), 547–553.
[8] Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. 2015. Credit Card Fraud Detection and Concept-Drift Adaptation with Delayed Supervised Information. In *IJCNN*.
[9] Elaine Ribeiro de Faria, André Carlos Ponce de Leon Ferreira, João Gama, et al. 2016. MINAS: multiclass learning algorithm for novelty detection in data streams. *Data Mining and Knowledge Discovery* 30, 3 (2016), 640–680.
[10] Denis dos Reis, André Maletzke, and Gustavo Batista. 2018. Unsupervised Context Switch for Classification Tasks on Data Streams with Recurrent Concepts. In *ACM/SIGAPP*, Vol. 33. ACM.
[11] Karl B Dyer, Robert Capo, and Robi Polikar. 2013. COMPOSE: A Semisupervised Learning Framework for Initially Labeled Nonstationary Streaming Data. *TNNLS* (2013).
[12] George Forman. 2005. Despite Inaccurate Classification. In *ECML*. Porto, 564–575.
[13] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. 2004. Learning with drift detection. In *SBIA*. 286–295.
[14] Wei Gao and Fabrizio Sebastiani. 2016. From classification to quantification in tweet sentiment analysis. *Soc Netw Anal Min* 6, 1 (2016). https://doi.org/10.1007/s13278-016-0327-z
[15] Pablo González, Alberto Castaño, Nitesh V Chawla, and Juan José Del Coz. 2017. A review on quantification learning. *ACM Computing Surveys (CSUR)* 50, 5 (2017), 74.
[16] Víctor González-Castro, Rocío Alaiz-Rodríguez, and Enrique Alegre. 2013. Class distribution estimation based on the Hellinger distance. *Information Sciences* 218 (2013), 146–164.
[17] Nacereddine Hammami and Mouldi Bedda. 2010. Improved tree model for arabic speech recognition. In *ICCSIT*, Vol. 5. 521–526.
[18] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. 2004. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 180–191.
[19] Ludmila I Kuncheva et al. 2008. Nearest neighbour classifiers for streaming data with delayed labelling. In *ICDM*. IEEE, 869–874.
[20] M. Lichman. 2013. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml.
[21] Mohammad M Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani Thuraisingham. 2011. Classification and novel class detection in concept-drifting data streams under time constraints. *TKDE* 23, 6 (2011), 859–874.
[22] Mohammad M Masud, Clay Woolam, Jing Gao, Latifur Khan, Jiawei Han, Kevin W Hamlen, and Nikunj C Oza. 2012. Facing the reality of data stream classification: coping with scarcity of labeled data. *KAIS* 33, 1 (2012), 213–244.
[23] Nasser M Nasrabadi. 2007. Pattern recognition and machine learning. *Journal of electronic imaging* 16, 4 (2007), 049901.
[24] Nikunj C Oza. 2005. Online bagging and boosting. In *SMC*, Vol. 3. IEEE, 2340–2345.
[25] David Pollard. 2002. *A user's guide to measure theoretic probability*. Vol. 8. Cambridge University Press.
[26] Denis Reis, Peter Flach, Stan Matwin, and Gustavo Batista. 2016. Fast Unsupervised Online Drift Detection Using Incremental Kolmogorov-Smirnov Test. In *ACM SIGKDD*, Vol. 22. ACM.
[27] Diego F. Silva, Vinícius M. A. Souza, Daniel Ellis, Eamonn Keogh, and Gustavo E.A.P.A. Batista. 2015. Exploring Low Cost Laser Sensors to Identify Flying Insect Species. *J Intell Robot Syst* 80, 1 (2015), 313–330.
[28] Vinícius MA Souza, Diego F Silva, João Gama, and Gustavo EAPA Batista. 2015. Data Stream Classification Guided by Clustering on Nonstationary Environments and Extreme Verification Latency. In *SDM*. SIAM, 873–881.
[29] Xindong Wu, Peipei Li, and Xuegang Hu. 2012. Learning from concept drifting data streams with unlabeled data. *Neurocomputing* 92 (2012), 145–155.
[30] Indrė Žliobaitė. 2010. Change with Delayed Labeling: when is it detectable?. In *ICDMW*. IEEE, 843–850.