# Classification of evolving data streams with infinitely delayed labels

2015 IEEE 14th International Conference on Machine Learning and Applications

# Classification of Evolving Data Streams with Infinitely Delayed Labels

Vinicius M. A. Souza
University of São Paulo
vsouza@icmc.usp.br

Diego F. Silva
University of São Paulo
diegofsilva@icmc.usp.br

Gustavo E. A. P. A. Batista
University of São Paulo
gbatista@icmc.usp.br

João Gama
University of Porto
jgama@fep.up.pt

***Abstract*—The majority of evolving data streams classification algorithms assume that the actual labels of the predicted examples are readily available without any time delay just after a prediction is made. However, given the high label costs, dependence of an expert, limitations in data transmission or even restrictions imposed by the problem's nature, there is a large number of real-world applications in which the availability of actual labels is infinitely delayed (never available). In these cases, it is necessary the use of algorithms that does not follow the traditional process of monitoring the error rate to detect changes in data distribution and uses the most recent labeled data to update the classification model. In this paper, we propose the method *MClassification* to classify evolving data streams with infinitely delayed labels. Our method is inspired on the use of Micro-Cluster representation from online clustering algorithms. Considering the presence of incremental drifts, our approach uses a distance-based strategy to maintain the Micro-Clusters' positions updated. An evaluation in several synthetic and real data shows that *MClassification* achieves competitive accuracy results to state-of-the-art methods and adequate computational cost. The main advantage of the proposed method is the absence of critical parameters that require user's prior knowledge, as occurs with rival methods.**

## I. Introduction

Data streams are continuous and unbounded evolving data that arrives over time. This type of data is increasingly present in real world applications, mainly due to the growing popularity of sensors and portable measurement devices. For instance, people carrying smartphones are able to produce massive stream data daily. Other examples of stream applications are network traffic analysis, text mining on daily newspapers, analysis of bank transactions, monitoring audio and video data for surveillance, analysis of RFID tracking logs, etc.

In non-stationary environments, static models are rapidly outdated due to changes in data. Thus, in order to maintain a stable accuracy, the algorithms need to constantly update their classification models. Most of literature considers that the actual label $y_t$ of every predicted example $(\vec{x}_t, \hat{y}_t)$ received in a time $t$ of the stream, is readily available without any time delay immediately before the arrival of $\vec{x}_{t+1}$ [1]. The immediate availability of true labels allows the monitoring of distributions or error rate to verify whether the current classifier is outdated and use the most recent labeled data to update the classification model. A general view of this process is presented in Fig. 1.

However, the assumption that delay time will be equal to zero for actual labels is very optimistic and cannot be fulfilled in many situations. Given high label costs, dependence of an expert, limitations/failures in data transmission or even the impossibility to obtain these labels, it is more reasonable

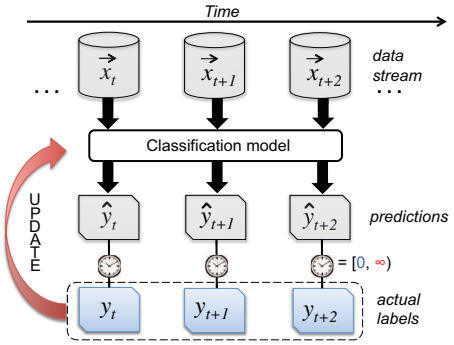



Fig. 1. A general view of the evolving data streams classification loop. Most of the algorithms in literature consider that the time delay to obtain the actual labels for each example is equal to zero

consider that this delay could vary from 0 to $\infty$. When the delay is an intermediate value, the process presented in Fig. 1 remains the same with a proportional lag for the identification of changes in data. In contrast, when the availability of actual labels is infinitely delayed it is impossible to detect changes using drift detectors and rely on the most recent labeled data to update the classification model. It is a non-trivial problem that was recently pointed out in SIGKDD *Explor.* as one of eight open challenges in data stream mining [2].

Examples of real applications that frequently have delayed the availability of actual labels are credit fraud, intrusion, and fault detection. Applications related to sensors and automation frequently have infinitely delayed labels. In sensors, an application with infinitely delayed labels is a laser that captures information about insects' wing beat when it passes the laser. The goal is to classify the signals into the insect species using audio features [3]. However, as pointed by [4] in a streaming scenario, meteorological features such as temperature and humidity are responsible for changing the insect's behavior and consequently the measured data over time. Moreover, obtaining the actual label of an insect that crossed the laser involves having an expert available in the field. Thus, we need to update the classification model over time without the actual labels of processed examples. In automation, the recent popularization of drones opens new challenges to the computerized automation of flights of these aerial vehicles. Given a drone initially trained in a known environment, they need to incrementally adapt to changes in speed and direction of the wind, altitude, temperature, and atmospheric pressure in an unsupervised way.

978-1-5090-0287-0/15 $31.00 © 2015 IEEE
DOI 10.1109/ICMLA.2015.174

There is a limited number of research papers in the literature that consider infinitely delayed labels. Some of the existing algorithms have a high computational cost, making their use not suitable for data streams. Other approaches have parameters in which the optimal values require tuning.

In this paper, we propose a method based on the use of Micro-Clusters for the classification of evolving data streams in a scenario where the availability of actual labels is infinitely delayed. Micro-Cluster is a well-known representation in online clustering that efficiently stores a set of similar data points. We propose a distance based strategy to maintain the positions of Micro-Clusters that are used to classify stream data with incremental drifts. Considering an evaluation on 16 synthetic datasets and 2 real-world problems, our algorithm achieves competitive accuracy to state-of-the-art methods with very good computational cost. The main advantage of our method is the absence of critical parameters that require user's prior knowledge for adjusting them as occurs with rival methods.

The remaining of the paper is organized as follows. Section II discusses the problem of evolving data in stream learning. Section III presents related works. Our proposed method is presented in Section IV and the experimental evaluation is shown in Section V. Finally, Section VI presents our conclusions and directions for future work.

## II. Evolving Data Streams

Due to non-stationarity of real environments, data streams suffer with changes in data distributions over time. In the context of machine learning, a *concept* can be defined as a set of instances generated by the same underlying function. When this function changes for some reason, we have the phenomenon of *concept drift* [5]. Given that $P$ is the probability of an event, $x$ is a feature vector and $y$ is a class label, the causes of concept drift can be [6]:

- **Feature change:** a change in the probability of the occurrence of a particular set of features values, i.e., a change in $P(x)$ but $P(y|x)$ remains the same. In other words, some previously infrequent feature values become more frequent and vice versa;
- **Conditional change:** a change in the conditional probability $P(y|x)$ to assign class label $y$ to a feature vector $x$, i.e., $P(x)$ remains the same but $P(y|x)$ changes;
- **Dual change:** a feature and conditional change, i.e., a change in both $P(x)$ and $P(y|x)$.

The cases when $P(x)$ changes while $P(y|x)$ does not change are referred as *virtual drift* [7]. Opposite cases, when $P(y|x)$ change while there are no changes in $P(x)$, occur due to changes in *hidden context*. Hidden context is the information that is not included into observable predictive features, but relevant in determining the class label [8]. According to [9], in scenarios where the actual labels are not available and $P(y|x)$ changes without changes in $P(x)$, the algorithm can identify them only by observing additional external features. In this work, we assume that we have access to all features responsible to change the concepts. Thus, we do not consider changes in $P(y|x)$ when changes are not observable in $P(x)$.

The changes in the underlying distributions may appear in different ways, such as abrupt, incremental, gradual or reoccurring. Fig. 2 illustrates the different patterns of drifts in a one-dimensional data where their mean evolves over time.

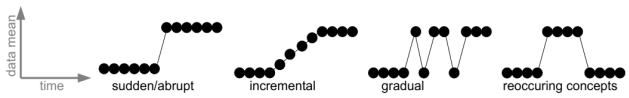



Fig. 2. Different patterns of changes over time [1]. The focus in this paper is on *incremental drifts*

In this paper, we deal with incremental drifts, where there are many intermediate concepts between one concept and another. Even in scenarios where there is knowledge of actual labels, this type of drift is often considered more challenging to detect than abrupt or gradual, given the significant overlap between concepts in a short period of time [10]. However, incremental drifts are important when labels are unavailable since we can use the similarities between two consecutive concepts to update the classification models.

## III. Related Work

We are aware of just three algorithms which consider infinitely delayed labels (this scenario is also called *initially labeled streaming environment* or *extreme verification latency*): Arbitrary Sub-Populations Tracker – APT [11], Compacted Object Sample Extraction – COMPOSE [10] and Stream Classification AlgoRithm Guided by Clustering – SCARGC [12]. In this section, we discuss how these three algorithms work.

APT considers that each class can be represented as a mixture of arbitrarily distributed sub-populations. The algorithm learns in a two-stage strategy: $i$) it uses expectation maximization to determine the optimal one-to-one assignment between the unlabeled and labeled data; and $ii$) it updates the classifier to reflect the population parameters of newly received data and the drift parameters, relating the previous time step to the current one. The algorithm has the assumption that the number of instances of each sub-population is constant over time. This limitation is hard to be fulfilled and may degrade the method after some iterations. Furthermore, the tests performed in [10] indicate a considerable high computational cost.

COMPOSE uses a generalization of convex hull called $\alpha$-shapes to deal with incremental drifts and infinitely delayed labels. The $\alpha$-shapes are geometrical constructs obtained from data which are then compacted (shrunk) to determine those current data points that are most likely to represent the distribution at the next time step. These instances are called *core supports* and serve as labeled instances to be propagated on next time step using semi-supervised algorithms that classify new unlabeled data in a transductive way.

The main weakness of COMPOSE is the high computational cost for the $\alpha$-shape construction, mainly with high dimensional data. Furthermore, the algorithm has two critical parameters that affect its performance: $\alpha$ and $CP$. The $\alpha$ parameter is related to the level of detail to model the classes' shapes. For a large value of $\alpha$ the resultant shape is the convex hull of the points and for a small value of $\alpha$ the resultant shape may become concave or even contain disconnected regions. The $CP$ parameter is related to the level of compaction applied

to the $\alpha$-shapes. If $\alpha$-shapes are too compact, relevant instances of the future distribution are lost. In contrast, if the $\alpha$-shapes are little compacted, instances of rival classes may overlap.

Souza et al. [12] presents a simple and efficient algorithm. SCARGC consists of a clustering followed by a classification step repeatedly applied in a closed loop fashion. During the classification phase, the algorithm also stores the examples in a *pool*. When the pool is full according to a determined value, the pool data are clustered into $k$ clusters. Thus, current and past cluster positions are compared to verify whether there are any changes in data distribution. If there is an indication of a significant difference, the data of current clusters receive the label from the most similar previous cluster. In this sense, the clustering step allows to track the drifting classes over time.

Due to its simplicity, SCARGC is very efficient in time. However, the algorithm is highly dependent of the clustering phase. The original proposal uses the $k$-means algorithm with a fixed $k$ over time. The main weakness of this approach is the assumption that the number of groups is constant over time as well as the need of user's knowledge to set this parameter appropriately. Another parameter of the algorithm is related to the pool size. The main influence of this parameter is in determining when verifying whether the current concepts are dissimilar to the previous concepts. However, the authors showed experimentally that in general, the use of small values for pool size provides good results.

In general, the main weaknesses of these methods are related to difficulties to correctly set their parameters (COMPOSE and SCARGC) and high computational costs in some situations (APT and COMPOSE). In this paper, we present a novel solution with a single parameter, practicable computational cost and competitive accuracy to other methods.

## IV. Our Proposal: Micro-Clusters for Classification - MClassification

In this paper we propose the algorithm *Micro-Clusters for Classification – MClassification* to deal with incremental evolving data streams in the challenging scenario where the actual labels of processed examples are infinitely delayed (never available) after their prediction. This is a more realistic scenario for a variety of real stream applications.

To adapt to changes over time, our algorithm uses the concept of Cluster Feature originally introduced by [13] in the BIRCH algorithm and then called Micro-Cluster by [14] in the CluStream algorithm. Before presenting our solution, we introduce the main concepts of Micro-Cluster.

Micro-Cluster (MC) is a compact representation that uses the triple $(N, \overrightarrow{LS}, \overrightarrow{SS})$ to store sufficient statistics from a set of examples, where:

- $N$ is the number of data points in a cluster;
- $\overrightarrow{LS} = \sum_1^N \vec{x_i}$ is the linear sum of the $N$ data points;
- $\overrightarrow{SS} = \sum_1^N (\vec{x_i})^2$ is the square sum of data points.

Thus, given a MC that summarizes the information about a set of $N$ data points it is possible to calculate measures such as Centroid and Radius of a MC without storing all the examples using the following equations:

$$Centroid = \frac{\overrightarrow{LS}}{N}$$

$$Radius = \sqrt{\frac{\overrightarrow{SS}}{N} - \left(\frac{\overrightarrow{LS}}{N}\right)^2}$$

MCs have interesting properties that make them a natural choice for the stream problems. The most important properties are *incrementality* and *additivity*. For example, given a set of data points where their statistics are stored in a $MC_A = (N_A, \overrightarrow{LS}_A, \overrightarrow{SS}_A)$, we can incrementally add a new example $\vec{x}$ in $MC_A$ updating their statistics in the following way:

$$\overrightarrow{LS}_A \leftarrow \overrightarrow{LS}_A + \vec{x}$$
$$\overrightarrow{SS}_A \leftarrow \overrightarrow{SS}_A + (\vec{x})^2$$
$$N_A \leftarrow N_A + 1$$

The additivity considers that if we have two disjoint Micro-Clusters $MC_A$ and $MC_B$, the union of these two groups is equal to the sum of its parts. Thus, the sufficient statistics of a new Micro-Cluster $MC_C = (N_C, \overrightarrow{LS}_C, \overrightarrow{SS}_C)$ that stores the information of $MC_A \cup MC_B$ are computed:

$$\overrightarrow{LS}_C \leftarrow \overrightarrow{LS}_A + \overrightarrow{LS}_B$$
$$\overrightarrow{SS}_C \leftarrow \overrightarrow{SS}_A + \overrightarrow{SS}_B$$
$$N_C \leftarrow N_A + N_B$$

Although it is efficient and appropriate to data stream problems, we noted that the MC representation has been essentially used in clustering problems. In this paper, we propose their use for classification of evolving data streams. Therefore, we modified the representation to store the information about the class of a set of data points. Thus, our representation is a 4-tuple $(N, \overrightarrow{LS}, \overrightarrow{SS}, y)$, where $y$ is a label for a set of data points. The general idea of the algorithm is presented next.

The algorithm begins receiving as input a reduced initial set of labeled data $\mathcal{T}$. Such data are necessary to define the problem as classification, including the number of classes and their initial disposition in feature space before the occurrence of changes. The only parameter of our algorithm is the maximum radius $r$ of the MCs. This parameter is better explained next.

From the initial labeled data $\mathcal{T}$, the algorithm builds a set of labeled MCs where each MC has information about only one example. In the classification phase, the predicted label $\hat{y}_t$ for each example $\overrightarrow{x_t}$ from the stream is given by the nearest MC according to Euclidean distance. At this moment, it is verified if the addition of $\overrightarrow{x_t}$ in their correspondent nearest MC using the *incrementality* property it will exceed the maximum MC radius $r$ defined by the user. If the radius does not exceed the threshold $r$, the example $\overrightarrow{x_t}$ is added to the nearest MC and its sufficient statistics $(N, \overrightarrow{LS}, \overrightarrow{SS})$ are updated. Thus, for each new example added, the centroid position of the updated MC is slightly dislocated in direction to the newly emerging concept of the class. On the other hand, if the radius exceeds the threshold, a new MC carrying the predicted label $\hat{y}_t$ is created to allocate the new example $\overrightarrow{x_t}$. Besides, the algorithm searches by the two farthest MCs from the predicted class to merge them using the *additivity* property. Therefore, the two farthest MCs

from $\overrightarrow{x_t}$ are merged into one MC that will be placed closest to the emerging new concept. This strategy of Micro-Clusters online maintenance is used over time to constantly adapt the classification model for incremental changes in data streams.

## V. Experimental Evaluation

In this section, we present the setup used in our experimental evaluation and the analysis of the results.

### A. Setup

We evaluate *MClassification* on 16 synthetic benchmark datasets and 2 real-world problems, as previously proposed in [12] to evaluate incremental evolving data. More details about this benchmark can be found in the original paper [12] and a visual description in video can be seen in the website *Nonstationary Environments – Archive*[1].

We compare our method against two bounds that simulate a static supervised learning classifier and a classifier that is constantly updated with zero delay of actual labels. We also compare our method against the most recent proposed algorithm for this scenario, SCARGC. A more detailed description about rival methods is presented below.

- **Static.** The classifier is trained with a batch of initial examples from the stream and it is not updated over time. Thus, it is possible to observe whether the data at the beginning of stream are sufficiently representative so that the classifier does not need to adapt over time in a static scenario;
- **Sliding.** Similarly to the previous setting, the classifier is initially trained with the first examples from the stream. In the test phase, the classifier is constantly updated whenever a new example is processed using a sliding window. The oldest example is dropped off from the window and the most recent processed example is added with its *actual* label. This setting represents a very optimistic scenario where the labels of processed examples do not have any delay in their availability;
- **SCARGC.** Our main rival is the SCARGC algorithm previously presented in Section III. We compared the accuracy results of SCARGC achieved with the parameters suggested in [12].

### B. Analysis of Results

Table I presents the average accuracy achieved by the methods over the entire stream. In this table, we highlight the best result in a comparison between the proposed method *MClassification* and SCARGC. For all synthetic datasets, we use the following parameters for *MClassification*: $|\mathcal{T}| = 150$ and $r = 0.1$, where $|\mathcal{T}|$ represents the size of initial labeled set and $r$ is the maximum MC radius.

We carried out the Friedman test with the Nemenyi posthoc with 95% as the confidence level to statistically compare the average accuracies achieved over time by the four algorithms evaluated. In Fig. 3 we show the critical difference diagram for ranked accuracies that illustrate the test. In this diagram, we connect in solid bars the groups of algorithms that are not significantly different from each other [15].

[1]https://sites.google.com/site/nonstationaryarchive/

TABLE I. Average accuracies over time on benchmark data

| Dataset | Static | Sliding | SCARGC | MClassification |
|---|---|---|---|---|
| *1CDT* | 97.01 | 99.88 | 99.75 | **99.89** |
| *1CHT* | 91.96 | 99.24 | 99.25 | **99.38** |
| *1CSurr* | 65.75 | 98.52 | **94.35** | 85.15 |
| *2CDT* | 54.38 | 93.47 | 90.92 | **95.23** |
| *2CHT* | 54.03 | 85.44 | 86.02 | **87.93** |
| *4CE1CF* | 95.81 | 97.15 | 94.08 | **94.38** |
| *4CR* | 25.06 | 99.98 | 99.95 | **99.98** |
| *4CRE-V1* | 26.17 | 97.64 | **97.39** | 90.63 |
| *4CRE-V2* | 27.11 | 89.37 | **91.90** | 91.59 |
| *5CVT* | 40.72 | 86.86 | **90.15** | 88.40 |
| *FG 2C 2D* | 81.30 | 93.84 | **95.16** | 62.48 |
| *GEARS 2C 2D* | 93.62 | 99.86 | **95.89** | 94.73 |
| *MG 2C 2D* | 49.00 | 90.40 | **92.71** | 80.58 |
| *UG 2C 2D* | 47.28 | 94.27 | **95.56** | 95.28 |
| *UG 2C 3D* | 60.64 | 92.86 | **94.77** | 94.72 |
| *UG 2C 5D* | 68.81 | 89.91 | 90.98 | **91.25** |
| *NOAA* | 66.19 | 72.01 | **68.61** | 67.54 |
| *KEYSTROKE* | 68.69 | 90.14 | 87.72 | **90.62** |

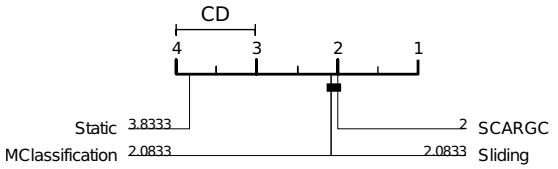


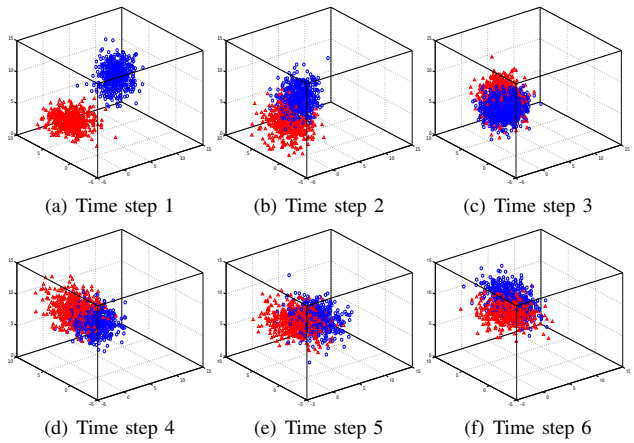

Fig. 3. Critical difference diagram considering the benchmark data

We can note in the diagram from Fig. 3 that the method with best mean rank is SCARGC. In the second place, there is a tie between the upper-bound *Sliding* and *MClassification*. However, it is important to note that there is no statistically significant difference among the results achieved by these three methods. The tie with *Sliding* is very positive given that this setting is very optimistic and does not consider any delay in the availability of actual labels. Although SCARGC slightly outperforms our method, there are no statistical difference between them and our method have the advantage of having a simpler parameter. Moreover, our method allows that the number of groups to change over time.

To better understand the evolving behavior of the datasets and detail the results achieved by the algorithms over time, we present some examples in more details. For instance, Fig. 4 shows 6 snapshots that illustrate the behavior of UG-2C-5D dataset. Although this dataset has 5 features, we display only 3 of them in the illustrations.

The results achieved over time for the UG-2C-5D dataset, considering 100 steps, are shown in Fig. 5. We can note that the *MClassification* slightly outperforms the SCARGC and *Sliding* and outperforms with a large margin the *Static* setting that does not consider change in data distribution.

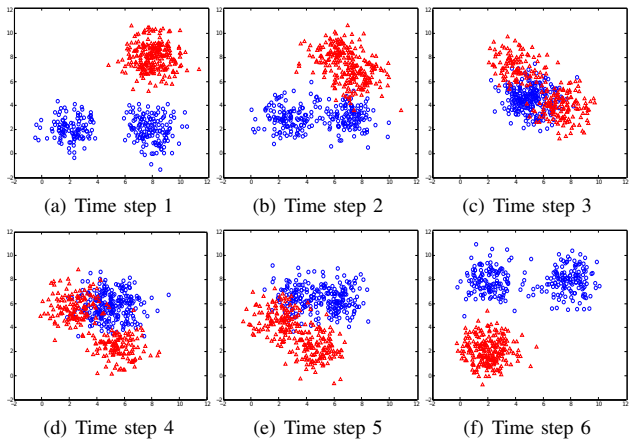
An interesting dataset is MG-2C-2D. This dataset has 2 classes, where the data of one of them begins with two Gaussian distributions and the data of the other class begins distributed in only one. Over time, the two classes change their positions and the parameters of the distributions. Fig. 6 illustrates the behavior of MG-2C-2D and Fig. 7 shows the respective results.



(a) Time step 1 (b) Time step 2 (c) Time step 3

(d) Time step 4 (e) Time step 5 (f) Time step 6

Fig. 4. Snapshots that illustrate the behavior of the two classes from UG-2C-5D dataset over time considering 3 of 5 features of the data



(a) Time step 1 (b) Time step 2 (c) Time step 3

(d) Time step 4 (e) Time step 5 (f) Time step 6

Fig. 6. Snapshots that illustrate the behavior of the two classes from MG-2C-2D dataset over time

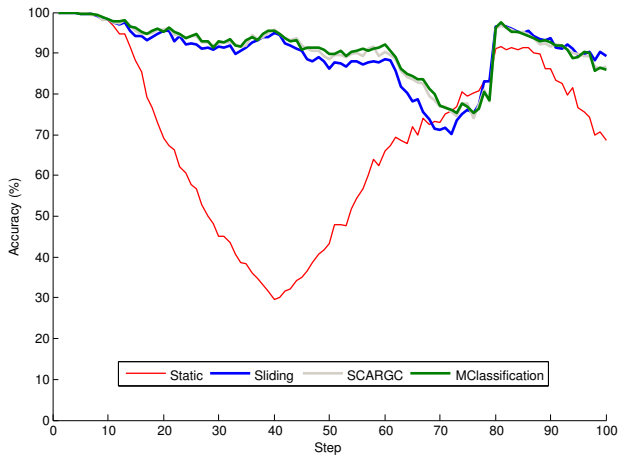



Fig. 5. Results achieved over time by the methods in UG-2C-5D dataset

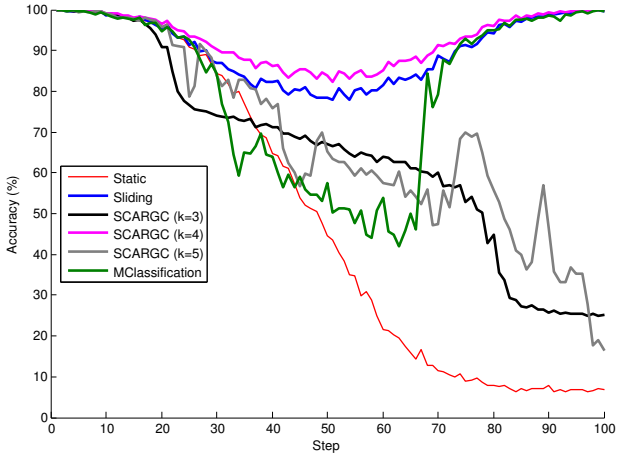



Fig. 7. Results achieved over time by the methods in MG 2C 2D dataset

Between the steps 30–70, we noted in Fig. 7 that *MClassification* shows a mean accuracy around 65%, while SCARGC with their optimal parameter ($k = 4$) and *Sliding* have around 85% of accuracy in the same period. Although this difference may seem high, *MClassification* shows a competitive result after this period. It is also important to note that SCARGC shows this results using 4 groups in their clustering phase ($k$ parameter). However, from the initial labeled set it is very hard to know a priori the evolution of the parameter of the distributions, being a more natural choice the use of 3 groups (see Fig. 6-a). Thus, the *MClassification* have the advantage of not requiring this crucial parameter. To illustrate the difficulty of SCARGC with non-optimal value for $k$ parameter, we also show the results with $k = 3$ and $k = 5$ in Fig. 7. As we can see, with $k = 3$ SCARGC achieves an average accuracy of $64.86\%$ over the entire stream and $68.18\%$ for $k = 5$, while *MClassification* achieves $80.58\%$ without a parameter about the number of groups.

For real world problems, we evaluated our algorithm in NOAA and Keystroke datasets. NOAA is a dataset of weather measurements collected over 50 years which contains eight features: temperature, dew point, sea-level pressure, visibility, average wind speed, max sustained wind speed, and minimum and maximum temperature. The classification task is to determine if it will rain or not in 18,159 daily readings. The results for this problem are presented in Fig. 8, where we note a very similar result achieved by SCARGC and *MClassification*. *Static* and *Sliding* were initially trained with the first 30 examples, SCARGC and *MClassification* use only the first 10 examples.

The second real data consists of the use of keystroke dynamics to recognize users by their typing rhythm that evolves over time. In this dataset, 4 users typed the password ".tie5Roanl" plus the *Enter* key 400 times captured in 8 sessions performed in different days. To perform the user classification task, we used 10 features extracted from the *flight time* for each pressed key. The flight time is the time difference between the instants when a key is released and the next key is pressed. The evaluation assumes that the algorithms were trained with labeled data from the first session and tested in the remaining 7 sessions. The results are presented in Fig. 9 and we can note that *MClassification* shows the best result.

To evaluate the time spent for classification we choose three datasets from the benchmark with 200,000 examples: UG-

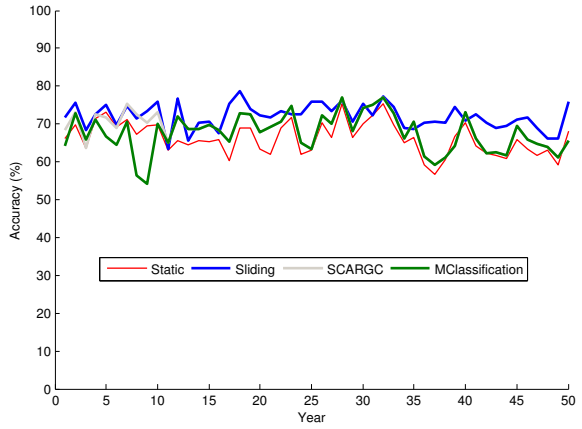



Fig. 8. Results achieved over time by the methods in the real data NOAA

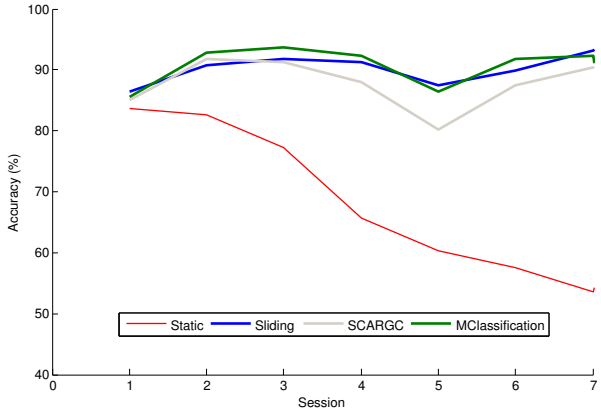



Fig. 9. Results achieved over time by the methods in the real data Keystroke

2C-2D, UG-2D-3D, and MG-2C-2D. We select these datasets because we have the time costs for all rival methods from the literature. The results are presented in Table II.

TABLE II. TIME COSTS (IN MINUTES) SPENT BY THE ALGORITHMS

| Dataset | APT | COMPOSE | SCARGC | MClassification |
|---|---|---|---|---|
| UG-2C-2D | 3,600 | 4,160 | 1.00 | 5.14 |
| UG-2C-3D | 22,776 | 26.660 | 1.97 | 12.26 |
| MG-2C-2D | 20,303 | 8.330 | 1.96 | 10.66 |

Due to its simplicity, SCARGC is very efficient and hard to beat in terms of the time cost. However, *MClassification* is faster than APT algorithm and has a competitive time cost compared to COMPOSE. We can note in Table II that in the 3-dimensional dataset UG-2C-3D, COMPOSE spends more than three times compared to 2-dimensional datasets (UG-2C-2D and MG-2C-2D). In fact, the increase of dimensionality is a problem for COMPOSE. For *MClassification*, the increase of dimensionality does not affect drastically its performance. Although our method does not show the best time results, it is important to note that about 12.255 minutes (in the worst case, UG-2C-3D dataset) to classify 200,000 examples represents an adequate and sufficient time rate of 0.0037 second to classify each example from the stream.

## VI. CONCLUSIONS

The problem of classification of evolving data with infinitely delayed labels is non-trivial and an open challenge in data stream mining research. The main contribution of this paper is to introduce the algorithm *MClassification* to deal with this problem that affects many real world problems. There are only few research papers in the literature that consider this scenario and some of the existing solutions have an impracticable computational cost or crucial parameters, which optimal values are difficult to find. The proposed algorithm in this paper shows competitive accuracy results to state-of-the-art methods, practical time costs and a single parameter. The main advantage of our method is the absence of critical parameters that require prior knowledge of the user to tuning them as occurs with rival methods. In future work, we intend to explore new strategies for the maintenance of MCs to deal better with outliers. An initial idea is to weight the MCs according to their age or utility so that more importance is given to the most recent MCs or those used for classification.

## ACKNOWLEDGMENT

This work was funded by FAPESP awards #2011/17698-5, #2013/26151-5, and 2015/07628-0.

## REFERENCES

[1] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, p. 44, 2014.

[2] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, and J. Stefanowsky, "Open challenges for data stream mining research," *ACM SIGKDD Explorations Newsletter*, vol. 16, no. 1, pp. 1–10, 2014.

[3] D. F. Silva, V. M. A. Souza, D. P. W. Ellis, E. Keogh, and G. E. A. P. A. Batista, "Exploring low cost laser sensors to identify flying insect species," *Journal of Intelligent & Robotic Systems*, pp. 1–18, 2014.

[4] V. M. A. Souza, D. F. Silva, and G. E. A. P. A. Batista, "Classification of data streams applied to insect recognition: Initial results," in *BRACIS*, 2013, pp. 76–81.

[5] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *SBIA*, 2004, pp. 286–295.

[6] J. Gao, W. Fan, J. Han, and S. Y. Philip, "A general framework for mining concept-drifting data streams with skewed distributions," in *SDM*, 2007.

[7] A. Tsymbal, "The problem of concept drift: definitions and related work," Tech. Rep., 2004.

[8] M. B. Harries, C. Sammut, and K. Horn, "Extracting hidden context," *Machine learning*, vol. 32, no. 2, pp. 101–126, 1998.

[9] I. Zliobaite, "Change with delayed labeling: when is it detectable?" in *ICDM Workshops*, 2010, pp. 843–850.

[10] K. B. Dyer, R. Capo, and R. Polikar, "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data," *IEEE TNNLS*, vol. 25, no. 1, pp. 12–26, 2014.

[11] G. Krempl, "The algorithm APT to classify in concurrence of latency and drift," in *IDA*, 2011, pp. 222–233.

[12] V. M. A. Souza, D. F. Silva, J. Gama, and G. E. A. P. A. Batista, "Data stream classification guided by clustering on nonstationary environments and extreme verification latency," in *SDM*, 2015, pp. 873–881.

[13] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *ACM SIGMOD Record*, vol. 25, no. 2, 1996, pp. 103–114.

[14] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *VLDB*, vol. 29, 2003, pp. 81–92.

[15] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *JMLR*, vol. 7, pp. 1–30, 2006.