G-Stream: Growing Neural Gas over Data Stream

Mohammed Ghesmoune, Hanene Azzag, and Mustapha Lebbah

University of Paris 13, Sorbonne Paris City
LIPN-UMR 7030 - CNRS
99, av. J-B Clément – F-93430 Villetaneuse, France
firstname.secondname@lipn.univ-paris13.fr

Abstract. Streaming data clustering is becoming the most efficient way to cluster a very large data set. In this paper we present a new approach, called G-Stream, for topological clustering of evolving data streams. G-Stream allows one to discover clusters of arbitrary shape without any assumption on the number of clusters and by making one pass over the data. The topological structure is represented by a graph wherein each node represents a set of "close" data points and neighboring nodes are connected by edges. The use of the *reservoir*, to hold, temporarily, the very distant data points from the current prototypes, avoids needless movements of the nearest nodes to data points and therefore, improving the quality of clustering. The performance of the proposed algorithm is evaluated on both synthetic and real-world data sets.

Keywords: Data Stream Clustering, Topological Structure, Growing Neural Gas.

1 Introduction

Clustering is the problem of partitioning a set of observations into clusters such that observations assigned in the same cluster are similar (or close) and the inter-cluster observations are dissimilar (or distant). The other objective of clustering is to quantify the data by replacing a group of observations (cluster) with one representative observation (or prototype). 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 (which requires a single pass through the data) where random access to data is not feasible and storing all arriving data is impractical. Mining data streams can be defined as the process of finding a complex structure in a large data. Clustering data streams requires a process capable of partitioning observations continuously with restrictions of memory and time. In literature, many data stream algorithms have been adapted from clustering algorithms, e.g., the density-based method DbScan [6,8], the partitioning method k-means [1], or the message passing-based method AP [14]. In this paper, we propose G-Stream (Growing Neural Gas over Data Stream), a novel algorithm for discovering clusters of arbitrary shape in an evolving data stream, whose main features and advantages are described below:

C.K. Loo et al. (Eds.): ICONIP 2014, Part I, LNCS 8834, pp. 207–214, 2014.

- The topological structure is represented by a graph wherein each node represents a cluster, which is a set of "close" data points and neighboring nodes (clusters) are connected by edges. The graph size is not fixed but may evolve.
- We use an exponential fading function to reduce the impact of old data whose relevance diminishes over time. For the same reason, links between nodes are also weighted by an exponential function.
- Unlike many other data stream algorithms that start by taking a significant number of data points for initializing the model (these data points can be seen several times), G-Stream starts with only two nodes. Several nodes (clusters) are created in each iteration, the opposite of traditional GNG [7].
- All aspects of G-Stream (including creation, deletion and fading of nodes, edges management, and reservoir management) are performed online.
- A reservoir is used to hold, temporarily, the very distant data points, compared to the current prototypes.

The remainder of this paper is organized as follows: Section 2 is dedicated to related works. Section 3 describes the G-Stream algorithm. Section 4 reports the experimental evaluation on both synthetic and real-world datasets. Section 5 concludes this paper.

2 Related Works

This section discusses previous works on data stream clustering problems, and highlights the most relevant algorithms proposed in literature to deal with this problem. Most of existing algorithms divided the clustering process in two phases: (1) *Online*, the data will be summarized, (2) *Offline*, final clusters will be generated. Both *CluStream* [2] and *DenStream* [6] use a temporal extension of the *Clustering Feature vector* [13] (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 in the *DenStream* algorithm) in order to determine the final clusters. *StreamKM++* [1] maintains a small sketch of the input data using the *merge-and-reduce* technique. The merge step is performed by a means of data structure, named *bucket set*. The reduce step is performed by a significantly different summary data structure, the *coreset tree*. *SOStream* [8] is a density-based clustering algorithm inspired by both the principle of DbScan algorithm and that of the self-organizing maps (SOM) [9]. *E-Stream* [12] 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* [14], an extension of the Affinity Propagation algorithm for data stream, uses a reservoir for saving potential outliers. *AING* [5], an incremental GNG that learns automatically the distance thresholds of nodes based on its neighbors and data points assigned to the concerned

Table 1. Comparison between algorithms (WL: weighted links, 2 phases : online+offline)

Algorithms	based on	topology	WL	phases	remove	merge	split	fade
G-Stream	NGas	✓	✓	online	✓	✗	✗	✓
AING	NGas	✓	✗	online	✗	✓	✗	✗
CluStream	k-means	✗	✗	2 phases	✓	offline	✗	✗
DenStream	DbScan	✗	✗	2 phases	✓	offline	✗	✓
SOStream	DbScan, SOM	✗	✗	online	✓	✓	✗	✓
E-Stream	k-means	✗	✗	2 phases	✓	✓	✓	✓
StreamKM++	k-means	✗	✗	2 phases	✓	✓	✓	✓
StrAP	AP	✗	✗	2 phases	✓	✗	✗	✓

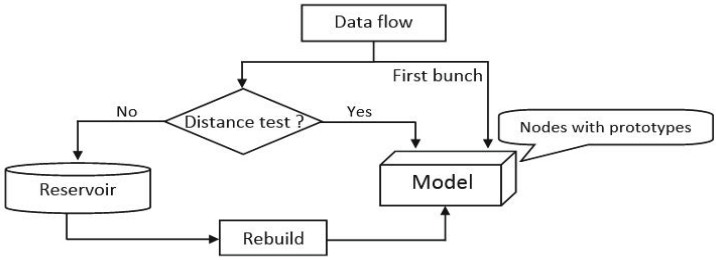
node. It merges nodes when their number reaches a given *upper bound*. Table 1 summarizes the main features offered by each algorithm in terms of: basic clustering algorithm, whether the algorithm identifies a topological structure or not, whether links (if those exists) between clusters (nodes) are weighted, how many phases does it adopt (online and offline), operations for updating clusters (remove, merge, and split cluster), and the *fading* function.

3 Growing Neural Gas over Data Stream

In this section we introduce Growing Neural Gas over data Stream (G-Stream) and highlight some of its novel features. G-Stream is based on Growing Neural Gas (GNG), which is an incremental self-organizing approach that belongs to the family of topological maps such as Self-Organizing Maps (SOM) [9] or Neural Gas (NG) [10]. It is an unsupervised algorithm capable of representing a high dimensional input space in a low dimensional feature map. Typically, it is used for finding topological structures that closely reflect the structure of the input distribution. We assume that the data stream consists of a sequence $\mathcal{DS} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ of n (potentially infinite) data streams arriving in times $T_1, T_2, ..., T_n$, where $\mathbf{x}_i = (x_i^1, x_i^2, ...x_i^d)$ is a vector in $\Re^d$. At each time, G-Stream is represented by a graph $\mathcal{C}$ where each node represents a cluster. Each node $c \in \mathcal{C}$ has a prototype $\mathbf{w}_c = (w_c^1, w_c^2, ...w_c^d)$ (resp. a distance threshold δ_c) representing its position (resp. the distance from the node to the farthest data point assigned to it). Starting with two nodes, and as a new data point is reached, the nearest and the second-nearest nodes are identified, linked by an edge, and the nearest node with and topological neighbors are moved toward the data point. Each node has an accumulated error variable and has weight, which varies over time using Fading function. Using Edge management, one, two or three nodes are inserted into the graph between the nodes with the largest error values. Nodes can also be removed if they are identified as being superfluous. Figure 1 represents a general diagram of the algorithm.

Fading Function: In most data stream scenarios, more recent data can reflect the emergence of new trends or changes in data distribution [3]. There

Fig. 1. Diagram of G-Stream algorithm.



are three models of window commonly studied in the data stream: landmark window, sliding window and 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 node is based on data points associated therewith: $weight(c) = \sum_{i=1}^{m} 2^{-\lambda_1(t-t_{i_0})}$, where m is the number of points assigned to the node c in the current time t. If the weight of a node is less than a parameter value then this node is considered as outdated and then deleted (with its links).

Edge Management: The edge management procedure performs operations related to updating graph edges, as illustrated in steps 13-14 of the algorithm. The way to increase the age of edges is inspired by the fading function in the sense that the creation time of a link is taken into account. Contrary to the *fading* function, the age of the links will be strengthened by the exponential function $2^{\lambda_2(t-t_0)}$, where $\lambda_2 > 0$, defines the rate of growth of the age over time. t denotes the current time and t_0 is the creation time of the edge. The next step is to add a new edge that connects the two closest nodes. The last step is to remove each link exceeding a maximum age, since these links are no longer useful because they were replaced by younger and shorter edges that were created during the graph refinement in steps 15-20.

Reservoir Management: The aim of using the reservoir is to hold, temporarily, the far data points. As mentioned before, each node has a threshold distance. The first bunch of data is assigned to nearest nodes without comparing distances thresholds. The distance threshold of each node is learned by taking the maximum distance of the node to the farthest point that it has been assigned. When the reservoir is full, its data is re-passed to the learning. They are placed in the heap of the data stream, $\mathcal{DS}$, to be dealt with first and the distance thresholds of nodes are updated accordingly.

4 Experimental Evaluations

In this section, we present an experimental evaluation of G-Stream algorithm. We compared our algorithm with the GNG algorithm and two relevant

Algorithm 1. G-Stream

Data: $\mathcal{DS} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$
Result: set of nodes $\mathcal{C} = \{c_1, c_2, ...\}$ and their prototypes $\mathbf{W} = \{\mathbf{w}_{c_1}, \mathbf{w}_{c_2}, ...\}$
Initialize node set $\mathcal{C}$ to contain two nodes, c_1 and c_2: $\mathcal{C} = \{c_1, c_2\}$;
while *there is a data point to proceed* **do**
  Get the next data point in the data stream, $\mathbf{x}_i$;
  Find the nearest node $bmu_1 \in \mathcal{C}$ and the second nearest node $bmu_2 \in \mathcal{C}$;
  if $\|\mathbf{x}_i - \mathbf{w}_{bmu_1}\| > \delta_{bmu_1}$ **then**
    put $\mathbf{x}_i$ in the reservoir;
    if *the reservoir is full* **then** Reservoir management
  else
    Increment the number of points assigned to bmu_1;
    $error(bmu_1) = error(bmu_1) + \|\mathbf{x}_i - \mathbf{w}_{bmu_1}\|^2$;
    Move bmu_1 and its topological neighbors towards $\mathbf{x}_i$:
    $\mathbf{w}_{bmu_1} = \mathbf{w}_{bmu_1} + \alpha_1.\|\mathbf{x}_i - \mathbf{w}_{bmu_1}\|$;
    $\mathbf{w}_c = \mathbf{w}_c + \alpha_2.\|\mathbf{x}_i - \mathbf{w}_c\|$ for all direct neighbors c of node bmu_1;
    Increment the age of all edges emanating from bmu_1 and weight them;
    if bmu_1 *and* bmu_2 *are connected by an edge* **then** set the age of this edge to zero **else** create an edge between bmu_1 and bmu_2, and mark its time stamp Remove edges whose age is greater than age_{max};
    if *the number of points passed is multiple of a parameter* β **then**
      for *i=1 to 3* **do**
        Find node q with the maximum accumulated error;
        Find the neighbor f of q with the largest accumulated error;
        Add the new node, r, half-way between nodes q and f;
        Insert edges connecting the new node r with nodes q and f, and remove the original edge between q and f;
    Application of *fading*, delete outdated and isolated nodes;
    Finally, decrease the error of all units;

data stream algorithms. Our experiments were performed on MATLAB platform using real-world and synthetic datasets. Table 2 overviews all the datasets used. The real-world databases were taken from the UCI repository [4]. DS1 is generated by `http://impca.curtin.edu.au/local/software/synthetic-data-sets.tar.bz2`. Uniform is generated with matlab code. The letter4 dataset is generated by a Java code `https://github.com/feldob/Token-Cluster-Generator`. The algorithms are evaluated using three performance measures: Rand, Normalized Mutual Information and Accuracy (Purity) with the aim of maximizing each measure [11]. As explained in section 3, GNG and G-Stream algorithms start with two nodes. We used an online version of GNG but without the parameters that we added and this, precisely, to show the interest and contribution of these parameters in G-Stream. Therefore, we did experiments by initializing two nodes randomly among the first 20 points and we repeated this 10 times. We used the same initialization for both algorithms (G-Stream and GNG) and the average

Table 2. Overview of all datasets

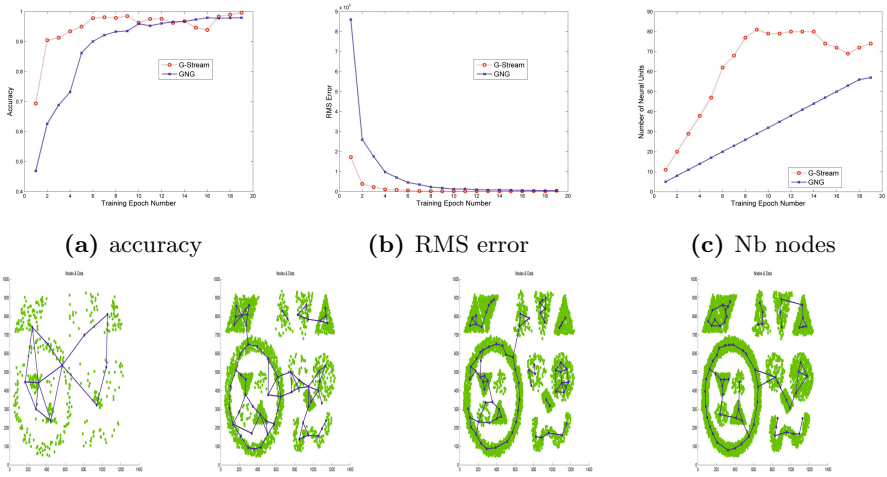
Datasets	size	#features	#classes
DS1	9153	2	14
Uniform	24000	2	4
letter4	9344	2	7
Shuttle	43500	9	7
L-recognition	20000	16	26
KddCup1	49402	34	18

value with its standard deviation is reported in Table 3. These results show that G-Stream algorithm outperforms the GNG algorithm on almost all the datasets. For comparison purpose, we used CluStream with **stream** R package `http://cran.r-project.org/web/packages/stream/index.html`. Comparison is also performed with StreamKM++ [1]. Results are reported in Table 3. Again, with reference to Table 3, it is noticeable that G-Stream's Accuracies (Acc) are higher for all datasets as compared to GNG, StreamKM++, and CluStream. The NMI values are also higher than the other algorithms except for CluStream in DS1, and for the two data stream algorithms in Uniform. The Rand values are also higher than the other algorithms except in Uniform and shuttle. We remind that G-Stream proceeds in one single phase whereas CluStream and StreamKM++ proceed in two phases (online and offline phase).

Table 3. Comparing G-Stream with different algorithms

Datasets		GNG	**G-Stream**	StreamKM++	CluStream
DS1	Acc	0.511±0.251	**0.993±0.006**	0.675±0.018	0.701±0.028
	NMI	0.491±0.132	0.712 ±0.004	0.702±0.021	0.723±0.022
	Rand	0.621±0.122	**0.846±0.001**	0.844±0.004	0.845±0.007
Uniform	Acc	1 ± 0	**1 ± 0**	0.998±0.004	0.995±0.012
	NMI	0.492±0	0.568±0.003	0.777±0.007	0.787±0.015
	Rand	0.754±0	0.765±0	0.855±0.003	0.868±0.011
letter4	Acc	0.577±0.201	**0.991±0**	0.687±0.026	0.934±0.026
	NMI	0.529±0.074	**0.607±0**	0.553±0.022	0.264±0.034
	Rand	0.686±0.084	**0.812±0**	0.794±0.014	0.341±0.004
Shuttle	Acc	0.963±0.002	**0.973±0.004**	0.822±0.003	0.899±0.017
	NMI	0.355±0	**0.362±0.007**	0.258±0.015	0.340±0.035
	Rand	0.378±0.001	0.376±0.001	0.753±0.039	0.559±0.059
L-recognition	Acc	0.077±0.068	**0.408±0.019**	0.161±0.009	0.181±0.009
	NMI	0.046±0.096	**0.437±0.015**	0.239±0.015	0.267±0.011
	Rand	0.541±0.139	**0.956±0.001**	0.861±0.006	0.851±0.007
KddCup1	Acc	0.929±0.085	**0.998±0.001**	0.768±0	0.998±0
	NMI	0.655±0.319	**0.602±0.032**	0.012±0.003	0.022±0.002
	Rand	0.824±0.206	**0.655±0.045**	0.623±0.003	0.369±0.083

Figure 2a (resp. Figure 2b) compares G-Stream (red line with circle) with GNG (blue line with cross) with respect to accuracy (resp. RMS error, number of nodes). On almost all times, the accuracy value (resp. RMS error) of G-Stream is higher (resp. is less) than the one of GNG. Figure 2c compares the two algorithms in terms of number of nodes creating the graph. Despite this we create several nodes at each iteration (against a single node for GNG), the number of nodes created by G-Stream becomes steady (against a continuously increase for GNG) due to the application of the fading function. The same result can be shown on the rest of the datasets. The second row of Figure 2 shows the



(a) accuracy **(b)** RMS error **(c)** Nb nodes

Fig. 2. DS1 Experimentation. Accuracy, RMS error, and number of nodes for G-Stream on DS1. The second row shows visual result of G-Stream on DS1 (dataset and topological result).

evolution of the creation of nodes by applying G-Stream on DS1 (green points represent data points of the data stream and blue ones are nodes of the graph with edges in blue lines). It illustrates that G-Stream manages to recognize the structures of the data stream and can separate these structures with the best visualization. Due to space limitations, we omitted the visual results about the other datasets since they have the same interpretation as that we have shown.

5 Conclusion

In this paper, we have proposed G-Stream, an efficient method for topological clustering an evolving data stream in an online manner. In G-Stream, nodes are weighted by a fading function as well as edges by an exponential function. Starting with two nodes, G-Stream confronts the arriving data points to the current prototypes, storing the very distant ones in a reservoir, learns the threshold

distances automatically, and many nodes are created in each iteration. Experimental evaluation over a number of real and synthetic data sets demonstrates the effectiveness and efficiency of G-Stream in discovering clusters of arbitrary shape. Our experiments show that G-Stream outperformed the GNG algorithm in terms of visual results and criteria as accuracy, rand and NMI. G-Stream is also compared, in terms of clustering quality, to two relevant data stream algorithms, results are promising. We plan in future to make adaptive windows, make our algorithm as autonomous as possible and develop it in Spark or Storm for testing on large datasets with other data stream algorithms.

Acknowledgments. This work has been supported by the French foundation PIA Grant Big data "Square Predict".

References

1. Ackermann, M.R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., Sohler, C.: Streamkm++: A clustering algorithm for data streams. ACM Journal of Experimental Algorithmics 17(1) (2012)
2. Aggarwal, C.C., Watson, T.J., Ctr, R., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: VLDB, pp. 81–92 (2003)
3. de Andrade Silva, J., Faria, E.R., Barros, R.C., Hruschka, E.R., de Carvalho, A.C.P.L.F., Gama, J.: Data stream clustering: A survey. ACM Comput. Surv. 46(1), 13 (2013)
4. Bache, K., Lichman, M.: UCI machine learning repository (2013), `http://archive.ics.uci.edu/ml`
5. Bouguelia, M.R., Belaïd, Y., Belaïd, A.: An adaptive incremental clustering method based on the growing neural gas algorithm. In: ICPRAM, pp. 42–49 (2013)
6. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: SDM, pp. 328–339 (2006)
7. Fritzke, B.: A growing neural gas network learns topologies. In: NIPS, pp. 625–632 (1994)
8. Isaksson, C., Dunham, M.H., Hahsler, M.: SOStream: Self organizing density-based clustering over data stream. In: Perner, P. (ed.) MLDM 2012. LNCS (LNAI), vol. 7376, pp. 264–278. Springer, Heidelberg (2012)
9. Kohonen, T., Schroeder, M.R., Huang, T.S. (eds.): Self-Organizing Maps, 3rd edn. Springer-Verlag New York, Inc., Secaucus (2001)
10. Martinetz, T., Schulten, K.: A "Neural-Gas" Network Learns Topologies. In: Artificial Neural Networks I, pp. 397–402 (1991)
11. Strehl, A., Ghosh, J.: Cluster ensembles — a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research 3, 583–617 (2002)
12. Udommanetanakit, K., Rakthanmanon, T., Waiyamai, K.: E-stream: Evolution-based technique for stream clustering. In: Alhajj, R., Gao, H., Li, X., Li, J., Zaïane, O.R. (eds.) ADMA 2007. LNCS(LNAI), vol. 4632, pp. 605–615. Springer, Heidelberg (2007)
13. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: An efficient data clustering method for very large databases. In: SIGMOD Conference, pp. 103–114 (1996)
14. Zhang, X., Furtlehner, C., Sebag, M.: Data streaming with affinity propagation. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 628–643. Springer, Heidelberg (2008)