Adaptive Global k-Nearest Neighbors for Hierarchical Classification of Data Streams*

Eduardo Tieppo[1,2], Jean Paul Barddal[1] and Julio Cesar Nievola[1]

***Abstract*— Data stream classification differs from batch learning classification methods as data is made available sequentially and may drift over time. Therefore, data stream classification can be simultaneous to all other kinds of classification problems, and it has been revisiting many aspects related to classification in the last years. So far, hierarchical classification was weakly addressed in streaming scenarios despite being a well-established research topic. To fill in this gap between such areas, in this paper, we propose the adaptive global k-Nearest Neighbors for the hierarchical classification of data streams (Global kNN-hDS). Our proposal classifies hierarchical data streams using a constrained memory buffer and a global classification approach. We compare our method against a state-of-the-art local kNN also tailored for streaming scenarios, and results show that our method obtains competitive prediction rates while being statistically faster.**

I. Introduction

Classification is a recurring task that has been successfully applied to several domains, from biomedicine with the enhancement of Computer-Aided Diagnosis [1], to politics, with the detection of fake news, and even its impact in presidential elections [2]. Because of the wide variety of problems in which classification techniques can be used, these techniques also need to deal with different kinds of data and respond to them accordingly [3].

Classification problems can have two exclusive (non-overlapping) classes (binary classification), several mutually exclusive classes (multi-class classification), many potentially simultaneous classes (multi-labeled classification), and, finally, classes that are layered in a hierarchical structure (a hierarchical classification).

This paper focuses on hierarchical classification, specifically on its intersection with another hot topic in machine learning: data streams. Data stream classification regards a data input characteristic, as instances are made available over time for both testing and training [4]. This classification task works with potentially unbounded data, which arrive continuously, and needs to be processed constrained by limited computational resources. Besides, a data stream model must be adaptive since the data is possibly non-stationary (its probability distribution may change over time) [4].

Despite being an established research topic, the hierarchical classification task was nearly overlooked when considering the data stream scenario.

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

[1]Programa de Pós-Graduação em Informática (PPGIa). Pontifícia Universidade Católica do Paraná. Curitiba, Brazil. {eduardo.tieppo, jean.barddal, nievola}@ppgia.pucpr.br

[2]Instituto Federal do Paraná (IFPR), Pinhais, Brazil. eduardo.tieppo@ifpr.edu.br

Therefore, our goal is to fill this gap by proposing the *Global kNN-hDS*, a global hierarchical data stream classification approach. Our proposal relies on k-Nearest Neighbors [5], which is applied with smaller computational resources usage when compared to the state-of-the-art local approaches.

Considering the previously described context, the main contributions of this study are two-fold:

- We propose the *Global kNN-hDS*, an adaptive global k-Nearest Neighbors method for hierarchical data streams classification. The method is adaptive, works with constrained time and memory, and is able to respond to different contexts in data stream scenarios via windowing strategies.
- We reduce the dependence on distance computations in the k-Nearest Neighbors method by applying a global approach instead of a local one, being able to process new instances performing fewer comparisons to find the nearest neighbors.

The remainder of this paper is organized as follows. Section II describes the background of both hierarchical classification and data stream classification. Section III introduces related works to our proposal, which is brought forward in Section IV. Section V describes the experimental setup and the results obtained. Finally, Section VI concludes this paper and states envisioned future works.

II. Theoretical background

This section provides definitions of both hierarchical and data stream classification tasks to allow a proper understanding of our proposal.

A. Hierarchical Classification

In hierarchical classification, class labels are organized in a hierarchically structured class taxonomy. Problems that fit in these scenarios have data organized with correlated parent and child class labels, such as music genres, news categories, and animal species. Thus, methods assign not one but a set (path) of related labels to an instance. Hierarchical classification problems and algorithms can be categorized accordingly to their data structure, label cardinality, and label depth [6]:

- The class taxonomy may be modeled using a Tree or a Directed Acyclic Graph (DAG) representation, according to how many parent nodes the same node has.
- Instances of a given problem can have only one single path of labels (SPL) associated with them or multiple

paths of labels (MPL). Thereby, methods can also decide between performing a single path prediction (SPP) or a multiple path prediction (MPP) depending on the problem characteristics.
- Methods can perform a mandatory (MLNP) or a non-mandatory (NMLNP) leaf-node prediction, depending on whether the problem supports partial depth labeling (PD) or actual classes of the problem are represented only in the leaf nodes with full depth labeling (FD).

Furthermore, a hierarchical classifier must be able to handle the class hierarchy in its algorithm. Fig. 1 illustrates different kinds of approaches.

In the local classifier per node (LCN) approach, one binary classifier per class handles each class in the hierarchy (except the root node). In the local classifier per parent node (LCPN) approach, one multi-class classifier per class (except on the leaf nodes) predicts between its child nodes. In the local classifier per level (LCL) approach, one multi-class classifier per level predicts between all nodes at the same level. Finally, in the global classifier (GC) approach, one single multi-class classifier is built to handle all classes using the hierarchy information.

Here, it is essential to highlight that a global approach is usually smaller in computational resources than a local approach since it deploys a single model in contrast to multiple models in the local approaches [6]. Fig. 2 illustrates this difference in two kNN methods using a local classifier approach (LCPN, in this case) and a global classifier (GC) approach.

Note that a kNN using a local approach (Fig. 2, kNN - LCPN) comprises one classifier at each level in the hierarchy to choose between its child nodes using sub-datasets at each step. The kNN-LCPN placed in the root node performs one

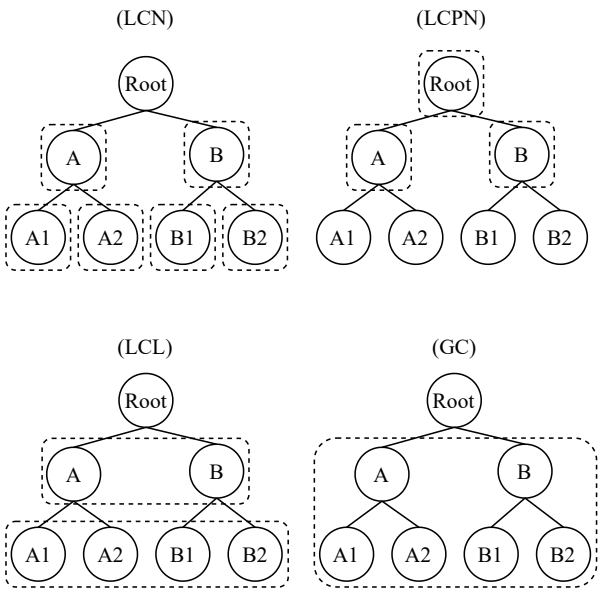

Fig. 1. Hierarchical classification approaches (circles represent classes and dashed squares enclose classes to be predicted by a classifier) (Adapted from [6]).

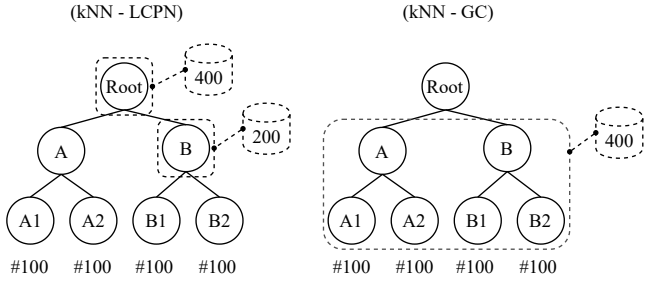

Fig. 2. Illustrative comparison between hierarchical classification approaches. While kNN-LCPN needs to compare instances at each level in the hierarchy, the kNN-GC performs only one comparison for the whole tree.

comparison using 400 instances to predict A or B. After that, another kNN-LCPN placed in the B node (the predicted one in the last step) repeats the process with its child nodes, performing another comparison with another 200 instances.

In contrast, a global approach kNN (Fig. 2, kNN - GC) performs only one comparison for the whole tree using the 400 instances available. It is important to note that this difference is dependent on the depth of the tree and tends to be even larger in deeper hierarchies.

B. Data Stream Classification

Data stream classification differs from traditional classification because data becomes available over time, and thus, models must be updated accordingly. Furthermore, classifiers tailored for learning from data streams must assume that the data is potentially unbounded. Due to the temporal and unbounded traits of data streams, several constraints are imposed on data stream classifiers [4]:

- Single-pass: each instance must be processed just once and cannot be reused;
- Bounded memory: learners must be able to work with a limited amount of memory. If memory consumption increases over time, they will be unable to process streams that are larger than the available memory. In practice, there is no problem with buffering instances and relaxing the previous constraint, yet, increases in memory consumption are forbidden;
- Limited time: the amount of time used to process an instance must be restrained. Otherwise, arriving instances will be buffered and put in jeopardy the aforementioned constraints;
- Readiness: methods must be ready to predict at any time, using all the data seen before to build the best model possible at that time.

Fig. 3 depicts an overview of how data stream classification takes place given the aforementioned constraints. The input data is obtained from the data stream (a), used for testing (b), and, only after that, for training to update the model (c). This is an iterative process, as input data is continuously arriving. The single-pass processing is achieved by discarding each instance after it has been processed. The constraints concerning limited resources can be achieved by

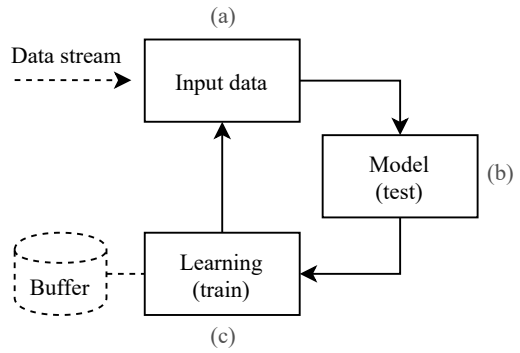

Fig. 3. Illustrative scheme of one possible method for data stream classification. An input data is obtained from the data stream (a), tested (b), incorporated into the model (c), and discarded (or optionally buffered for a short time); then, the cycle starts again.

setting upper bounds on (c). The readiness is represented in (b), as model predictions can be requested at any time.

In data stream classification, the relationship between predictive and target features may change over time, giving rise to a phenomenon named concept drift [7]. Thus, classification models should be able to detect and adapt to these drifts. As a result, data stream classifiers are often coupled with techniques that allow them to either (i) detect that a drift occurred, thus allowing a model reset, or (ii) continuously adapt the model as new data becomes available, usually via windows that reflect most recent data [8].

Regarding validation, data stream classification methods are usually assessed using the prequential assessing method, an interleaved test-then-train strategy, where each instance is used to test the model (predicting and using evaluation metrics) before it is used for training and updating the model [8].

For more information on data stream mining, the interested reader is referred to [9], [10].

III. Related works

In the work [6], the authors presented a comprehensive analysis of how hierarchical classification methods previously reported in the literature performed when compared to non-hierarchical approaches (flat methods).

The analysis comprised 31 methods and depicted that in 90% of the experiments, hierarchical classifiers achieved superior results than flat methods in batch learning, while data stream mining is not mentioned.

On the other hand, recent surveys on data stream classification do not even mention hierarchical classification [9], [11].

Recently, studies published across different research areas proposed methods improperly associated with the hierarchical classification of data streams. For instance, studies have proposed methods regarding hierarchical classification of data streams, but either (i) use a batch configuration for training using the entire dataset, or (ii) do not consider any changes in the data distribution [12]–[16]. In other words, those methods are actually hierarchical classification methods in which the data source was a stream, but it is assumed to be stationary, and models are not updated.

An exception is the work of [17], in which the authors proposed an incremental method for the hierarchical classification of data streams and obtained competitive results compared to established methods in both areas. Their method is based on the traditional k-Nearest Neighbors (kNN) technique [5], represents the data hierarchically, and classifies new data using a top-down strategy within the hierarchy. The proposed algorithm is described as SPP, NMLNP, uses a local approach (LCPN), and a memory buffer on nodes to forget instances. After a predetermined number of initial instances used for training (burnout window), the stream is processed on an instance basis, thus following a prequential protocol.

It is noteworthy that kNN has also been applied in other hierarchical classification proposals, usually following a local approach (LCN or LCPN) [6], [18]. Likewise, the kNN method has also been used in data stream classification. Therefore, the traditional algorithm needs to be adapted to work with the time and memory constraints by, for example, forgetting older instances as the stream progresses [19]–[22].

The previously cited method proposed in [17] successfully merged both areas (data stream classification and hierarchical classification) but presents limitations in the data stream classification scenario since the computational cost for classifying new instances is dependent on the number of instances that the model stores.

In this work, we reduce this dependence by applying a global approach instead of a local one, processing new instances performing fewer comparisons to find the nearest neighbors. As mentioned above, the global kNN is more complex but smaller in computational resources than a local kNN since it represents a single global model against many possible local ones [6]. The next section details our proposal.

IV. The Adaptive Global k-Nearest Neighbors for Hierarchical Data Stream Classification

Our proposal, hereafter referred to as Global k-Nearest Neighbors for Hierarchical Data Streams (*Global kNN-hDS*), is an adaptive method for the hierarchical classification of data streams based on the traditional k-Nearest Neighbors (kNN) technique [5].

The method performs single path predictions (SPP) and mandatory leaf-node predictions (MLNP) and uses a global classification (GC) approach. *Global kNN-hDS* processes new instances with less computational efforts when compared to local approaches.

The global approach is achieved by selecting the kNN of an instance and comparing the labels for each level, picking those instances with the most frequent label for that level. Next, the process is repeated for the next level of the label path until it reaches a leaf node.

Fig. 4 illustrates this process with label paths extracted from an instrument dataset using $k = 7$.

The "Chordophone" label is chosen in the first level because it is the most frequent between the nearest neighbors,

then five label paths remain. In the second level, "Arco" is the most frequent label and three label paths remain. The process is repeated at the third level, resulting in the label path "Chordophone.Arco.Violin" for that given instance.

The proposed method is also adaptive since it uses a sliding window as a mechanism to forget older data. The method implements the window through a memory buffer at each leaf node in the tree. Thus, the method inherently responds to concept drifts.

Fig. 5 shows the algorithm of the proposed adaptive global k-Nearest Neighbors method for hierarchical classification of data streams (*Global kNN-hDS*).

We assume that we are inside the prequential loop; thus, the algorithm receives an instance as input, in addition to the number of nearest neighbors and the buffer size.

From lines 1 to 7, we obtain all the instances temporarily stored in all the descendant nodes of the root node. In lines 8-10, we calculate the Euclidean distance between the new instance and the data stored at the tree (the possible nearest neighbors obtained in the previous step). Thus, we obtain the k-nearest neighbors and their labels by ordering the possible nearest neighbors by the Euclidean distance (lines 11-12).

In line 13, the model predicts the label path by choosing the label returned by the function *mostFrequentInLevel()* (as previously described in Fig. 4).

Next, we complete the prequential method by incorporating the new data into the original dataset using its correct label (line 14).

In lines 15-17, we test whether the number of instances in each node exceeds the stipulated buffer size. If so, we apply a sliding window strategy by forgetting the older instances of that node, performing the proposed adaptive learning, and assuring that the method can work with a constrained and constant memory amount.

(Level 1)	**Chordophone**	>	*Pizzicato*	>	*Guitar*
	Chordophone	>	*Arco*	>	*Violin*
	Aerophone	>	*Wood*	>	*Flute*
	Chordophone	>	*Pizzicato*	>	*Guitar*
	Chordophone	>	*Arco*	>	*Violin*
	Chordophone	>	*Arco*	>	*Cello*
	Aerophone	>	*Wood*	>	*Flute*
(Level 2)	*Chordophone*	>	Pizzicato	>	*Guitar*
	Chordophone	>	**Arco**	>	*Violin*
	Chordophone	>	Pizzicato	>	*Guitar*
	Chordophone	>	**Arco**	>	*Violin*
	Chordophone	>	**Arco**	>	*Cello*
(Level 3)	*Chordophone*	>	*Arco*	>	**Violin**
	Chordophone	>	*Arco*	>	**Violin**
	Chordophone	>	*Arco*	>	Cello

Fig. 4. Global approach illustrated in an instrument dataset using $k = 7$ on the kNN. The most frequent label in each level is chosen, and the other label paths are discarded. The process repeats until it reaches a leaf node.

Algorithm
Global kNN-hDS - Adaptive global k-Nearest Neighbors for hierarchical classification of data streams

Input
k: number of nearest neighbors
buffer_size: maximum number of instances to be stored in each node
instance: a hierarchically labeled instance

```
1  for child_node in tree.descendants do
2      if child_node.data then
3          for target_data in child_node.data do
4              targets_list.append(current_node.id,target_data);
5          end
6      end
7  end
8  for target in targets_list do
9      target.append(euclideanDistance(new_data,target));
10 end
11 targets_list = sorted(targets_list, key=euclidean_distance);
12 knn = targets_list[0:k];
13 predicted_labels = mostFrequentInLevel(knn.labels);
14 correct_node.data.append(new_data);
15 if (len(correct_node.data) > buffer_size then
16     correct_node.data.pop(0);
17 end
```

Fig. 5. Global kNN-hDS - Adaptive global k-Nearest Neighbors for hierarchical classification of data streams

V. Analysis

This section defines the experimental protocol, comprising the description of the datasets, evaluation protocol, and the results obtained during a comparison with related work.

A. Experimental protocol

We used 16 hierarchically labeled datasets obtained from the literature described in Table I.

We pre-processed HAR and Insects dataset variants to create a label hierarchy (pre-existing but not explicit). The FMA (Free Music Archive) dataset was pre-processed to include only instances with full depth labeling (FD) and single paths of labels (SPL). These datasets contain different features, instances, and domains, thus allowing the assessment of how our proposal behaves in different scenarios.

During the experiments, classifiers were assessed in terms of hierarchical F-measure [27]. Like traditional classification metrics, the hierarchical F-Measure (hF) relies on hierarchical precision and recall components, but instances are associated with a path of labels, and the entire path is evaluated. The hierarchical F-Measure is depicted in Equation 1, while its precision (hP) and recall (hR) components are described in Equations 2 and 3, respectively. In both precision and recall metrics, $\widehat{P}_i$ is the set of labels predicted for the i-th instance, and $\widehat{T}_i$ is its corresponding ground-truth label set.

$$hF = \frac{2 \times hP \times hR}{hP + hR} \tag{1}$$

TABLE I

DATASETS USED IN THE EXPERIMENT.

Dataset	# of Instances	# of Features	# of Classes	Reference
Entomology	21,722	33	14	[17]
FMA	10,761	518	109	[23]
HAR	10,299	561	6	[24], [25]
Ichthyology	22,444	15	15	[17]
Insects (i-bal)	57,018	33	6	[26]
Insects (i-imb)	452,044	33	6	[26]
Insects (a-bal)	52,848	33	6	[26]
Insects (a-imb)	355,275	33	6	[26]
Insects (i-g-bal)	24,15	33	6	[26]
Insects (i-g-imb)	143,323	33	6	[26]
Insects (i-a-r-bal)	79,986	33	6	[26]
Insects (i-a-r-imb)	452,044	33	6	[26]
Insects (i-r-bal)	79,986	33	6	[26]
Insects (i-r-imb)	452,044	33	6	[26]
Insects (o-o-c)	905,145	33	24	[26]
Instruments	9,419	30	31	[17]

$$hP = \frac{\sum_i \left| \widehat{P}_i \bigcap \widehat{T}_i \right|}{\sum_i \left| \widehat{P}_i \right|} \tag{2}$$

$$hR = \frac{\sum_i \left| \widehat{P}_i \bigcap \widehat{T}_i \right|}{\sum_i \left| \widehat{T}_i \right|} \tag{3}$$

Furthermore, we measured the time performance by calculating the number of instances that are processed per second.

We compared our proposal (*Global kNN-hDS*) against the local kNN variant proposed in [17], hereafter referred to as *Local kNN-hDS*. We set up both methods with identical parameters (k = 7, buffer size = 100) in all performed tests.

Finally, the results obtained by both methods were assessed using Wilcoxon hypothesis tests [28] with a 95% confidence level according to the protocol provided in [29].

The experiments in this paper were performed using Python 3.7. The proposed script containing the *Global kNN-hDS* method and datasets used in the experiments are available for download at `http://www.ppgia.pucpr.br/~jean.barddal/datasets/Global_kNN-hDS.zip`.

B. Discussion

Table II shows the hierarchical F-measure obtained by both classifiers in the datasets (greater values highlighted in bold).

The *Global kNN-hDS* method obtained better hF rates in 12 out of the 16 datasets. However, hF values are similar across local and global approaches, such that the average difference between them is 0.03%, favoring the global proposal.

Despite the improvements, the Wilcoxon signed-rank test showed that there is no statistical difference between hF rates obtained by the methods ($W = 40.5$, $p\text{-}value = 0.2671$).

TABLE II

HIERARCHICAL F-MEASURE (hF) RATES OBTAINED DURING EXPERIMENTS.

	hF (%)	
Dataset	**Local kNN-hDS [17]**	**Global kNN-hDS**
Entomology	**57.84**	57.81
FMA	24.28	**24.73**
HAR	**96.39**	96.26
Ichthyology	45.93	**46.01**
Insects (i-bal)	84.22	**84.27**
Insects (i-imb)	82.08	**82.17**
Insects (a-bal)	83.28	**83.33**
Insects (a-imb)	81.38	**81.46**
Insects (i-g-bal)	83.52	**83.56**
Insects (i-g-imb)	**87.71**	**87.71**
Insects (i-a-r-bal)	81.81	**81.86**
Insects (i-a-r-imb)	81.42	**81.50**
Insects (i-r-bal)	83.62	**83.65**
Insects (i-r-imb)	81.38	**81.47**
Insects (o-o-c)	**64.01**	63.70
Instruments	**73.59**	73.41
Avg. hF (%)	74.53	**74.56**

In terms of processing time, Table III compares the average number of instances per second processed by both methods in each dataset (greater values highlighted in bold).

The *Global kNN-hDS* method was able to process more instances per second across all datasets, with an average rate of 90.31 instances against 34.80 of the local approach.

We highlight that this value varies according to the number of features in the dataset (it needs more computational effort to calculate distances between two instances, such as in the HAR dataset with 561 features) and the number of classes (more comparisons due to the number of instances in memory buffers of each class node, such as in FMA dataset, with 109 classes).

TABLE III

INSTANCES PER SECOND RATES OBTAINED DURING EXPERIMENTS.

	Instances per second	
Dataset	**Local kNN-hDS [17]**	**Global kNN-hDS**
Entomology	34.88	**54.82**
FMA	1.47	**1.78**
HAR	6.53	**8.64**
Ichthyology	54.01	**105.32**
Insects (i-bal)	39.03	**118.30**
Insects (i-imb)	40.37	**117.28**
Insects (a-bal)	40.57	**117.71**
Insects (a-imb)	40.25	**114.12**
Insects (i-g-bal)	44.71	**131.11**
Insects (i-g-imb)	41.51	**121.43**
Insects (i-a-r-bal)	40.95	**119.61**
Insects (i-a-r-imb)	40.54	**118.43**
Insects (i-r-bal)	40.76	**119.52**
Insects (i-r-imb)	40.40	**118.73**
Insects (o-o-c)	20.34	**34.76**
Instruments	30.43	**43.39**
Avg. inst/sec	34.80	**90.31**

On average, the global method was able to process 2.60 times more instances than the local approach. The FMA, HAR, Instruments, and Entomology datasets (in that order) resulted in the smallest differences in the rate of instances per second obtained by both methods, equal to 1.38 times. In contrast, in most insect datasets (except for "out-of-control"), the global approach was able to process 2.92 times more instances than the local approach.

To conclude our analysis, we performed a one-tailed Wilcoxon test to verify whether the rates of instances processed by the global approach are greater than the rates of the local method. The test indicated a statistical difference between instances per second rates obtained by both methods ($W = 136.0$, $p\text{-}value = 0.0002$) and showed that our proposal is significantly faster when compared to the local approach.

VI. Conclusion

This paper proposed a global and adaptive approach for the hierarchical classification of data streams based on k-Nearest Neighbors. We compared our proposal with a local kNN variant, and results showed that it is statistically faster while achieving similar prediction rates.

Furthermore, as a byproduct of this research, a Python implementation of our method and the datasets used in the experiments are made publicly available. We believe these are essential aspects to serve as baselines for future studies on the hierarchical classification of data streams.

In future works, we are interested in analyzing the behavior of the method with other window types. Also, we intend to study new models to decrease the processing time in kNN distance computations and new strategies to forget data considering the application of traditional drift detectors to increase the responsiveness to changes in the data distribution.

References

[1] N. I. Yassin, S. Omran, E. M. El Houby, and H. Allam, "Machine learning techniques for breast cancer computer aided diagnosis using different image modalities: A systematic review," *Computer methods and programs in biomedicine*, vol. 156, pp. 25–45, 2018.

[2] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *ACM SIGKDD Explorations Newsletter*, vol. 19, no. 1, pp. 22–36, 2017.

[3] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 3, no. 3, pp. 1–13, 2007.

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

[5] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine learning*, vol. 6, no. 1, pp. 37–66, 1991.

[6] 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.

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

[8] 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.

[9] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39–57, 2017.

[10] A. Cano, "A survey on graphic processing unit computing for large-scale data mining," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 1, p. e1232, 2018.

[11] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou *et al.*, "Open challenges for data stream mining research," *ACM SIGKDD explorations newsletter*, vol. 16, no. 1, pp. 1–10, 2014.

[12] Y. Wang, Z. Gong, and J. Guo, "Hierarchical classification of business information on the web using incremental learning," in *2009 IEEE International Conference on e-Business Engineering*. IEEE, 2009, pp. 303–309.

[13] H. Purohit, A. Hampton, S. Bhatt, V. L. Shalin, A. P. Sheth, and J. M. Flach, "Identifying seekers and suppliers in social media communities to support crisis coordination," *Computer Supported Cooperative Work (CSCW)*, vol. 23, no. 4-6, pp. 513–545, 2014.

[14] S. A. Khowaja, A. G. Prabono, F. Setiawan, B. N. Yahya, and S.-L. Lee, "Contextual activity based healthcare internet of things, services, and people (hiotsp): An architectural framework for healthcare monitoring using wearable sensors," *Computer Networks*, vol. 145, pp. 190–206, 2018.

[15] L. Cao, Y. Wang, B. Zhang, Q. Jin, and A. V. Vasilakos, "Gchar: An efficient group-based context—aware human activity recognition on smartphone," *Journal of Parallel and Distributed Computing*, vol. 118, pp. 67–80, 2018.

[16] K.-Y. Huang, C.-H. Wu, Q.-B. Hong, M.-H. Su, and Y.-H. Chen, "Speech emotion recognition using deep neural network considering verbal and nonverbal speech sounds," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5866–5870.

[17] 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.

[18] M. Aly, "Survey on multiclass classification methods," *Neural Netw*, vol. 19, pp. 1–9, 2005.

[19] M. Khan, Q. Ding, and W. Perrizo, "k-nearest neighbor classification on spatial data streams using p-trees," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2002, pp. 517–528.

[20] P. Zhang, B. J. Gao, X. Zhu, and L. Guo, "Enabling fast lazy learning for data streams," in *2011 IEEE 11th International Conference on Data Mining*. IEEE, 2011, pp. 932–941.

[21] J. P. Barddal, H. M. Gomes, J. Granatyr, A. de Souza Britto, and F. Enembreck, "Overcoming feature drifts via dynamic feature weighted k-nearest neighbor learning," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 2186–2191.

[22] J. P. Barddal, H. Murilo Gomes, F. Enembreck, B. Pfahringer, and A. Bifet, "On dynamic feature weighting for feature drifting data streams," in *Machine Learning and Knowledge Discovery in Databases*, P. Frasconi, N. Landwehr, G. Manco, and J. Vreeken, Eds. Cham: Springer International Publishing, 2016, pp. 129–144.

[23] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "Fma: A dataset for music analysis," *arXiv preprint arXiv:1612.01840*, 2016.

[24] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones." in *Esann*, 2013.

[25] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[26] 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.

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

[28] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in statistics*. Springer, 1992, pp. 196–202.

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