Change detection in data streams through unsupervised learning

Guénaël Cabanes and Younès Bennani
LIPN-CNRS, UMR 7030
99 Avenue J-B. Clément, 93430 Villetaneuse, France
Email: cabanes@lipn.univ-paris13.fr

***Abstract*—In many cases, databases are in constant evolution, new data is arriving continuously. Data streams pose several unique problems that make obsolete the applications of standard data analysis methods. Indeed, these databases are constantly on-line, growing with the arrival of new data. In addition, the probability distribution associated with the data may change over time. We propose in this paper a method of synthetic representation of the data structure for efficient storage of information, and a measure of dissimilarity between these representations for the detection of change in the stream structure.**

***Index Terms*—Concept drift, usupervised lerning, data streams.**

I. Introduction

In many cases, databases are constantly evolving, they are characterized by a changing structure over time, new data arriving continuously. Sometimes, the evolution and the mass of data is so important that it is impossible to store them in a database. Only an analysis "on the fly" is possible. These processes are called "data streams analysis" and are the subject of numerous studies in recent years due to the large number of potential applications in many fields [1], [2], [3], [4]. The study of data streams is a difficult problem: the computing and storage cost are high and the size of involved datasets is big. In the field of data mining, the main challenges for the study of data streams are the ability to compute a condensed description of the stream properties [5], [6], [7], but also the detection of change in the stream structure [8], [9], [10].

Data streams pose several unique problems that make obsolete the applications of standard data analysis. Indeed, these databases are constantly online, growing with the arrival of new data. Thus, efficient algorithms must be able to work with a constant memory footprint, despite the evolution of the stream, as the entire database cannot be retained in memory. This may implies forgetting some information over time. Another difficulty is known as the "concept drift" problem: the probability distribution associated with the data may change over time. Any learning algorithm adapted to streams should be able to detect and manage these situations. In the context of supervised learning (each data is associated with a given class, that the algorithm must learn to predict), several solutions have been proposed for the classification of data streams in the presence of concept drift. These solutions are generally based on adaptive maintenance of a discriminatory structure, for example using a set of binary rules [11], decision trees [12] or ensembles of classifiers [13], [14].

This paper deals with an unsupervised framework (class labels are unknown), which requires adaptations to the presence of concept drift for the analysis of data streams. We propose a method of synthetic representations of the data structure and a heuristic measure of dissimilarity between these models to detect temporal variations in the structure of the stream (concept drifts). The advantage of this method is the comparison of structures by means of models that describe them, allowing comparisons at any time scale without overloading the memory. Thus, it is possible to compare the structure of the stream in two potentially very distant time periods, since the models describing these periods can be stored in memory at very low cost.

Section 2 provides a general description of the unsupervised method for stream analysis and detection of change in the stream structure. Section 3 presents the algorithm computing a synthetic representations of the structural information of the data. The construction of a model of the data distribution is proposed in Section 4, and a dissimilarity measure for the detection of change in the stream structure is presented in Section 5. Finally, Section 6 describes a method to compress the information and save memory cost.

II. General description of the proposed method

The unsupervised method of stream analysis and detection of change in the stream structure proceeds in four steps:

- Construction of a synthetic representation at regular intervals over a data reservoir which empties itself as new data is stored. This synthetic representation is based on the learning of a SOM (Self-Organizing Map [15]) and allows automatic data clustering. During the learning, each SOM prototype is extended with novel information extracted from the data. These information will be used in the following step to infer the distribution probability. More specifically, the attributes added to each prototype are the following.
  - *Density modes*. It is a measure of the data density surrounding the prototype (local density). The local density is a measure of the amount of data present in an area of the input space. We use a Gaussian kernel estimator [16] for this task.
  - *Local variability*. It is a measure of the data variability that is represented by the prototype. It can

be defined as the average distance between the prototypes and the represented data.
  - *The neighborhood*. This is a prototype's neighborhood measure. The neighborhood value of two prototypes is the number of data that are well represented by each one.
- Estimation of the data distribution from each synthetic representation. This estimation is done offline and does not require data storage. It is modeled as a density function from a mixture of Gaussian spherical kernels.
- Distributions comparison for the detection of concept drift. We propose the use of a dissimilarity measure that can compare the two density functions estimated in the previous step.
- Compression of recorded information over the stream for a fixed memory size storage taking into account a continuous arrival of new data.

III. Construction of a synthetic representation

In this step, some general information are extracted from the data and stored in the prototypes during the learning of the SOM. In our algorithm, the prototypes of the SOM will be "enriched" by the addition of new numerical values extracted from the data structure.

The enrichment algorithm proceeds in three phases :

Input :

- The data $X = \{x^{(k)}\}_{k=1}^{N}$.

Output :

- The density D_i and the local variability s_i associated to each prototype w_i.
- The neighborhood values $v_{i,j}$ associated with each pair of prototype w_i and w_j.

1) **Initialization**:
   - Initialize the SOM parameters
   - $\forall i, j$ initialize to zero the local densities (D_i), the neighborhood values ($v_{i,j}$), the local variability (s_i) and the number of data represented by w_i (N_i).
2) **Choose randomly a data** $x_k \in X$:
   - Compute $d(w, x_k)$, the euclidean distance between the data x_k and each prototype w_i.
   - Find the two closest prototypes (BMUs: Best Match Units) w_{u^*} and $w_{u^{**}}$:

$$u^* = \arg\min_i(d(w_i, x_k))$$

and

$$u^{**} = \arg\min_{i \neq u^*}(d(w_i, x_k))$$

.

3) **Update structural values**:
   - Variability:

$$\begin{aligned} s_{u^*}(t) &= s_{u^*}(t\text{-}1) \\ &\quad -\varepsilon(t)r(t)\left(s_{u^*}(t\text{-}1) - d(w_{u^*}, x_k)\right) \end{aligned}$$

   - Density:

$$\begin{aligned} \forall j, D_j(t) &= D_j(t\text{-}1) \\ &\quad -\varepsilon(t)r(t)\left(D_j(t\text{-}1) - G(t)\right) \end{aligned}$$

with

$$G(t) = e^{-\frac{\|x^{(k)} - w_j(t)\|^2}{2\sigma^2}}$$

   - Neighborhood:

$$\begin{aligned} \nu_{u^*u^{**}}(t) &= \nu_{u^*u^{**}}(t\text{-}1) \\ &\quad -\varepsilon(t)r(t)\left(\nu_{u^*u^{**}}(t\text{-}1) - 1\right) \\ \nu_{u^*i}(t) &= \nu_{u^*i}(t\text{-}1) \\ &\quad -\varepsilon(t)r(t)\left(\nu_{u^*i}(t\text{-}1)\right) \\ &\quad \forall i \text{ neighbor of } u^* \end{aligned}$$

With $\varepsilon(t)$ the learning rate and $r(t) = \frac{1}{1+e^{\left(-\frac{t}{t_{max}}\right)}}$.

4) **Update the SOM prototypes** w_i as defined in [15].
5) **Repeat** T **times step** 2 **to** 4**, until** $t = t_{max}$.

At the end of this process, each prototype is associated with a density and a variability value, and each pair of prototypes is associated with a neighborhood value. The substantial information about the distribution of the data is captured by these values. Then, it is no longer necessary to keep data in memory. This information can be used directly to perform clustering of the SOM and highlight the structure of the stream for the period recorded in the reservoir (see [17]). The complexity of the whole process is linear with the number of data, enabling rapid analysis of the stream during its evolution.

IV. Estimation of the data distribution

This step involves estimating the underlying distribution of data using a two-levels method. The idea here is to estimate the data distribution from a topological model of the data. We therefore propose to estimate a density function that associates a density value to each point in the input space. We know the value of this function at the location of each prototype (it is D_i for a prototype w_i). We must infer from this an approximation of the distribution function.

The hypothesis here is that this function may be properly approximated in the form of a mixture of spherical Gaussian kernels ($\{K_i\}_{i=1}^{M}$). Each kernel K_i is a Gaussian function centered on a prototype w_i and M is the number of prototype. The density function can therefore be written as:

$$f(x) = \sum_{i=1}^{M} \alpha_i K_i(x)$$

with

$$K_i(x) = \frac{1}{\sqrt{2\pi}.h_i} e^{-\frac{|w_i - x|^2}{2{h_i}^2}} \text{ and } \sum \alpha_i = 1$$

The most popular method to fit mixture models (i.e. to find h_i and α_i) is the expectation-maximization (EM) algorithm

[18]. However, this algorithm needs to work in the data input space. As here we work on enriched SOM instead of dataset, we cannot use EM algorithm (see [19]).

Thus, we propose the heuristic to choose h_i:

$$h_i = \frac{\sum_j \frac{v_{i,j}}{N_i+N_j}(s_i N_i + d_{i,j} N_j)}{\sum_j v_{i,j}} \quad (1)$$

$d_{i,j}$ is the distance between w_i and w_j. The idea is that h_i is the standard deviation of data represented by K_i. These data are also represented by w_i and their neighbors. Then h_i depends on the variability s_i computed for w_i and the distance $d_{i,j}$ between w_i and its neighbors, weighted by the number of data represented by each prototype and the connectivity value between w_i and its neighborhood.

Now, since the density D for each prototype w is known ($f(w_i) = D_i$), a gradient descent method can be used to determine the weights α_i. These weights are solution of the following linear system of equations :

$$D = \sum_{i=1}^{M} \alpha_i K_i(w)$$

with

$$D = [D_j]_{j=1}^{M} \text{ et } w = [w_j]_{j=1}^{M}$$

However, there are an infinity of solutions to this equation, which makes impossible any resolution based on matrix inversion. In addition, the solution obtained by calculating the pseudo-inverse [20] is often unsatisfactory, particularly because it can find values of α that no longer guarantee the constraint $\forall x, f(x) > 0$. We therefore use to solve this equation a very simple gradient descent algorithm. The α_i are initialized by the values of D_i, and have their value reduced gradually (with a minimum value of 0) to best satisfy $D = \sum_{i=1}^{M} \alpha_i K_i(w)$. Thus, the values of α are proportional to the average values of D_i, which satisfies the assumption that each D_i density is generated primarily by the prototype w_i. For this, we optimize the following criterion:

$$\alpha = \arg\min_{\alpha} \frac{1}{M} \sum_{i=1}^{M} \left[\sum_{j=1}^{M} (\alpha_j K_j(w_i)) - D_i \right]^2$$

Thus, we obtain a density function which is a model of the data represented by the SOM. This kind of method for estimating the data distribution has been used successfully in [19] in a different context. Figures 1 and 2 show some examples of estimated density.

V. Distributions comparison for the detection of concept drift

We propose in this step a heuristic measure of dissimilarity between two distributions represented by density functions calculated in the previous step. The objective of this measure is to compare the distributions of two sets of data described in the same space, so as to detect if their distributions are identical, similar or quite different (see Fig. 3). This comparison allows the detection of changes in the structure of a data stream (concept drift).

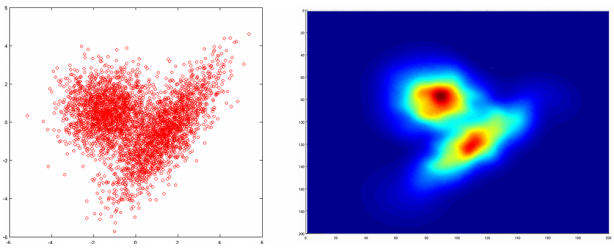

Fig. 1. "Engytime" dataset (left) and the estimated density function (right).

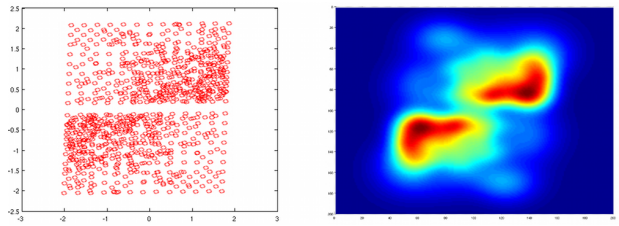

Fig. 2. "Wingnut" dataset (left) and the estimated density function (right).

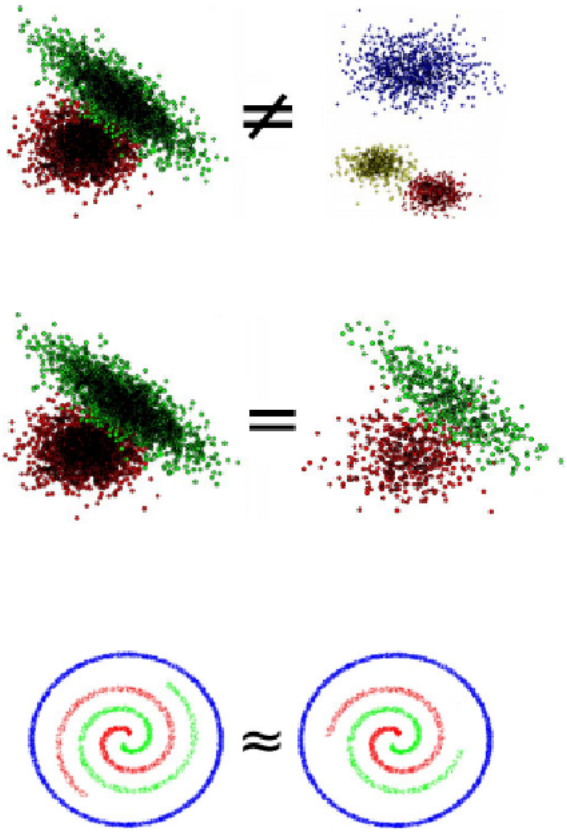

Fig. 3. Example of comparisons of data distributions.

A. Dissimilarity measure

It is possible to define a dissimilarity measure between two data sets A and B, each represented by a SOM and a density function:

$$SOM_A = \left[\{w_i^A\}_{i=1}^{M^A}, f^A\right]$$

and

$$SOM_B = \left[\{w_i^B\}_{i=1}^{M^B}, f^B\right]$$

With M^A and M^B the number of prototypes of models SOM_A and SOM_B, f^A and f^B are the density function of A and B computed during the previous step.

The dissimilarity between A and B is :

$$\begin{aligned} CBd(A,B) &= \frac{\sum_{i=1}^{M^A} f^A(w_i^A) log\left(\frac{f^A(w_i^A)}{f^B(w_i^A)}\right)}{M^A} \\ &\quad + \frac{\sum_{j=1}^{M^B} f^B(w_j^B) log\left(\frac{f^B(w_j^B)}{f^A(w_j^B)}\right)}{M^B} \\ &= CBd_A + CBd_B \end{aligned}$$

The idea is to compare the density functions f^A and f^B for each prototype w of A and B. If distributions are identical, these values must be very close. This measure is an adaptation of the weighted Monte Carlo approximation of the symmetrical Kullback-Leibler measure (see [21]), using the prototypes of a SOM as a sample of the database. The idea is to compare for each prototype i the density D_i estimated from the data and the theoretical density $Fd(w_i)$ at this prototype location, estimated by the density function of the other model. If the models are identical, the two density measurements should be very close.

In addition, the index satisfies the properties of a dissimilarity measure.

- Positivity : $CBd(A,B) > 0$
- Symmetry : $CBd(A,B) = CBd(B,A)$
- Separation : $CBd(A,A) = 0$.

In order to demonstrate the performance of the proposed dissimilarity measure, we used five artificial datasets generators.

Generators “Ring 1”, “Ring 2”, “Ring 3”, “Spiral 1” and “Spiral 2” generate five types of non-convex two-dimensional data sets, with different density and variance. “Ring 1” is a ring of radius 1 (high density), “Ring 2” a ring of radius 3 (low density) and “Ring 3” a ring of radius 5 (medium density). “Spiral 1” and “Spiral 2” are two parallel spirals. The density in the spiral decreases with radius. Data from different distributions can be generated randomly to control (Figure V-A).

When the number of prototypes is sufficient, the different distributions are well differentiated. As can be seen in Figure 5, the distances between models corresponding to the same distribution are much smaller than the distances between models of different distributions. To see the similarities between models,

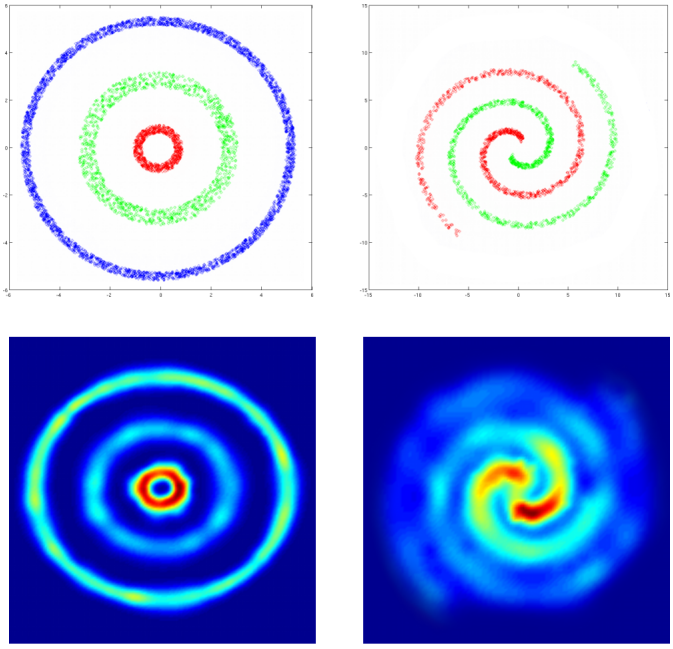

Fig. 4. Visualizations of data “Rings” 1 to 3 and ”Spirals“ 1 and 2 (top) and their density function estimated by our algorithm (bottom).

we used a Sammon projection [22] in two dimensions, which respects the similarities between elements in the projection space.

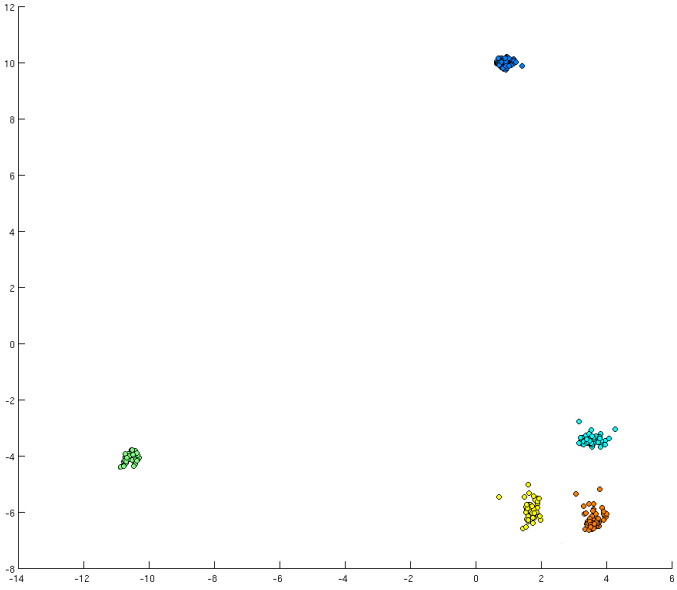

Fig. 5. Visualizations of the similarities between models (one point = one model). Blue: models of Ring 5, Green: Ring 1, Turquoise: Ring 3, Yellow and Orange: Spirals 1 and 2.

To test the ability of the method to detect a concept drift, we presented to the system random “Spiral 1” data until time 5 (each time step represents one thousand data), then we presented “Ring 5” data until time 20, then “Spiral 2” until time 25 and finally “Spiral 1” until time 30. The system learns

for each time period an enriched SOM and compares it to the one computed in the previous period. All our experiments are based on the use of the "SOMToolbox" [23] by taking the default settings for the learning of the SOM.

Figure 6 represents the difference between two models of two consecutive sets over time. Changes in the structure of the stream are perfectly detected by the system. Indeed, when the stream does not change, the corresponding models are very close and the dissimilarity measure provides a very low value. On the contrary, if the structure of the stream varies, the corresponding models are much less similar, and the dissimilarity measure is significantly higher.

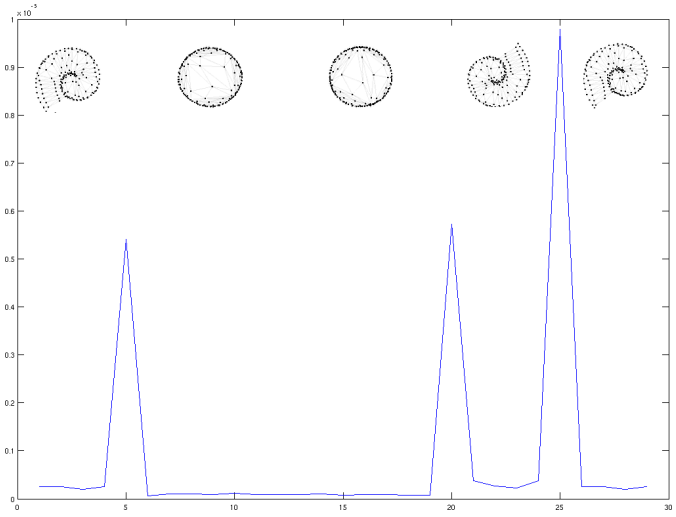

Fig. 6. Dissimilarity between the two models of two consecutive sets over time. Some models are shown to illustrate the temporal variations. These variations in the structure of the stream (time 5, 20 and 25) are perfectly detected.

B. Extension to cluster comparisons

The extension of the proposed method to the clusters comparison is natural, since the SOM-based models are well suited for data clustering. The idea is to detect, after clustering two data sets, if some clusters are common, if some are similar with some variations (concept drift), and if some clusters are unique to each set:

1) Define density function for each cluster C_k for datasets A and B:
$$f_k(x) = \sum_{i \in C_k} \alpha_i K_i(x))$$
2) For each cluster k belonging to one of these datasets, find the most similar cluster k' in the other set :
$$k' = \arg\min_i CBd(k,i)$$
3) Weight the distance between k and k' using the mean distance between k and the others clusters in the same dataset.
$$CBd_w(k,k') = \frac{CBd(k,k')}{\frac{1}{|C_k|}\sum_{i \in C_k, i \neq k} CBd(k,i)}$$
Note that CBd_w is not symmetrical.
4) It is then possible to compare datasets:
   - If $CBd_w(k,k') < 1$, there is a concept drift between k and k'.
   - If $CBd_w(k,k') \approx 0$, k and k' are the same clusters.
   - If $CBd_w(k,k') \geq 1$, k and k' are different clusters.

In this way, it is easy to detect common clusters to both datasets and particular clusters to each one.

It is also possible to detect mergers or fission of clusters from a dataset to another:

1) Decompose the index: $CBd(k,k') = CBd_k + CBd_{k'}$ like in Section V-A.
2) Define a merger-fission index between k and k':
$$MF(k,k') = \frac{CBd_k}{CBd_{k'}}$$
3) Analyze MF :
   - If $MF(k,k') \approx 1$, the two clusters are well separated or very similar.
   - If $MF(k,k') \ll 1$, k is a part of cluster k'.
   - If $MF(k,k') \gg 1$, k' is a part of cluster k.

In this way, if two clusters k_1 and k_2 from one dataset A both have as closest corresponding cluster k' from dataset B and if $MF(k',k_1) \geq 1$ and $MF(k',k_2) \geq 1$, then k' is a merger of k_1 and k_2, and k_1 and k_2 are a fission of k'.

This method therefore allows a very detailed analysis of variations in structures between two data sets. In particular, this kind of analysis is very useful for understanding changes in the structure of a data stream, such as appearance and disappearance of clusters, and phenomena of mergers, fissions or concept drifts.

VI. Compression and storage of the stream structure

One difficulty in the analysis of a data stream is that the stream is potentially infinite. We cannot afford to store indefinitely SOM maps representing different times periods, since the storage capacities are limited. We therefore wish to propose a method of merging enriched SOM, which will compress the stored information.

The idea is to merge two or more SOM representing successive instants when the data structure represented by these two maps is sufficiently similar. We propose, if the available computing power permits, to build and update a matrix of similarity between the density functions representing the structure of the stream for consecutive periods. One just have to compare each new stored SOM with the most recent SOM already stored. Then, if the space is insufficient, it is possible to merge the two most similar adjacent enriched SOM. Thus, at each compression of stored information, the maximum information on variations of the stream is retained.

The merging of SOM can be done by generating data from the density functions and running enriched SOM algorithm on these data.

Let have N enriched SOM and their density function:

$$SOM^1 = \{N_i^1, w_i^1, \alpha_i^1, h_i^1\}_{i=1}^{M^1}$$

$$\dots$$

$$SOM^N = \{N_i^N, w_i^N, \alpha_i^N, h_i^N\}_{i=1}^{M^N}$$

The data generation algorithm is as follows:

1) Select randomly a SOM. Each SOM A, made of M^A neurones, have a probability to be chosen:

$$P(A) = \frac{\sum_{i=1}^{M^A} N_i^A}{\sum_{K=1}^{N} \sum_{i=1}^{M^K} N_i^K}$$

In other words, the more a SOM represents a large amount of data, the greater its chance of being selected.

2) Randomly select a neuron i of the selected SOM according to the parameter α, which represents the contribution of the neuron for the density function:

$$P(i) = \frac{\alpha_i}{\sum_{j=1}^{M} \alpha_j}$$

3) Generate random data according to a spherical Gaussian distribution centered on w_i with a standard deviation of h_i.

It is then enough to apply an enriched SOM algorithm on data generated and to estimate a density function for a condensed representation of the structure of the N original SOM.

VII. Conclusion

In this paper, we propose an unsupervised method for analyzing data streams. This method allows the analysis, the compression and the storage of information about the stream structure and its variations over time. It is fast enough to be applicable to large data streams: the complexity of synthetic representation is linear in the size of the reservoir and the subsequent steps do not require data to be kept in memory, as they are based exclusively on the synthetic representation. We have shown through some examples that the method is able to detect changes in the structure of the stream and to detect concept drifts.

These preliminary results must now be confirmed by a scaling, through real-world applications on large data stream with an increasing number of dimensions. It is important to ensure that this type of method is able to detect more progressive concept drifts than those tested. In addition, we are currently working on an adaptive version of the enriched SOM, which will be able to follow the stream in real time and adapt its performance over time, with an incremental fusion of SOM during the process.

Acknowledgement

This work was supported in part by the E-Fraud Box project (ANR-09-SECU-03) financed by the ANR (Agence Nationale de la Recherche).

References

[1] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," in *Proceedings of IEEE International Conference on Data Engineering*, 2002.

[2] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Very Large Data Base*, 2003, pp. 81–92.

[3] S. Guha and B. Harb, "Approximation algorithms for wavelet transform coding of data streams," *IEEE Transactions on Information Theory*, vol. 54, no. 2, pp. 811–830, 2008.

[4] A. Balzanella, Y. Lechevallier, and R. Verde, "A new approach for clustering multiple streams of data," in *Classification and Data Analysis*, S. Ingrassia and R. Rocci, Eds., 2009, pp. 417–420.

[5] G. S. Manku, S. Rajagopalan, and B. G. Lindsay, "Random sampling techniques for space efficient online computation of order statistics of large datasets," in *Special Interest Group on Management of Data Conference*, 1999, pp. 251–262.

[6] J. Gehrke, F. Korn, and D. Srivastava, "On computing correlated aggregates over continual data streams," in *Special Interest Group on Management of Data Conference*, 2001, pp. 13–24.

[7] G. S. Manku and R. Motwani, "Approximate frequency counts over data streams," in *Very Large Data Base*, 2002, pp. 346–357.

[8] R. Schweller, A. Gupta, E. Parsons, and Y. Chen, "Reversible sketches for efficient and accurate change detection over network data streams," in *IMC'04: proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, New York, NY, USA, 2004, pp. 207–212.

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

[10] C. Aggarwal and P. Yu, "A Survey of Synopsis Construction Methods in Data Streams." in *Data Streams: Models and Algorithms*, C. Aggarwal, Ed. Springer, 2007, pp. 169–207.

[11] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996.

[12] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, CA: ACM Press, 2001, pp. 97–106.

[13] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification." ACM Press, 2001, pp. 377–382.

[14] J. Z. Kolter and M. A. Maloof, "Using additive expert ensembles to cope with concept drift," in *ICML*, 2005, pp. 449–456.

[15] T. Kohonen, *Self-Organizing Maps*. Berlin: Springer-Verlag, 2001.

[16] B. Silverman, "Using kernel density estimates to investigate multimodality," *Journal of the Royal Statistical Society, Series B*, vol. 43, pp. 97–99, 1981.

[17] G. Cabanes and Y. Bennani, "A local density-based simultaneous two-level algorithm for topographic clustering," in *Proceeding of the International Joint Conference on Neural Networks (IJCNN)*, 2008, pp. 1176–1182.

[18] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, pp. 1–38, 1977.

[19] G. Cabanes and Y. Bennani, "Unsupervised topographic learning for spatiotemporal data-mining," *Advances in Artificial Intelligence*, vol. 2010, Article ID 832542, 12 pages, 2010.

[20] A. Ben-Israel and T. N. E. Greville, *Generalized Inverse: Theory and Applications.* New York: Springer Verlag, 2003.

[21] J. R. Hershey and P. A. Olsen, "Approximating the Kullback Leibler divergence between Gaussian mixture models," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 4, 2007, pp. 317–320.

[22] J. Sammon Jr., "A nonlinear mapping for data structure analysis," *IEEE Transactions on Computer*, vol. 18, no. 5, pp. 401–409, May 1969.

[23] J. Vesanto, "Neural network tool for data mining: SOM Toolbox," 2000.