Classifying Hierarchical Data Streams using Global Classifiers and Summarization Techniques

Eduardo Tieppo, Jean Paul Barddal, Júlio Cesar Nievola
Programa de Pós-Graduação em Informática (PPGIa)
Pontifícia Universidade Católica do Paraná (PUCPR)
Curitiba, Brazil
{eduardo.tieppo, jean.barddal, nievola}@ppgia.pucpr.br

***Abstract*—The hierarchical classification of data streams requires models capable of handling a class hierarchy and updating themselves whenever a new example arrives, within restrained processing time and memory consumption. Current state-of-the-art models store raw instances and handle the hierarchy locally, performing a high number of computations at every hierarchy level and with all, eventually redundant, data. This paper introduces Global k-Nearest Centroids (kNC) and Global Dribble, two novel methods for the hierarchical classification of data streams. Both methods use summarization techniques to represent data with constant computational resources usage and a global classification approach to process instances in less time when compared to local strategies. We compare both methods with a state-of-the-art local classifier, and the proposed methods achieved a higher number of correct predictions and process instances nearly twice as fast.**

***Index Terms*—Hierarchical Classification, Data Stream Classification, Data Summarization**

I. Introduction

Hierarchical classification of data streams is applied to problems in which the classes associated with instances are hierarchically structured, and these instances are provided to the model one by one from the data stream over time. This kind of problem can be observed, for example, in the real-time identification of insect vectors of diseases hierarchically organized by their biological taxonomy, having an important impact on public health policies [1].

Hierarchical data stream classifiers inherit constraints and challenges from both traditional hierarchical classification and data stream classification. First, regarding hierarchical classification, classifiers must deal with the existing hierarchy in data classes, thus predicting a label path representing hierarchical relationships between labels (or classes) for a given instance instead of an independent label for a given instance [2]–[4]. Focusing on data stream classification, classifiers must handle potentially infinite data, and thus, must process data sequentially and discard them right after, while also assuming that their underlying distribution may change over time, a phenomenon called concept drift [5]–[8].

Despite the research conducted on both data stream and hierarchical classification areas separately, they do not comprehend each other [2], [9]. Besides, current state-of-the-art methods introduced for the hierarchical classification of data streams present limitations when applied to real-world problems, as they use complete representations of data and eventually perform redundant steps in their learning models [10]. In other words, methods that are computationally heavy-weighted in terms of processing time and memory consumption may represent infeasible strategies in handling potentially unbounded hierarchical data streams [6]–[8].

Supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

In this study, we propose two methods for the hierarchical classification of data streams: Global k-Nearest Centroids and Global Dribble. Both methods use summarization techniques, i.e., incremental centroids [11], and cluster feature vectors [12] to work with potentially unbounded hierarchical data streams with constant memory. Both of the proposed methods follow a global strategy to handle the hierarchy in their learning process, and thus, process new instances with fewer comparisons when compared to local state-of-the-art k-Nearest Neighbors methods [10].

In summary, our contributions are as follows:

- We propose Global k-Nearest Centroids and Global Dribble, two methods for the hierarchical classification of data streams using summarization techniques. We adapt and apply summarization techniques, i.e., incremental centroids [11], and cluster feature vectors [12], as part of a hierarchical data stream classification process, being able to work with constant memory and time.
- We propose a global classification approach to handle the class hierarchy based on the label paths of the nearest neighbors, thus reducing the dependence on distance computations and processing new instances performing less distance computations.
- As a byproduct of this research, the source code of the proposed methods and the datasets used in the experiments are made available for reproducibility.

The remainder of this paper is organized as follows. Section II describes the problem of hierarchical classification of data streams and Section III brings forward related works. Section IV describes the proposed methods for the hierarchical classification of data streams using summarization techniques. Section V comprises the experimental protocol and the discussion of the results obtained. Finally, Section VI concludes this paper and states envisioned future works.

II. Problem Statement

In this section, we introduce the basics of hierarchical classification of data streams. More specifically, we describe different approaches that can be used by hierarchical data stream classifiers to deal with a class hierarchy, including the local approach used by state-of-the-art related works and the global approach applied in our proposal, which allow a more efficient computational resource usage without noticeable losses in the prediction quality rates.

Formally, we define $hDS = [(\vec{x}^t, \vec{y}^t)]_{t=0}^{\infty}$ to be a hierarchical data stream providing examples in the $(\vec{x}^t, \vec{y}^t)$ format. Each example arrives at a unique exclusive timestamp t, where $\vec{x}^t$ is a set of d-dimensional features and their respective values. Additionally, $\vec{y}^t$ represents the associated ground-truth label path (hierarchically structured classes) for the given instance $\vec{x}^t$ [13].

Given a data stream hDS, a hierarchical data stream classifier learns a function $f^t : \vec{x}^t \mapsto \vec{y}^t$ that maps features and their values $\vec{x}^t \in \mathbb{R}^d$ to label paths $y^t \in Y$ [6].

In streaming scenarios, classifiers need to adopt strategies to support potentially unbounded hierarchical data streams, which can consequently change their underlying distribution over time. In other words, they need to work with constrained computational resources to be able to process incoming instances without discarding them and respond appropriately to possible concept drifts [5], [9], [14].

To this end, hierarchical data stream classifiers can be incremental or adaptive by updating or retraining their models using part of or all data as new data becomes available. Incremental methods do not have strategies to forget the information previously learned and may become unstable due to the size of the model and are not well suited to adapt to concept drifts. On the other hand, adaptive methods forget the information previously learned either explicitly or implicitly.

Explicit drift detection regards the application of drift detectors that mostly use statistical techniques with time window strategies to monitor data behavior and trigger if some change in data distribution occurs, making the model more suitable to adapt to changes in data over time [9], [15].

Hierarchical data stream classifiers can use different approaches to handle the class hierarchy. Figure 1 illustrates these approaches. Approaches that fall in the Local classifiers per node (LCN) category use one binary classifier per class in the hierarchy. Local classifiers per parent node (LCPN) apply one multi-class classifier per class to predict between its child nodes. Local classifiers per level (LCL) use one multi-class classifier at the same level in the hierarchy. Finally, a global classifier (GC) is a multi-class classifier able to handle all classes simultaneously while also considering class hierarchy [2].

As stated at the beginning of this section, we highlight that a global approach requires less computational resources than a local approach since it uses a single classifier in contrast to multiple ones used in any local approach [2], and thus, we further investigate whether it is more fitted to the constraints imposed by the hierarchical data streams setting.

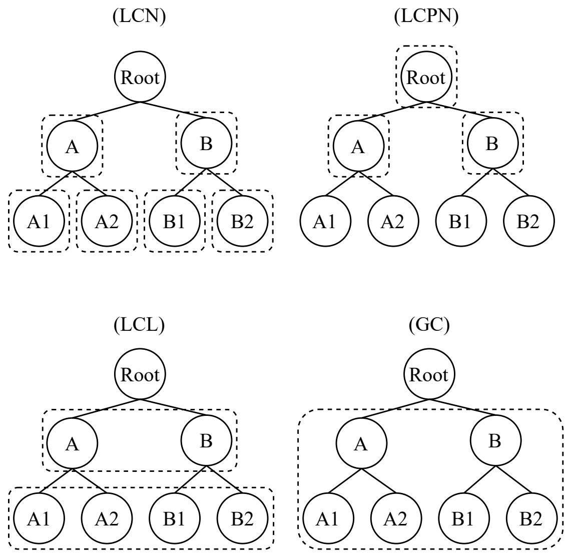

Fig. 1. Hierarchical classification approaches (circles represent classes and dashed squares enclose classes to be predicted by a classifier).

III. Related Work

The hierarchical classification of data streams task was first addressed in [10], where the authors proposed a k-Nearest Neighbors (kNN) classifier to classify hierarchically structured data streams. The method stores n instances on buffers assigned to each class node in the hierarchy. When the model receives a new instance, it applies a sliding window strategy [14] to forget older data as a first-in/first-out strategy is followed. Additionally, the method uses local classifiers per parent node, building sub-datasets of instances at each step to perform the distance computations required by the kNN method.

Despite being an effective hierarchical data stream classifier, the method proposed in [10] has two related main drawbacks regarding hierarchical classification of data streams: (i) it stores raw instances on node buffers, and (ii) it uses local classifiers per parent node.

Let us elaborate on that. Let n be the number of d-dimensional instances stored on each node buffer, c and g the number of children and descendants of a given parent node p, and h the height of the tree representing the class hierarchy. By using local classifiers per parent node, for each incoming instance i the method needs to perform $((n \times c) + (n \times g)) \times c$ distance computations per level on h. In other words, depending on the h depth, computations tend to the quadratic form $c^2 \times (c \times n + g \times n)^2$ of n. On the other side, a global classifier maintains $((n \times c) + (n \times g)) \times c$ with a linear n regardless of h.

Furthermore, by using raw instances on the node buffers, the distance computations are performed n times on the standard d^2 basis. Alternatively, data summarization techniques can be used to reduce this dependence and perform only d^2 assuming, for example, in the best scenario, $n = 1$ to represent the original instances in a single statistical descriptor [11], [14].

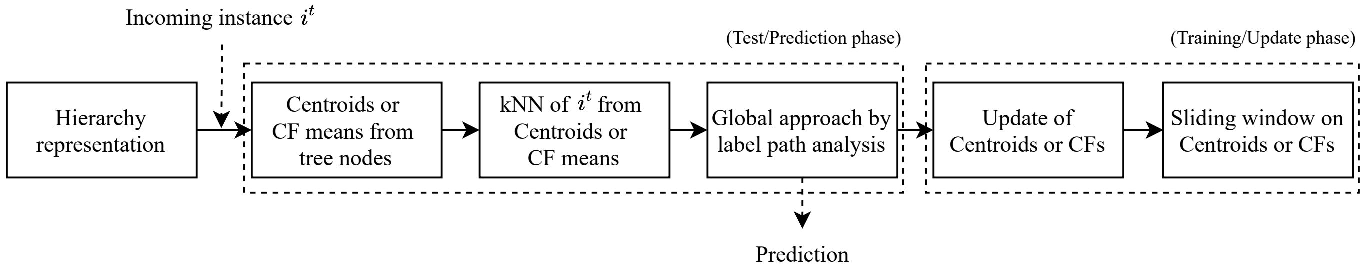
Incoming instance i^t
(Test/Prediction phase)
(Training/Update phase)
Hierarchy representation
Centroids or CF means from tree nodes
kNN of i^t from Centroids or CF means
Global approach by label path analysis
Update of Centroids or CFs
Sliding window on Centroids or CFs
Prediction

Fig. 2. General view of Global kNC and Global Dribble learning process.

In this study, we use incremental centroids [11] and cluster feature vectors [12] to summarize instances and reduce the dependence on distance computations. Also, we perform a global approach to handle the class hierarchy based on distance computations and label path analysis, thus tackling both drawbacks described above in existing local approaches.

IV. Proposed Methods

In this section, we propose Global k-Nearest Centroids (Global kNC) and Global Dribble, two global methods for the hierarchical classification of data streams that rely on summarization techniques to represent data.

Figure 2 illustrates the steps of the learning process shared by both methods. After building the hierarchy representation, each method receives instances $i^t = (\vec{x}^t, \vec{y}^t)$. A test/prediction phase starts by obtaining all data from nodes, comparing them to find nearest neighbors, and applying a global approach via label path analysis. At this point, the model can be requested to predict a class (label path) for $\vec{x}^t$. Next, the instance $\vec{x}^t$ and the corresponding label path $\vec{y}^t$ are used to update the model, and older data is discarded through a sliding window. These steps are described in detail below.

In the hierarchy representation step, both methods build a tree of nodes. Each node represents a class in the class taxonomy and contains references to its parent nodes, an instance counter, and a data structure that can be an incremental centroid in Global kNC or a Cluster Feature Vector (CF) in Global Dribble.

The incremental centroid represents the incremental mean of the instances associated with a given class node [11]. The incremental mean μ_t is based on incoming instances i^t and is computed as described in Equation 1, where $\bar{x}_{t-1}$ represents the current average, t the number of instances observed thus far, and $\vec{x}^t$ composes the the arriving instance i^t to be incorporated.

$$\mu_t = \frac{\bar{x}_{t-1}(t-1) + \vec{x}^t}{t} \tag{1}$$

It is essential to highlight that Global kNC summarizes data using centroids and discards the instances afterward, resulting in concise information storage with smaller memory consumption and fewer distance computations.

On Global Dribble, data are stored using Cluster Feature Vectors (CFs) [12], representing instances associated with a given class node as hyperspherical regions with a mean and a radius. A CF is a triplet in the format $CF = (N, LS, SS)$, where N is the number of instances of the cluster summary, and LS and SS are N-dimensional vectors representing the linear and square sum of the instances, respectively. The mean (μ) and the radius (r) of a CF are computed as described in Equations 2 and 3, where d is the number of features available.

$$\mu(CF_i) = \frac{LS_i}{N_i} \tag{2}$$

$$r(CF_i) = \frac{1}{d}\sum_{j=1}^{d}\sqrt{\frac{N_i(SS_i) - 2(LS_i^2) + N_i(LS_i)}{N_i^2}} \tag{3}$$

As CFs work with additive components, two cluster feature vectors CF_i and CF_j can be merged by summing their components. Equation 4 describes the additive property of CFs [12], [16]:

$$CF_k = CF_i + CF_j = (N_i + N_j, LS_i + LS_j, SS_i + SS_j) \tag{4}$$

As in the Global kNC method, it is relevant to highlight that Global Dribble summarizes data using CFs and also discards instances right after, thus allowing smaller memory consumption as a consequence of an effective data representation.

After that, when receiving a new i^t instance, both methods apply the prequential approach in their processes [7], [9]. In other words, the new instance is tested and, only after that, is used to update the model (training).

For testing, the methods obtain all centroids (Global kNC) or CFs (Dribble) stored at the tree nodes. On Global kNC, obtaining the centroids is straightforward since the centroids themselves (mean instances) are stored incrementally. Meanwhile, Global Dribble computes the means of the CFs representing the central point of the hyperspheres. Besides, Global Dribble applies an outlier control, retrieving only CFs with a minimal number of instances represented.

Next, the algorithm calculates the Euclidean distance between the instance i^t and the CFs' centers to find its k-nearest neighbors.

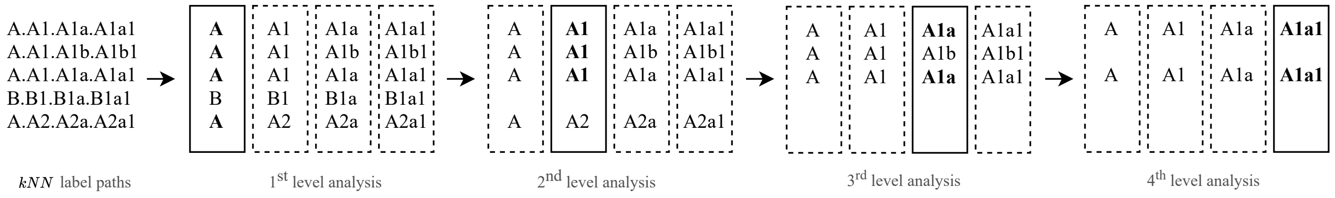

Fig. 3. Global approach through label path analysis using label frequency per hierarchy level.

Note that the building of datasets for distance computations considers the entire hierarchy of the tree. Therefore, the resulting kNN can represent nodes at any hierarchy level. In contrast, in local approaches, the building of datasets is restricted to portions of the hierarchy (by node, by parent node, or by level) and needs to be performed several times until reaching the deeper levels of the hierarchy.

In the global approach step, both methods analyze label paths retrieved from the kNN step, not comparing instances but the frequency of component labels of each level of the label paths.

Figure 3 illustrates this process with illustrative label paths and using $k = 5$ on the kNN. The most frequent label in each level is chosen, and the other label paths are discarded in the analysis of the next level. Ties are decided using the lowest distance between neighbors. The process is repeated until it reaches a leaf node.

After performing the test/prediction step, the instance i^t is used as a training instance being incorporated into the model using its ground-truth label. In Global kNC, the centroids are updated using Equation 1. In Global Dribble, i^t is first represented as a CF and then added to an existent CF using Equation 4.

Note that despite being similar in the test/prediction phase, both methods use quite different data storage and summarization strategies in the training/update phase. While Global kNC summarizes instances in the order of arrival and obtains sliding mean centroids over the data, Global Dribble checks whether the instance i^t is encompassed by any of the CFs stored in the respective node representing the class of i^t. If the instance is encompassed by one of the CFs (i.e., if the instance is between the mean and the radius of the hypersphere), it is added to the CF. Otherwise, it starts a new CF.

Finally, the methods apply a sliding window to adequately respond to possible concept drifts, as well as to maintain constant computational resource usage during the data stream processing.

The Global kNC method works with two upper boundaries n and m in its centroids representing, respectively, the number of centroids to be stored in each node of the hierarchy and the number of instances summarized in each centroid. Upon reaching the upper bound m, Global kNC creates a new centroid and associates it with a given node. If the number of centroids exceeds the upper bound n, the oldest centroid is discarded. Similarly, Global Dribble also works with two upper boundaries n and m on its CFs representing the number of CFs to be stored in each node of the hierarchy and the number of instances summarized in each CF. However, when the upper bound m is surpassed, the oldest instance is subtracted from the CF. In other words, the values representing the current center of the hypersphere described by the CF are removed. Additionally, if the number of CFs exceeds the upper bound n, there are no older CF to be discarded since CFs represent different subconcepts within the same class (unlike the centroids applied in Global kNC). To solve this, the two closest CFs are merged using their additive property (Equation 4).

Algorithms 1 and 2 depict the pseudocodes for the Global kNC and Global Dribble methods that encompass the steps described above.

Both algorithms receive a hierarchical data stream hDS providing instances $(\vec{x}, \vec{y})$ over time, and the above described n, m and k parameters. The hierarchy representation step (line 1) and the test/prediction phase (lines 2-10) are similar

Algorithm 1:
Global k-Nearest Centroids (kNC) method for the Hierarchical Classification of Data Streams

```
input : $hDS$ – a hierarchical data stream providing instances
        $(\vec{x}, \vec{y})$
        $n$ – maximum number of centroids
        $m$ – maximum number of instances to be summarized on a
        centroid
        $k$ – number of nearest centroids
output: $\hat{\vec{y}}_i$ – a predicted label path for the input instance
1  Tree ← classTaxonomy($hDS$);
2  foreach ($\vec{x} \in hDS$) do
3      foreach (childNode ∈ Tree.root.descendants) do
4          targets ← targets ∪ {(childNode.label, childNode.data)};
5      end
6      foreach (target ∈ targets) do
7          target ← target ∪ {euclideanDistance($\vec{x}$,target.data)};
8      end
9      targets = $(\text{targets})_{1..k}$;
10     $\hat{\vec{y}}_i$ ← mostFrequentInLevel(targets);
11     correctNode ← Tree.$\vec{y}_i$;
12     if (correctNode.newestCentroid.count < $m$) then
13         correctNode.newestCentroid ← $\mu$;
14     end
15     else
16         correctNode.data ← correctNode.data ∪ {newCentroid($\vec{x}$)};
17     end
18     if (correctNode.count > $n$) then
19         correctNode.data ←
             correctNode.data \ {correctNode.oldestCentroid};
20     end
21 end
```

Algorithm 2:

Global Dribble method for the Hierarchical Classification of Data Streams

```
input : $hDS$ – a hierarchical data stream providing instances ($\vec{x}, \vec{y}$)
        $n$ – maximum number of $CFs$
        $m$ – maximum number of instances to be summarized on a $CF$
        $k$ – number of nearest $CF$ means
output: $\widehat{y_i}$ – a predicted label path for the input instance
1  Tree ← classTaxonomy($hDS$);
2  foreach ($\vec{x} \in hDS$) do
3      foreach (childNode ∈ Tree.root.descendants) do
4          targets ← targets ∪ {(childNode.label, childNode.data)};
5      end
6      foreach (target ∈ targets) do
7          target ← target ∪ {euclideanDistance($\vec{x}$,target.data)};
8      end
9      targets = (targets)$_{1..k}$;
10     $\widehat{y_i}$ ← mostFrequentInLevel(targets);
11     correctNode ← Tree.$\vec{y_i}$;
12     if (nearestCF.distance <= nearestCF.radius) then
13         correctNode.nearestCF ←
             correctNode.nearestCF + newCF($\vec{x}$);
14         if (correctNode.nearestCF.count > $m$) then
15             correctNode.nearestCF ←
                 correctNode.nearestCF−correctNode.nearestCF.mean;
16         end
17     end
18     else
19         correctNode.data ← correctNode.data ∪ {newCF($\vec{x}$)};
20         if (correctNode.count > $n$) then
21             $CF_1, CF_2$ ← findClosestCFs(correctNode.data);
22             $CF_1 \leftarrow CF_1 + CF_2$;
23         end
24     end
25 end
```

between Global kNC and Global Dribble. However, note that Dribble performs an additional step to compute CFs' means (represented by "childNode.data" on line 4). The global approach by label path analysis is performed by a function that implements the strategy described on Figure 3 (line 10). This function receives a set of label paths, recursively removes infrequent labels at each level of the label hierarchy, and returns the most frequent label path of the initial set.

Next (from line 11 onwards), both methods update themselves and apply the sliding windows differently. The Global kNC method updates the centroids using the incremental mean (cf. Equation 1) (line 13) and checks the upper boundaries m and n. If m is reached, the method creates a new centroid instead of filling the newest one (lines 12-16). If n is surpassed, the method applies the sliding window strategy by discarding the oldest centroid (lines 18-20). On Global Dribble, the method updates the CFs using their additive property or creating a new CF (lines 13 and 19). In line 13, the incoming instance is encompassed by the nearest CF, while in line 19, it is not. The method also checks the upper boundaries m and n. If m is surpassed, the method subtracts a mean representation of the hypersphere from the CF (lines 14-16). If n is surpassed, the two closest CFs are merged to match the number of CFs to n (lines 20-23).

V. Analysis

In this section, we describe the experimental protocol used to evaluate our methods, compare them to the related work described in Section III, and discuss the results regarding prediction and performance rates.

A. Experimental Protocol

To verify the impact of the global approach in the proposed methods on prediction and performance rates, we perform an experimental evaluation of Global kNC and Global Dribble, comparing them against the local kNN proposed in [10].

Table I depicts the datasets used as hierarchical data streams in our experiments and their main characteristics.

We compared all methods with the same settings across all datasets varying k, n, and m parameters. We experimented with $k \in \{1, 3, 5\}$, $n \in \{1, 5, 10, 15, 20\}$ and $m \in \{5, 10, 30\}$. As described in Section IV, n represents the upper bound for the number of centroids in Global kNC, and for the number of CFs in Global Dribble. On Local kNN, n represents the buffer size of raw instances. Similarly, m represents the upper bound for the number of instances summarized in a centroid (Global kNC) or a CF (Global Dribble).

We assessed the classifiers predictive correctness using the well-known hierarchical F-measure (hF) [17] following a prequential evaluation method (interleaved test-then-train) [7], [18]. The hierarchical F-Measure is the harmonic mean of hierarchical precision (hP) and hierarchical recall (hR). It was computed and incrementally averaged for all incoming instances from the data stream. Equations 5, 6 and 7 depict hF metric and its components hP and hR, respectively, where $\widehat{y_i}$ stands for the predicted label path and y_i is the ground-truth label path for the i-th instance.

$$hF = \frac{2 \times hP \times hR}{hP + hR} \tag{5}$$

$$hP = \frac{\sum_i |\widehat{y_i} \bigcap y_i|}{\sum_i |\widehat{y_i}|} \tag{6}$$

TABLE I
Description of hierarchical stream datasets used in the experiments.

Dataset	Instances	Features	Classes	Labels per level
Entomology [10]	21,722	33	14	4,6,9,14
Ichthyology [10]	22,444	15	15	2,6,10,15
Insects-a-b [1]	52,848	33	6	1,1,2,6
Insects-a-i [1]	355,275	33	6	1,1,2,6
Insects-i-a-r-b [1]	79,986	33	6	1,1,2,6
Insects-i-a-r-i [1]	452,044	33	6	1,1,2,6
Insects-i-b [1]	57,018	33	6	1,1,2,6
Insects-i-g-b [1]	24,15	33	6	1,1,2,6
Insects-i-g-i [1]	143,323	33	6	1,1,2,6
Insects-i-i [1]	452,044	33	6	1,1,2,6
Insects-i-r-b [1]	79,986	33	6	1,1,2,6
Insects-i-r-i [1]	452,044	33	6	1,1,2,6
Insects-o-o-c [1]	905,145	33	24	4,10,14,24
Instruments [10]	9,419	30	31	5,10,31

$$hR = \frac{\sum_i |\widehat{y_i} \bigcap y_i|}{\sum_i |y_i|} \quad (7)$$

Regarding performance assessment, we measured the time performance of all methods by calculating the number of instances that the method can process and classify per second.

The experiments in this paper were performed using Python 3.7. The scripts containing the source code of Global kNC and Global Dribble methods, as well as the datasets, are freely available for download and reproducibility[1].

Finally, the results obtained by all methods were compared using a significance test of multiple comparisons. More specifically, we used the Friedman test [19] to make multiple comparisons in non-parametric data assuming a null hypothesis that there is no significant difference between the results of all methods in terms of predictive and performance rates. In case of the null hypothesis being rejected, we applied the Nemenyi *post-hoc* test [20] to identify significant differences between two specific classifiers. All significance tests considered a 95% confidence level according to the protocol provided in [21].

B. Results and Discussion

In this section, we detail and discuss the results obtained during experimentation. The analysis was performed concerning predictive correctness and computational performance comparison between all methods. Overall, both global methods outperform the local method in both predictive correctness and computational resources usage. These claims are detailed and validated below.

1) Predictive correctness: Table II depicts the hierarchical F-measure (hF) obtained by the methods tested (highest values per dataset are highlighted in bold). These results represent the best hF rates obtained by methods considering the averaged best-performing parameter configuration across all datasets. In addition, we appended to Table II the average ranking for all methods per dataset.

First, we highlight that the Global kNC and Global Dribble methods obtained the best results in 13 of the 14 datasets, with average hF rates of 75.63% and 74.38%, respectively, against values close to 72% of the Local kNN method. Consequently, both methods obtained the best average rankings of 1.36 for Global kNC and 1.86 for Global Dribble.

In addition, we emphasize that Local kNN and Global kNC methods obtained their best results with more comprehensive data representations ($n = 20$), while Global Dribble had the best performance with small representation ($n = 5$). The Dribble method usually obtains better prediction rates with small numbers of CFs (n) due to the possible noise incorporation since larger numbers of CFs allow the storage of a few instances that are not representative for the model, in contrast to the main CFs with most data.

[1]http://www.ppgia.pucpr.br/~jean.barddal/datasets/globalkNCDribble.rar

TABLE II
hF (%) OBTAINED BY METHODS CONSIDERING THE AVERAGED BEST-PERFORMING PARAMETER SETTING ACROSS ALL DATASETS.

	Local kNN	**Global kNC**	**Global Dribble**
Datasets	$n = 20$, $k = 1$	$n = 20$, $m = 10$, $k = 3$	$n = 5$, $m = 30$, $k = 1$
Entomology	51.51	**57.41**	53.71
Ichthyology	40.55	**41.72**	37.11
Insects-a-b	80.95	**84.37**	83.33
Insects-a-i	79.14	**82.60**	82.55
Insects-i-a-r-b	79.49	**84.28**	83.42
Insects-i-a-r-i	78.52	**82.62**	82.11
Insects-i-b	79.78	**84.03**	83.91
Insects-i-g-b	83.29	**87.99**	86.66
Insects-i-g-i	78.94	82.93	**83.11**
Insects-i-i	78.63	82.57	**83.08**
Insects-i-r-b	80.14	**84.50**	83.48
Insects-i-r-i	78.60	82.62	**82.70**
Insects-o-o-c	55.24	**65.56**	59.50
Instruments	**65.42**	55.59	56.68
Avg. hF	72.16	**75.63**	74.38
Avg. Ranking	2.79	**1.36**	1.86

2) Processing rates: In addition to predictive correctness, data stream classifiers must be able to handle data sequentially within limited time frames.

Therefore, Table III depicts the instances per second rates obtained by methods with the same best-performing parameters settings (highest values per dataset are highlighted in bold).

Global Dribble method obtained the best results in all datasets, consequently reaching the best average of instances processed per second (479.60) and the best average ranking (1.00) among all methods. Such a performance by Dribble

TABLE III
INSTANCES PER SECOND RATES OBTAINED BY METHODS WITH AVERAGED BEST-PERFORMING PARAMETERS SETTINGS.

	Local kNN	**Global kNC**	**Global Dribble**
Datasets	$n = 20$, $k = 1$	$n = 20$, $m = 10$, $k = 3$	$n = 5$, $m = 30$, $k = 1$
Entomology	127	200	**382**
Ichthyology	157	291	**380**
Insects-a-b	151	353	**543**
Insects-a-i	153	354	**542**
Insects-i-a-r-b	153	354	**541**
Insects-i-a-r-i	153	351	**542**
Insects-i-b	148	345	**541**
Insects-i-g-b	158	372	**542**
Insects-i-g-i	154	359	**543**
Insects-i-i	152	354	**545**
Insects-i-r-b	153	356	**543**
Insects-i-r-i	153	353	**540**
Insects-o-o-c	75	127	**275**
Instruments	79	164	**256**
Avg. inst/sec	140.43	309.40	**479.60**
Avg. Ranking	3.00	2.00	**1.00**

can be associated with its best-performing results obtained with small values of n since the method may eventually take advantage of the representation of data streams with more stable concepts.

In other words, depending on the data distribution, methods that use data summarization techniques can benefit from summarizing the entire data stream with sufficient representativeness in even just a few centroids or CFs.

It is also important to highlight that, despite the fact of being adaptive, both Global Dribble and Global kNC methods can also work incrementally by eliminating the storage's upper boundaries from the centroids or CFs, and thus, enhance their ability to represent stabler data streams in their summarization strategies.

Additionally, note that the Global Dribble method obtained its best performance with small values in n and therefore performed computations and distance comparisons between a much smaller volume of data when compared to other methods.

Also, we highlight that Global kNC could obtain a better average ranking against Local kNN even using equal size data representations ($n = 20$).

3) Hyper-parameter sensitivity: Moreover, we evaluate the general behavior of the methods over variations of the n parameter. Tables IV and V depict, respectively, the average Hierarchical F-Measure (%) and the average instances per second rates obtained by methods on each variation of n.

Global kNC and Global Dribble showed the best average results in all variations of n, with Global Dribble obtaining the best result with $n = 1$ and Global kNC with $n \in \{5, 10, 15, 20\}$, resulting in the best average ranking (1.40) for the Global kNC method.

Local kNN and Global Dribble methods had the second-best results, with equal average rankings of 2.40. As discussed above, Global Dribble performs better with small values of n, while Local kNN benefits from bigger data storage.

In terms of instances per second rates, Global kNC and Global Dribble stand out from Local kNN with the first and second rankings in all n variations, processing, respectively, around 415 and 381 instances per second. On average, both methods can process at least 70% more instances than the local approach.

TABLE IV
AVERAGE HIERARCHICAL F-MEASURE (%) OBTAINED BY METHODS ON EACH VARIATION OF n.

n	Local kNN	Global kNC	Global Dribble
1	62.82	68.77	**72.52**
5	67.51	**73.52**	69.88
10	69.90	**74.65**	69.74
15	71.17	**75.14**	69.49
20	72.08	**75.44**	69.28
Avg. hF	68.70	**73.50**	70.18
Avg. Ranking	2.40	**1.40**	2.40

TABLE V
AVERAGE INSTANCES PER SECOND RATES OBTAINED BY METHODS ON EACH VARIATION OF n.

n	Local kNN	Global kNC	Global Dribble
1	459.07	**551.76**	527.77
5	316.64	**467.20**	466.17
10	213.57	**398.77**	379.49
15	169.21	**351.00**	297.00
20	140.43	**310.36**	236.85
Avg. inst/sec	259.79	**415.82**	381.46
Avg. Ranking	3.00	**1.00**	2.00

4) Overall results and statistical validation: Finally, to validate the better results obtained by Global kNC and Global Dribble in the experiments, we applied Friedman and Nemenyi statistical significance tests (as described in Section V-A). We used as sample sets the results obtained by all methods with all parameter variations and performed multiple pairwise comparisons considering the hF and the instances per second rates. Table VI summarizes these sample sets by averaging all hF and instance per second rates obtained by methods and shows the overall average ranking of methods.

The Friedman test showed a statistical difference between hF rates and the *post-hoc* Nemenyi test identified the difference between both global methods and the local kNN. Figure 4 shows a critical difference ($CD = 0.40$) chart obtained after Friedman and Nemenyi tests for the hF rates obtained by the methods.

Global kNC and Global Dribble do not differ significantly, with average rankings of 1.37 and 1.73, respectively. In contrast, Local kNN (in third place) obtained an average ranking of 2.90, surpassing more than twice the critical difference from Global Dribble.

Similarly, regarding computational performance, the Friedman test showed a statistical difference between instances per second rates obtained by the methods. Likewise, the *post-hoc* Nemenyi test identified a difference between both global methods and local kNN. Figure 5 shows the critical differences ($CD = 0.40$) chart obtained after Friedman and Nemenyi tests for the instances per second rates.

The difference between Global kNC (avg. ranking $= 1.31$) and Global Dribble (avg. ranking $= 1.69$) against the local method is even more noticeable, with Local kNN in third place (avg. ranking $= 3.00$), surpassing the critical difference from Global Dribble 3.31 times and from Global kNC 4.27 times.

TABLE VI
OVERALL AVERAGE HIERARCHICAL F-MEASURE (%) AND INSTANCES PER SECOND RATES OBTAINED BY METHODS.

	Local kNN	Global kNC	Global Dribble
Avg. hF	69.60	74.42	73.80
Avg. Ranking	2.90	1.37	1.73
Avg. inst/sec	259.79	418.81	395.49
Avg. Ranking	3.00	1.31	1.69

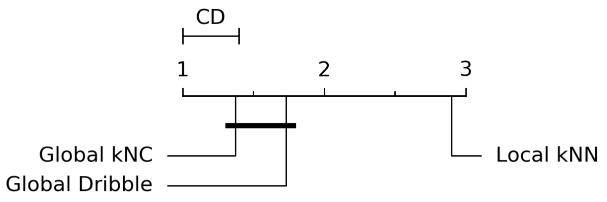

Fig. 4. Critical differences chart for the hierarchical F-measure (hF) rates obtained by methods.

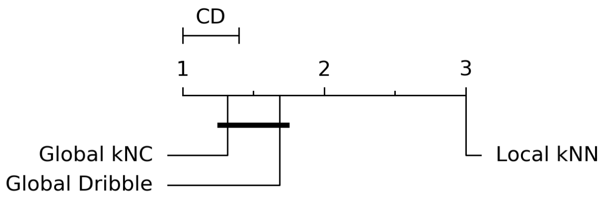

Fig. 5. Critical differences chart for the rates of instances per second rates obtained by methods.

These results prove that Global kNC and Global Dribble methods can use the global approach to obtain a more effective strategy to classify hierarchical data streams, statistically outperforming the Local kNN method in prediction correctness and processing speed.

VI. Conclusion

In this study, we introduced Global kNC and Global Dribble, two global methods for hierarchical data stream classification based on data summarization techniques.

Experimental results showed that the proposed methods obtained significantly better hierarchical goodness-of-fit and processing speed rates when compared to local counterparts.

To facilitate the reproducibility of the proposed methods and experiments, we provide their source code and the datasets used.

As future works, we plan to analyze different distance metrics in the kNN and the development of novel summarization techniques and global classification approaches. Furthermore, we also wish to investigate different stopping criteria during the hierarchy traversing to perform partial predictions with varying confidence rates and to study the impact of adding explicit drift detectors on the classification results.

References

[1] V. M. A. Souza, D. M. Reis, A. G. Maletzke, and G. E. A. P. A. Batista, "Challenges in benchmarking stream learning algorithms with real-world data," *Data Mining and Knowledge Discovery*, pp. 1–54, 2020.

[2] C. N. Silla and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, no. 1-2, pp. 31–72, 2011.

[3] A. Freitas and A. Carvalho, "A tutorial on hierarchical classification with applications in bioinformatics," in *Research and trends in data mining technologies and applications*. IGI Global, 2007, pp. 175–208.

[4] F. Wu, J. Zhang, and V. Honavar, "Learning classifiers using hierarchically structured class taxonomies," in *International Symposium on Abstraction, Reformulation, and Approximation*. Springer, 2005, pp. 313–320.

[5] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[6] J. Gama, *Knowledge discovery from data streams*. Chapman and Hall/CRC, 2010.

[7] A. Bifet and R. Kirkby, "Data stream mining a practical approach," 2009.

[8] B. R. Prasad and S. Agarwal, "Stream data mining: platforms, algorithms, performance evaluators and research trends," *International journal of database theory and application*, vol. 9, no. 9, pp. 201–218, 2016.

[9] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[10] A. R. S. Parmezan, V. M. Souza, and G. E. Batista, "Towards hierarchical classification of data streams," in *Iberoamerican Congress on Pattern Recognition*. Springer, 2018, pp. 314–322.

[11] M. Steinbach, L. Ertöz, and V. Kumar, "The challenges of clustering high dimensional data," in *New directions in statistical physics*. Springer, 2004, pp. 273–309.

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

[13] E. Tieppo, R. R. d. Santos, J. P. Barddal, and J. C. Nievola, "Hierarchical classification of data streams: a systematic literature review," *Artificial Intelligence Review*, pp. 1–40, 2021.

[14] H.-L. Nguyen, Y.-K. Woon, and W.-K. Ng, "A survey on data stream clustering and classification," *Knowledge and information systems*, vol. 45, no. 3, pp. 535–569, 2015.

[15] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 2007, pp. 443–448.

[16] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[17] S. Kiritchenko and F. Famili, "Functional annotation of genes using hierarchical text categorization," *Proceedings of BioLink SIG, ISMB*, 01 2005.

[18] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Machine learning*, vol. 90, no. 3, pp. 317–346, 2013.

[19] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.

[20] P. Nemenyi, "Distribution-free multiple comparisons," in *Biometrics*, vol. 18. International Biometric Society, 1962, p. 263.

[21] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. Jan, pp. 1–30, 2006.