Growing Hierarchical Trees for Data Stream Clustering and Visualization

Nhat-Quang Doan
ICT Department
University of Science and Technology of Hanoi
18 Hoang Quoc Viet Str, Cau Giay
Hanoi, Vietnam
Email: doan-nhat.quang@usth.edu.vn

Mohammed Ghesmoune
Hanane Azzag
Mustapha Lebbah
University of Paris 13, Sorbonne Paris City
LIPN-UMR 7030 - CNRS
99, av. J-B Clément – F-93430 Villetaneuse, France
Email: firstname.secondname@lipn.univ-paris13.fr

***Abstract*—Data stream clustering aims at studying large volumes of data that arrive continuously and the objective is to build a good clustering of the stream, using a small amount of memory and time. Visualization is still a big challenge for large data streams. In this paper we present a new approach using a hierarchical and topological structure (or network) for both clustering and visualization. The topological network is represented by a graph in which each neuron represents a set of similar data points and neighbor neurons are connected by edges. The hierarchical component consists of multiple tree-like hierarchic of clusters which allow to describe the evolution of data stream, and then analyze explicitly their similarity. This adaptive structure can be exploited by descending top-down from the topological level to any hierarchical level. The performance of the proposed algorithm is evaluated on both synthetic and real-world datasets.**

Keywords*—*data stream, clustering, visualization, hierarchical tree, neural network, neural gas

I. Introduction

A data stream is a sequence of potentially infinite, non-stationary (i.e., the probability distribution of the unknown data generation process may change over time) data arriving continuously. In this case, random data access is not feasible and newly arriving data are discarded after a single pass through the learning process thus data storing becomes more impractical.

Streaming algorithms have been introduced to find data pattern reflecting continuously online data in real-time. Besides, the streaming algorithms are capable of learning fast and incrementally in order to overcome memory and time limitations. In the literature, many streaming algorithms have been adapted from clustering algorithms, e.g., the density-based method DbScan [1], [2], the partitioning method k-means [3], or the message passing-based method AP [4], the evolving algorithm G-Stream [5].

Data stream clustering can be processed for further analysis of dynamic patterns evolving over time, event tracking or future change trend detection. An attractive solution is to visualize data streams to reveal insight that may suggest further experiments to conduct. An interactive visualization should be able to express incremental information projected directly onto a low dimensional subspace. There are two main issues for visualizing streaming data: the amount of data to be displayed and the display of newly arriving data.

To address both data stream clustering and visualization at the same time, we propose the growing heuristic topological and hierarchical structure GH-Stream (Growing Hierarchical Trees over Data Stream), a variant of G-Stream which does not require the number of clusters to be specified beforehand. This type of structure consists of a topological network and multiple trees which can be exploited by descending from a general part to any particular part, i.e. from the topological level to any level of a hierarchical tree. When new data arrive, some mechanisms are used to remove or add new nodes (neurons in the topological level or tree nodes in the hierarchical level). This facilitates the visual task and adapts to the data change trends. Thus, the main contribution of this work is to present an incrementally hierarchical and topological structure that can be used to analyze data streams at any particular step.

The remainder of this paper is organized as follows: in Section II, we briefly review related works about data stream methods. Section III is devoted to introducing the GH-Stream principle and algorithm. In Section IV, we perform some experiments and discuss the results. The paper ends with Section V, which includes a general conclusion and annonces possible future works.

II. Related Works

This section briefly presents previous works on data stream clustering, and highlights the most relevant algorithms proposed in the literature. Table I summarizes the main features of different algorithms for data stream clustering: the basic clustering algorithm, whether the algorithm provides a hierarchical and/or topological structure, whether the links (if they exist) between clusters are weighted, how many phases an algorithm adopts (online and offline), the network adaptation (remove, merge, and split clusters), and whether a *fading* function is used.

Relating to the clustering problems, we can list several algorithms such as: *CluStream* [6] and *DenStream* [1] use a temporal extension of the *Clustering Feature vector* [7] (called *micro-clusters*) to maintain statistical summaries about data locality and timestamps during the online phase. By creating two kinds of micro-clusters (*potential* and *outlier micro-clusters*), *DenStream* overcomes one of the drawbacks of *CluStream*, its sensitivity to noise. In the offline phase, the micro-clusters found during the online phase are considered as *pseudo-points* and will be passed to a variant of k-means in the *CluStream* algorithm (resp. to a variant of DbScan

TABLE I: Comparison between algorithms (WL: weighted links, 2 phases : online+offline).

Algorithms	based on	hierarchy	topology	WL	phases	remove	merge	split	fade
AING	NG	✗	✓	✗	online	✗	✓	✗	✗
CluStream	*k*-means	✗	✗	✗	2 phases	✓	offline	✗	✗
DenStream	DbScan	✗	✗	✗	2 phases	✓	offline	✗	✓
SOStream	DbScan, SOM	✗	✗	✗	online	✓	✓	✗	✓
E-Stream	*k*-means	✗	✗	✗	2 phases	✓	✓	✓	✓
SVStream	SVC, SVDD	✗	✗	✗	online	✓	✓	✓	✓
StreamKM++	*k*-means	✓	✗	✗	2 phases	✓	✓	✓	✓
StrAP	AP	✓	✗	✗	2 phases	✓	✗	✗	✓
ODAC	top-down tree	✓	✗	✗	online	✗	✗	✓	✗
G-Stream	NG	✗	✓	✓	online	✓	✗	✗	✓
GH-Stream	NG	✓	✓	✓	online	✓	✗	✗	✓

in the *DenStream* algorithm) in order to determine the final clusters. *StreamKM++* [3] maintains a small outline of the input data using the *merge-and-reduce* technique. The merge step is performed by means of a data structure, named the *bucket set*. The reduce step is performed by a significantly different summary data structure, the *coreset tree*.

ClusTree [8] is an anytime algorithm that organizes micro-clusters in a tree structure for faster access and automatically adapts micro-cluster sizes based on the variance of the assigned data points. Any clustering algorithm, e.g. k-means or DbScan, can be used in its offline phase. *SOStream* [2] is a density-based clustering algorithm inspired by both the principle of the DbScan algorithm and that of self-organizing maps (SOM) [9]. *E-Stream* [10] classifies the evolution of data into five categories: appearance, disappearance, self evolution, merge, and split. It uses another data structure for saving summary statistics, named α-bin histogram. *StrAP* [4], an extension of the Affinity Propagation algorithm for data streams, uses a reservoir for saving potential outliers.

In *SVStream* [11], the data elements of a stream are mapped into a kernel space, and the support vectors are used as the summary information of the historical elements to construct cluster boundaries of arbitrary shape. *SVStream* has been based on support vector clustering (SVC) and support vector domain description (SVDD) [11]. *AING* [12], an incremental GNG that learns automatically the distance thresholds of nodes based on its neighbors and data points assigned to the node of interest. It merges nodes when their number reaches a given *upper-bound*. *ODAC* [13] aims to build a tree-like structure using the agglomerative strategy. It uses correlation-based dissimilarity measure between time series over a data stream in order to split or merge tree subtrees. G-Stream [5] is recently developped based on a growing neural network [14] to learn data streams but it lacks a visual component.

For the visual aspect, a dynamic approach is employed by several methods such as [15]–[17] to visualize data streams evolving over time. The principle is to get the visualization modified constantly in accord with the new data. The modification is triggered by an update and usually expressed through motion. Another approach to address visualization is to create numerous views at different time points. Each view relates to each other and it is possible to show the differences between views.

III. Growing Hierarchical Trees for Data Stream

The implementations of GH-Stream are strongly influenced by clustering tasks and visualization objectives. GH-Stream is developed using several rules from AntTree to add a new hierarchical dimension in G-Stream. In term of human perception, a hierarchical tree is an efficient tool and an optimal representation to represent the nature of data structure. In this way, we are interested in particularly in AntTree [18] to model the ability of artificial ants to build automatically complex structures. Due to the self-assembly rules defined by AntTree, this approach can be adaptive to the self-organizing models.

As an online clustering algorithm, GH-Stream is able to find data patterns in large datasets evolving over time. Furthermore, as a visualization framework, GH-Stream provides a solution to stream data abstraction and changes necessarily over time to reflect the stream evolution. With a dynamic two-level structure, we present an evolving visualization of a continuous data stream. Such hierarchical and topological structures have been studied for visualization in [19], [20]. In this paper, we will show how to benefit from this structure for visualizing data streams. We create various views for different time intervals. Here, the new view is modified from the old one ensuring that the user is able to perceive the differences between the two.

A. Dynamic multi-level structure for clustering

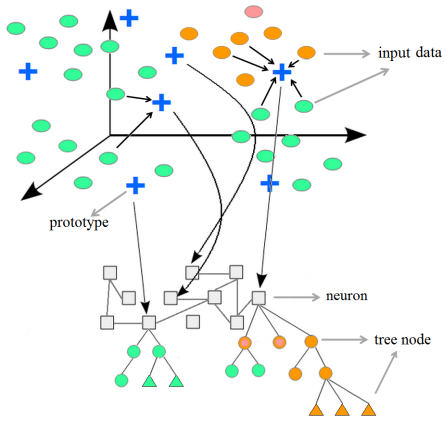


Fig. 1: Hierarchical and topological structure.

The proposed structure is illustrated in Fig. 1. Several elements can be found in this structure:

- *Network* describes the topological space where data will be mapped discretely. Note that this network will extend if new data points arrive.

- *Neuron* or network node (square node) represents a cluster in the topological space. Each neuron is associated with a prototype (or a weight vector) to which input data are assigned, and with a hierarchical *tree*. This neuron is also the tree root to which the first tree nodes are going to connect. Here the tree structure will evolve if new data points arrive.
- *Topological link* is created between a pair of neurons if they are considered as neighbors due to a given neighborhood function. A variable exists to control this type of link.
- *Tree node* (circle or triangle node) corresponds respectively to a data point in the projected space. Data in old streams are represented by circle nodes and newly arriving data by triangle nodes.
- *Hierarchical link* is created between a pair of tree nodes if it satisfies a similarity test.

The proposed structure has a feature for effective visualization of tree nodes which represent input data. A test is applied in order to verify the similarity between a pair of data. A hierarchical link is formed if and only if these two are similar; otherwise a new subtree is created and considered as a new sub-cluster. Thus data from new streams continuously and recursively grouped in a sub-tree are close to those from old streams. GH-Stream is able to detect clusters and represent these clusters in a topological and hierarchical structure. The confidence of each cluster may be easily observed because of hierarchical relations between data.

B. Algorithm

In this section, we give the algorithm details of GH-Stream. Suppose that a data stream is denoted by $\mathcal{DS} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ of *m* (potentially infinite) data streams arriving in times $T_1, T_2, ..., T_m$, where $\mathbf{x}_i = (x_i^1, x_i^2, ...x_i^d)$ is a vector in $\mathbb{R}^d$. The proposed network $\mathcal{W}$ consists of neurons, each neuron $k \in \mathcal{W}$ is associated with a tree $tree_k$ and with a prototype $\mathbf{w}_k \in \mathbb{R}^d$.

In Algorithm 1, GH-Stream is divided into four main steps: 1) initialization, 2) assignement, 3) tree construction, and 4) adaptation.

1) ***Initialization****:* At the beginning, GH-Stream is randomly initialized with only a 2-tree network in which these neurons are connected by a topological link. During the learning process, the network evolves and adapts to cover the data patterns. Thus, the GH-Stream network is more flexible and able to overcome the sensitivity to topology.

2) ***Assignment****:* As a new data point is reached, the nearest and the second-nearest neurons are identified, linked by an edge, and the nearest neuron and its topological neighbors are moved toward the data point. This assures that the quantization error of the current stream is minimized in regard to the data assuming that the prototype vectors are constant.

$$k_0 = \arg \min_{k=1,..,K} \|\mathbf{x}_i - \mathbf{w}_k\|^2 \quad (1)$$

where K is the current number of trees in the network.

Algorithm 1 GH-Stream

Require: $\mathcal{DS} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$
Ensure: $tree, \mathcal{W}$
1: Initialize network $\mathcal{W}$ associated with two trees and create an empty reservoir $\mathcal{R}$ (a list contains disconnected tree nodes)
2: **while** there is a data point to proceed **do**
3: $\mathbf{x}_i \leftarrow$ the next point in the current data stream
4: find k_0 using Equation 1 and find $tree_0$, the tree associated to k_0
5: $constructTree(tree_0, \mathbf{x}_i, \mathcal{R})$
6: $adaptation(tree, \mathbf{x}_i)$;
7: remove outdated and isolated neurons (trees), and put all disconnected tree nodes into $\mathcal{R}$
8: **if** $\mathcal{R}$ is full **then**
9: $constructTree(tree_0, \mathbf{x}_i, \mathcal{R})$
10: **end if**
11: **end while**

3) ***Tree construction****:* Here we show how to adapt the self-assembly rules inspired by AntTree. During the learning process, the status of a tree node can be varied due to the connecting or disconnecting rules. Therefore, we define three possibilities for the tree node status:

1) *Initial* : when a new data point arrives, it has initial status as the default one;
2) *Connected* : A tree node is currently connected to another node;
3) *Disconnected* : A tree node which was connected at least once but now is disconnected. We denote a resevoir +$\mathcal{R}$ to contain all disconnected nodes.

Once the data has been assigned to the nearest tree, they will take part in building trees. Data has to pass through the similarity test in Algorithms 2 and 3. At first, the tree is empty and since the similarity test can only be computed with at least two tree nodes, then the first two tree nodes are automatically connected to a tree as in the first test (Line 1 in Algorithm 2). The second test (Line 6 in Alogorithm 2) is used to find the best position in the hierarchical structure for each data point. It can be either to create a new subtree at the current tree node or pass top-down to become a leaf node.

During the learning process, there is a chance that objects could be *disconnected*. Concerning the disconnection, there are two distinct cases:

1) remove a tree node (Line 7 in Algorithm 1),
2) disconnect tree node(s) (Line 7 in Algorithm 2).

Whenever a tree node is disconnected from a tree, we have to check whether other child nodes exist in $subtree_{\mathbf{x}_i}$. If it is the case, we disconnect all of them or for the specific $subtree_{\mathbf{x}_i}$. A simple example of disconnection for a group of nodes (or subtree) is depicted in Fig. 2a. Given $tree_{old}$ as in this example, the tree node $\mathbf{x}$ consisting of three violet nodes is disconnected from this tree. All the nodes connected to $\mathbf{x}$ must be recursively disconnected too (Line 8 in Algorithm 1 or Line 7 in Algorithm 2); it applies to two child nodes of $\mathbf{x}$. Therefore $subtree_{\mathbf{x}}$ has *disconnected* status and is immediately put onto the list $\mathcal{R}$.

Suppose that a disconnected $\mathbf{x}_i$ becomes *connected* at a

Algorithm 2 constructTree

Require: $tree_0, \mathbf{x}_i$
Ensure: $tree_0$
1: **if** less than 2 tree nodes are connected to the root of $tree_0$ **then**
2: connect $\mathbf{x_i}$ to the root of $tree_0$
3: **else**
4: $T_{root} = max(d(\mathbf{x}_i, \mathbf{x}_j))$ {where $\mathbf{x_i}$ and $\mathbf{x}_j$ are any pair of data connected to the root of $tree_k$; $d(\mathbf{x}, \mathbf{x}_j) = ||\mathbf{x}_i - \mathbf{x}_j||^2$, $\mathbf{x}, \mathbf{x}_j$ are normalized}
5: $\mathbf{x}^+ = \arg\min_r d(\mathbf{x}_i - \mathbf{x}_r)$ {$\forall \mathbf{x}_r$ is connected to the root of $tree_0$}
6: **if** $d(\mathbf{x_i}, \mathbf{x}^+) > T_{root}$ **then**
7: disconnect $\mathbf{x}^+$ from the root {disconnect recursively $subtree_{\mathbf{x}^+}$}
8: put $subtree_{\mathbf{x}^+}$ into $\mathcal{R}$
9: **if** $\mathbf{x}_i$ is *disconnected* **then**
10: $subtree_{\mathbf{x}_i} \leftarrow$ all nodes recursively connected to $\mathbf{x}_i$ before its disconnection
11: **end if**
12: connect $\mathbf{x}_i$ and $subtree_{\mathbf{x}_i}$ to $\mathbf{x}^{pos}$ {The subtree structure is kept as it was before the disconnection}
13: **else**
14: $connectRecursive(tree_0, \mathbf{x}_i, \mathbf{x}^+)$
15: **end if**
16: **end if**

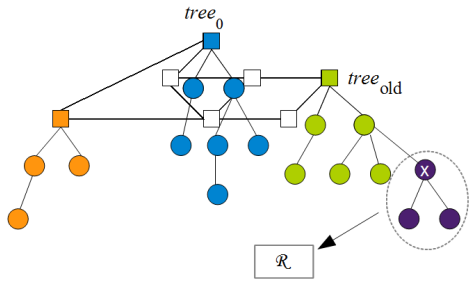
Algorithm 3 connectRecursive

Require: $tree_0, \mathbf{x}_i, \mathbf{x}^{pos}$
Ensure: $tree_0$
1: **if** no tree nodes connected to $\mathbf{x}^{pos}$ **then**
2: $tree_0$ = connectSubTree($tree_0, \mathbf{x}_i, \mathbf{x}^{pos}$)
3: **else**
4: $\mathbf{x}^+ = \arg\min_{\mathbf{x}_r} d(\mathbf{x}_i, \mathbf{x}_r)$ {$\forall \mathbf{x}_r$ is connected to $\mathbf{x}^{pos}$}
5: **if** $d(\mathbf{x}_i, \mathbf{x}^{pos}) > d(\mathbf{x}_i, \mathbf{x}^+)$ **then**
6: $connectRecursive(tree_0, \mathbf{x}_i, \mathbf{x}^+)$
7: **else**
8: **if** $\mathbf{x}_i$ is *disconnected* **then**
9: $subtree_{\mathbf{x}_i} \leftarrow$ all nodes recursively connected to $\mathbf{x}_i$ before its disconnection
10: **end if**
11: connect $\mathbf{x}_i$ and $subtree_{\mathbf{x}_i}$ to $\mathbf{x}^{pos}$ {The subtree structure is kept as it was before the disconnection}
12: **end if**
13: **end if**

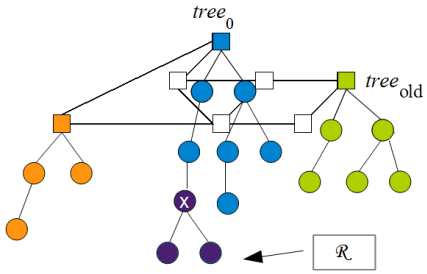
moment, we will keep this subtree structure by re-connecting these child nodes together (Line 12 in Algorithm 2 or Line 11 in Algorithm 3); hence this way can accelerate the learning process. For example, let us take again the example in Fig. 2a. After getting the new assignment, $\mathbf{x}$ is going to connect to $tree_0$. It leads to the fact that the child nodes of $\mathbf{x}$ have $tree_0$ as their best match tree too. We systematically connect this subtree to $tree_0$ and the result is shown in Fig. 2b. It should be recalled that this subtree is not kept till the end of learning, the nodes in the subtree may be disconnected in next iterations.

*4) **Adaptation**:* Once a data point has been assigned to a prototype, this prototype and its neighbors are adjusted and moved toward the assigned object according to the "winner take most" rule [14]. In most data stream scenarios, more recent data can reflect the emergence of new trends or changes in the data distribution [21].

There are three models of windows commonly studied in the data stream: landmark window, sliding window and



(a) Disconnect $subtree_{\mathbf{x}}$ from $tree_{old}$ and put it into $\mathcal{R}$



(b) Re-connect $subtree_{\mathbf{x}}$ to $tree_0$

Fig. 2: Rules to build hierarchical structure. Neuron is colored according to a majority vote of data gathered within this neuron.

Algorithm 4 adaptation: network adaptation

Require: $tree, subtree_{\mathbf{x}_i}, \mathcal{W}$
Ensure: $tree, \mathcal{W}$
1: $error(tree_0) = error(tree_0) + \|\mathbf{x}_i - \mathbf{w}_0\|^2$
2: move $\mathbf{w}_0$ and its topological neighbors towards $\mathbf{x}_i$

$$\Delta_0 = \sum_{j \in subtree_{\mathbf{x}_i}} \epsilon_b(\mathbf{x}_j - \mathbf{w}_0)$$

$$\Delta_r = \sum_{j \in subtree_{\mathbf{x}_i}} \epsilon_r(\mathbf{x}_j - \mathbf{w}_r)$$

$\forall r$ is neighbor to k_0

3: find the second nearest tree $tree_1$ of $\mathbf{x}_i$.
4: **if** $tree_0$ and $tree_1$ are connected by an edge **then**
5: set the age of that edge to 0.
6: **else**
7: create a new edge between them.
8: **end if**
9: remove the edges with an age larger than Max_{age}
10: decrease the error of all neurons
11: find two neurons with the largest accumulated error
12: insert new neurons in the half-way between these two
13: update the edges connecting to these two and decrease their error

damped window. We consider, like many others, the damped window model, in which the weight of each data point decreases exponentially with time T via a fading function $f(T) = 2^{-\lambda_1(T-T_0)}$, where $\lambda_1 > 0$, defines the rate of decay of the weight over time, T denotes the current time and T_0 is the timestamp of the data point. The weight of a neuron is based on data points associated with:

$$W_c = \sum_{i=1}^{m} 2^{-\lambda_1(T-T_{i_0})},$$

where m is the number of points assigned to $tree_c$ at the current time T. If the weight of a neuron is less than a threshold then this node is considered as outdated and then deleted with its links. This task is assured by Line 7 in Algorithm 1.

C. Complexity

Algorithm 1 is repeated n times (n data points) to complete the learning process. For each time, there are three operations: assignment, tree construction, adaptation. The assignment and adaptation processes require one operation for each data point, but the tree construction requires $\log n$ operations. To sum up, GH-Stream has the complexity of $O(n \log n)$.

IV. Experiments

This section is devoted to the experiments to illustrate the proposed model for data stream clustering and visualization.

A. Datasets

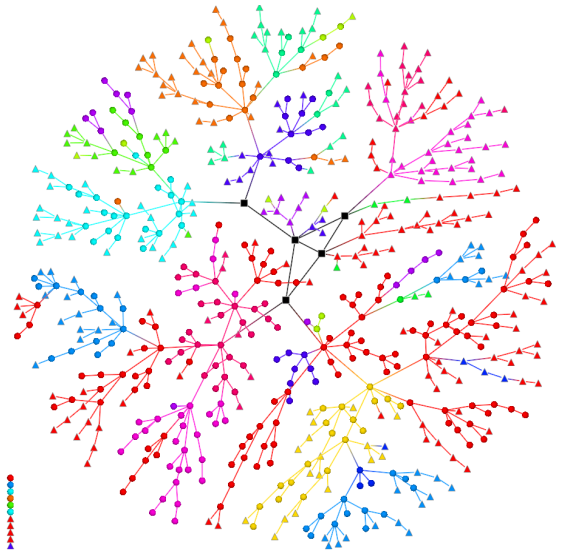
The experiments are performed using real-world and synthetic datasets. Table II overviews all the dataset features. COIL100 is available in http://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php. This dataset contains images of 100 different objects with 72 images per object. DS1 is a synthetic dataset found in http://impca.curtin.edu.au/local/software/synthetic-data-sets.tar.bz2. Hyperplane and Sea are datasets with concept drift. These two are available in http://www.win.tue.nl/~mpechen/data/DriftSets/.Letter4 is generated by a Java code https://github.com/feldob/Token-Cluster-Generator.

TABLE II: Data features

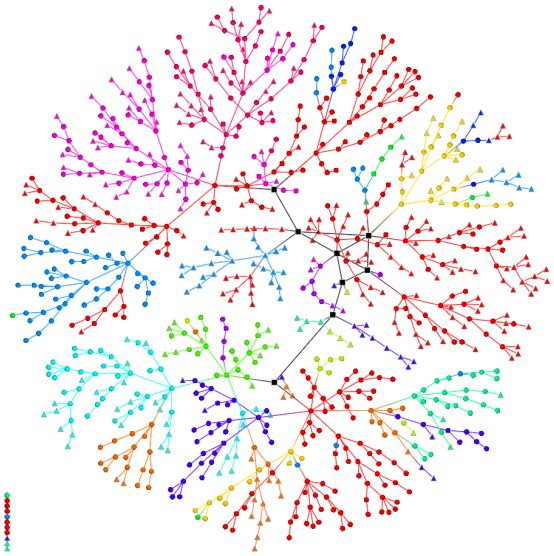
Datasets	# data sample	# feature	# class
COIL100	7,200	1,024	100
DS1	9,153	2	14
Hyperplane	100,000	10	5
Letter4	9,344	2	7
Sea	60,000	3	2

B. Numerical validation

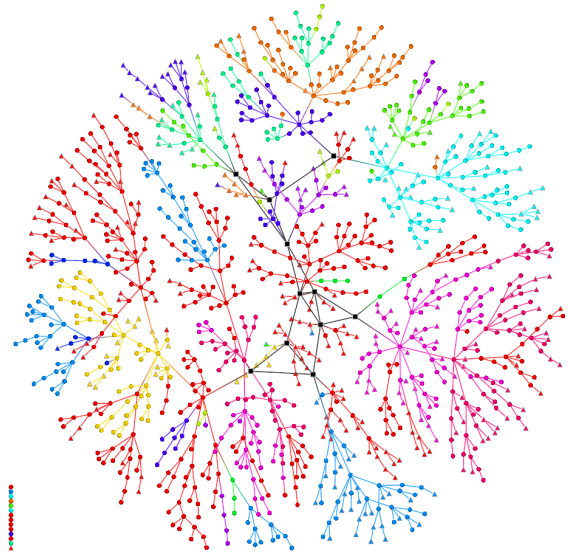
In this subsection, we performed extensive experiments to evaluate the GH-Stream performance for data stream clustering. The experimental parameters for all the datasets are 600 data per stream, epoch = 300 (after 300 iterations, new neurons are added into the network), and MaxAge = 250. Due to the nature of different algorithms, they output different number of clusters at the end of learning process. The GH-Stream



(a) Visualization at T_2 (300 tree nodes in a 5-tree network)

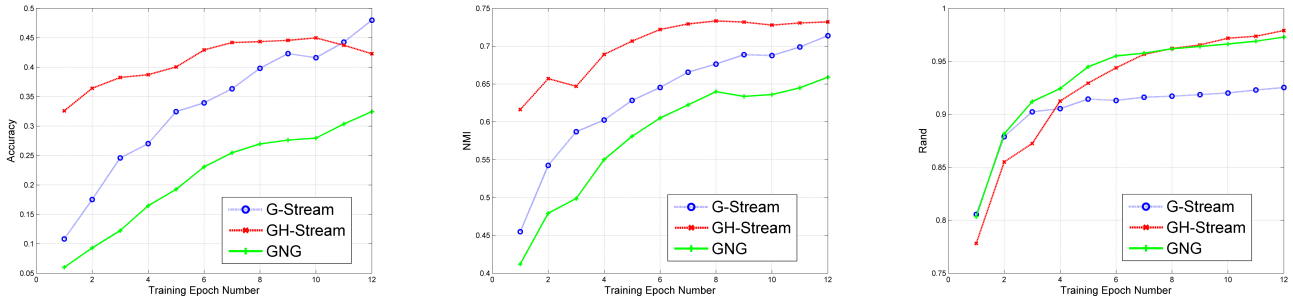


(b) Visualization at T_3 (600 tree nodes in a 8-tree network)



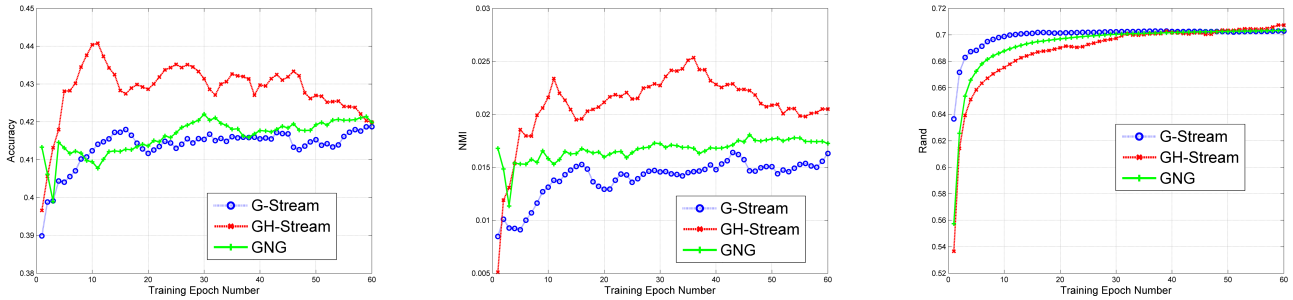
(c) Visualization at T_4 (900 tree nodes in a 11-tree network)

Fig. 5: Visualization of the DS1 dataset. Each class is represented by a single color.



(a) Accuracy (b) NMI (c) Rand index

Fig. 3: Performance of difference methods vs number of epoch over time during the learning process for COIL100



(a) Accuracy (b) NMI (c) Rand index

Fig. 4: Performance of difference methods vs number of epoch over time during the learning process for Hyperplane

TABLE III: Competitive performance of different approaches in term of criterion quality (Accuracy, NMI, and Rand)

Datasets	Criterion	GNG	G-Stream	StreamKM++	CluStream	GH-Stream
COIL100	Acc	0.323 ± 0.009	0.233 ±0.009	**0.427 ± 0.015**	0.373 ± 0.034	0.374 ± 0.005
	NMI	0.655 ± 0.004	0.577 ±0.007	0.606 ±0.0231	0.671 ± 0.011	**0.687 ± 0.007**
	Rand	0.973 ± 0.008	0.921 ±0.012	0.883 ±0.003	0.977 ± 0.001	**0.979 ± 0.001**
DS1	Acc	0.511 ± 0.251	**0.993 ± 0.006**	0.675 ± 0.018	0.701 ± 0.028	0.970 ± 0.010
	NMI	0.491 ± 0.132	0.712 ± 0.004	0.702 ± 0.021	0.723 ± 0.022	**0.730 ± 0.007**
	Rand	0.621 ± 0.122	0.846 ± 0.001	0.844 ± 0.004	0.845 ± 0.007	**0.854 ± 0.001**
HyperPlan	Acc	0.423 ± 0.002	0.396 ± 0.005	0.425 ± 0.000	**0.438 ± 0.008**	0.427 ± 0.003
	NMI	0.018 ± 0.001	0.010 ± 0.002	**0.020 ± 0.000**	0.017 ± 0.004	0.019 ± 0.001
	Rand	0.704 ± 0.000	0.667 ± 0.000	0.603 ± 0.000	0.652 ± 0.001	**0.705 ± 0.000**
Letter4	Acc	0.577 ± 0.201	0.991 ± 0.001	0.687 ± 0.026	0.934 ± 0.026	**0.997 ± 0.003**
	NMI	0.529 ± 0.074	0.607 ± 0.002	0.553 ± 0.022	0.264 ± 0.034	**0.657 ± 0.006**
	Rand	0.686 ± 0.084	0.812 ± 0.001	0.794 ± 0.014	0.341 ± 0.004	**0.818 ± 0.002**
Sea	Acc	0.838 ± 0.002	0.788 ± 0.009	0.824 ± 0.001	0.822 ± 0.006	**0.839 ± 0.002**
	NMI	0.138 ± 0.001	0.146 ± 0.004	**0.164 ± 0.000**	0.158 ± 0.009	0.148 ± 0.001
	Rand	0.470 ± 0.001	**0.507 ± 0.001**	0.470 ± 0.006	0.491 ± 0.003	0.471 ± 0.000

efficiency is evaluated with different algorithms using three quality criteria: Accuracy, Normalized Mutual Information (NMI), and Rand Index. Each criterion should be maximized. Each method is run 10 times with random initializations and the Table III shows the average and standard deviation of quality criteria over these 10 runs.

For the selected datasets, we notice that our GH-Stream provides good clustering results comparing to other methods. GH-Stream generally outperformed the others in term of quality criteria NMI and Rand index in most of cases such as COIL100, DS1, Letter4 datasets. On the other hand, GH-Stream gives comparable accuracy results.

In the direct comparison with G-Stream, GH-Stream uses less input parameters. For more specific, GH-Stream does not require the distance threshold for the similarity test. According to Table III, GH-Stream is better in many cases.

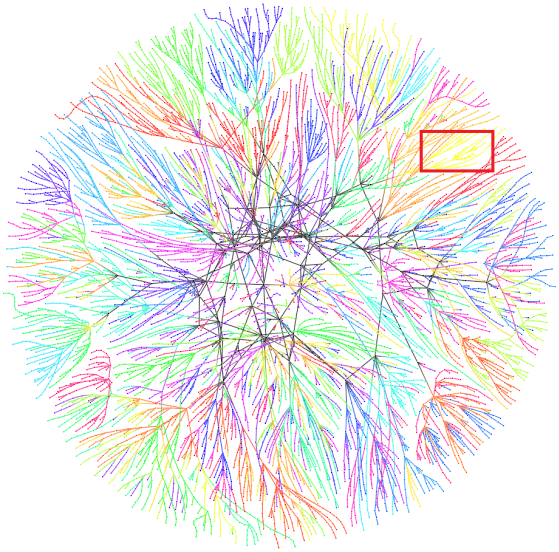
Some other experiments are carried out to analyze the clustering evolution during the learning process. In Fig. 3 and 4, we show the changes in the quality criteria over time for two datasets COIL100 and Hyperplane. The GH-Stream performance dominates over the others G-Stream and GNG with regard to Accuracy and NMI in the beginning of learning. Moreover, GH-Stream also provides further information on data visualization which will be studied thoroughly in the next subsection.

C. Visual validation

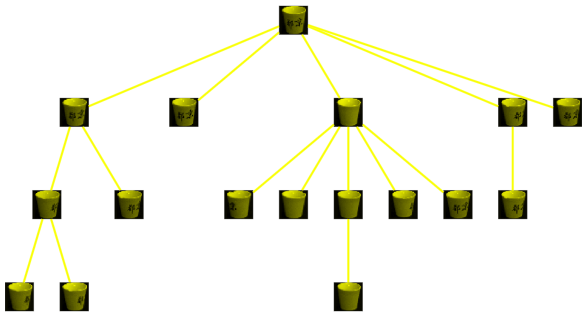
GH-Stream does not only provide a good clustering but also is a good tool for data stream visualization based on Tulip [22] as the framework to visualize the graph. Using GEM layout, we provide multiple views to describe the changes in data stream clustering.

At T_i, there is data from old streams and newly arriving data from the current stream. By using different symbols (square for data from old streams and triangle for data from the current stream), it is quite easy to anticipate the differences among the visualizations provided by GH-Stream. Let us take DS1 in Fig. 5 as a visual example. Data in the reservoir $\mathcal{R}$ can be clearly seen as the isolated nodes (bottom left in this figure). Due to the self-assembly rules, new data arrive and connect to those in the same class. A good classification is indicated by the hierarchical trees or subtrees with the same color found in the proposed structure. However, some regions are colored with different colors due to the fact that in streaming algorithm, data are assigned only once so it is not possible to correct misclassifications. This leads to further investigation. Practically, with an interactive visual tool, users are able to interpret or correct data points from misclassifications by moving their respective subtree and creating new clusters.

Fig. 6 shows the visual result after learning all 7200 images from the dataset COIL100. Many regions with a single color can be observed again as in the previous example. In this case, each data point corresponds to an image. When we zoom as in Fig. 6b to visualize in depth the similarity in the hierarchical structure, we see that images describe a cup belonging to one class; in addition, all 72 images of this class are found in the same tree (the same group). Thus, a couple of questions are raised: what are the neighbors of this tree? Are images found in these neighbor trees similar? To answer this question, another zoom taken from Fig. 6a is shown in Fig. 7. In this figure, images from different objects are put in the same groups but



(a) Complete visualization after the learning process (7200 tree nodes in a 127-tree network)



(b) Zoom sample extracted from Fig. 6a. These images are in the same class "cup"

Fig. 6: Visualization of the COIL100 dataset. Each class is represented by a unique color.

it is notable that they have a similar shape or form such as a cup or a box. To sum up, GH-Stream is a good tool for visual tasks for data stream mining.

V. Conclusion

In this paper, we have proposed a new clustering approach with the objective of data stream visualization which adapts a neural network to a hierarchical and topological space. GH-Stream offers a good clustering as well as an efficient visualization to deal with the data that arrive over time in streams. GH-Stream is able to detect new classes and output satisfying results. We also studied thoroughly the visual results and showed how to exploit the proposed structure.

In the future, GH-Stream will be improved to be more autonomous as it is an objective of data stream mining. Another perspective is to enhance the visualization component for big data so that GH-Stream provides remarkably more degrees of freedom. One promising direction is to employ TreeMap in order to overcome the complexity and redundancy problems in visualization.

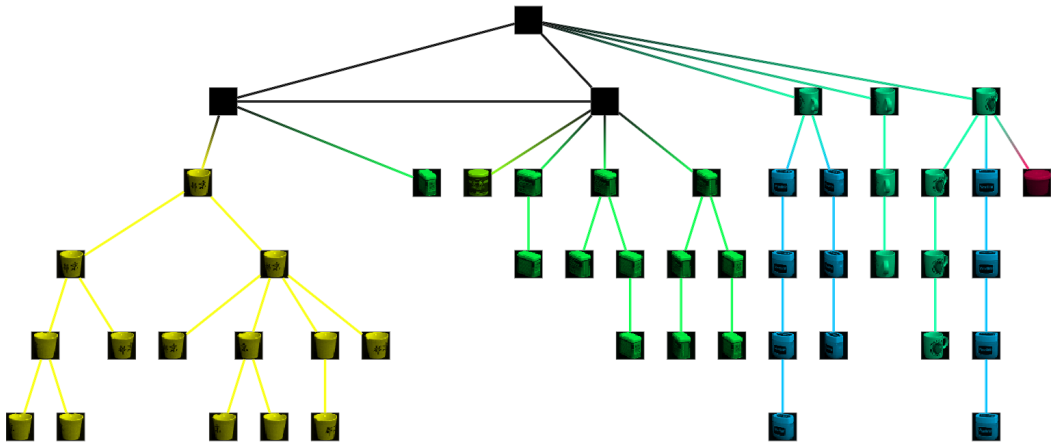


Fig. 7: Zoom sample extracted from Fig. 6a. A 3-tree network shows both hierarchical and topological relations.

REFERENCES

[1] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *SDM*, 2006, pp. 328–339.

[2] C. Isaksson, M. H. Dunham, and M. Hahsler, "SOStream: Self organizing density-based clustering over data stream," in *MLDM*, 2012, pp. 264–278.

[3] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, "StreamKM++: A clustering algorithm for data streams," *ACM Journal of Experimental Algorithmics*, vol. 17, no. 1, 2012.

[4] X. Zhang, C. Furtlehner, and M. Sebag, "Data streaming with affinity propagation," in *ECML/PKDD (2)*, 2008, pp. 628–643.

[5] M. Ghesmoune, H. Azzag, and M. Lebbah, "G-stream: Growing neural gas over data stream," in *Neural Information Processing*, ser. Lecture Notes in Computer Science, C. Loo, K. Yap, K. Wong, A. Teoh, and K. Huang, Eds., vol. 8834. Springer International Publishing, 2014, pp. 207–214.

[6] C. C. Aggarwal, T. J. Watson, R. Ctr, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *In VLDB*, 2003, pp. 81–92.

[7] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: An efficient data clustering method for very large databases," in *SIGMOD Conference*, 1996, pp. 103–114.

[8] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The ClusTree: indexing micro-clusters for anytime stream mining," *Knowledge and information systems*, vol. 29, no. 2, pp. 249–272, 2011.

[9] T. Kohonen, M. R. Schroeder, and T. S. Huang, Eds., *Self-Organizing Maps*, 3rd ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2001.

[10] K. Udommanetanakit, T. Rakthanmanon, and K. Waiyamai, "E-stream: Evolution-based technique for stream clustering," in *ADMA*, 2007, pp. 605–615.

[11] C. Wang, J. Lai, D. Huang, and W. Zheng, "SVStream: A support vector-based algorithm for clustering data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1410–1424, 2013. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TKDE.2011.263

[12] M.-R. Bouguelia, Y. Belaïd, and A. Belaïd, "An adaptive incremental clustering method based on the growing neural gas algorithm," in *ICPRAM*, 2013, pp. 42–49.

[13] P. P. Rodrigues, J. Gama, and J. P. Pedroso, "Odac: Hierarchical clustering of time series data streams." in *SDM*, J. Ghosh, D. Lambert, D. B. Skillicorn, and J. Srivastava, Eds. SIAM, 2006.

[14] B. Fritzke, "Unsupervised clustering with growing cell structures," in *In Proceedings of the International Joint Conference on Neural Networks*. IEEE, 1991, pp. 531–536.

[15] M. Hao, D. A. Keim, U. Dayal, D. Oelke, and C. Tremblay, "Density displays for data stream monitoring." *Comput. Graph. Forum*, vol. 27, no. 3, pp. 895–902, 2008.

[16] Y. Ishikawa and M. Hasegawa, "T-scroll: Visualizing trends in a time-series of documents for interactive user exploration." in *ECDL*, ser. Lecture Notes in Computer Science, L. Kovcs, N. Fuhr, and C. Meghini, Eds., vol. 4675. Springer, 2007, pp. 235–246.

[17] G. Chin, M. Singhal, G. Nakamura, V. Gurumoorthi, and N. Freeman-Cadoret, "Visual analysis of dynamic data streams," *Information Visualization*, vol. 8, no. 3, pp. 212–229, Jun. 2009.

[18] H. Azzag, G. Venturini, A. Oliver, and C. Guinot, "A hierarchical ant based clustering algorithm and its use in three real-world applications," *European Journal of Operational Research*, vol. 179, no. 3, pp. 906–922, June 2007.

[19] N. Doan, H. Azzag, and M. Lebbah, "Growing self-organizing trees for knowledge discovery from data," in *The 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, Australia, June 10-15, 2012*, 2012, pp. 1–8.

[20] N. Doan, H. Azzag, M. Lebbah, and G. Santini, "Self-organizing trees for visualizing protein dataset," in *The 2013 International Joint Conference on Neural Networks, IJCNN 2013, Dallas, TX, USA, August 4-9, 2013*, 2013, pp. 1–8.

[21] J. de Andrade Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. de Carvalho, and J. Gama, "Data stream clustering: A survey," *ACM Comput. Surv.*, vol. 46, no. 1, p. 13, 2013.

[22] D. Auber, "Tulip : A huge graph visualisation framework," in *Graph Drawing Softwares*, ser. Mathematics and Visualization, P. Mutzel and M. Jünger, Eds. Springer-Verlag, 2003, pp. 105–126.

ACKNOWLEDGMENT

This research has been supported by the Embassy of France in Hanoi, the French Foundation FSN, PIA Grant Big data-Investissements d'Avenir. The project is titled "Square Predict" (http://square-predict.net/). We thank anonymous reviewers for their insightful remarks.