A new unsupervised neural approach to stationary and non-stationary data

A new unsupervised neural approach to stationary and non-stationary data

Vincenzo Randazzo[a], Giansalvo Cirrincione[bc], Eros Pasero[a]

[a] DET, Politecnico di Torino, Turin, Italy
{vincenzo.randazzo, eros.pasero}@polito.it

[b] Université de Picardie Jules Verne, Amiens, France
[c] University of South Pacific, Suva, Fiji
exin@u-picardie.fr

Abstract. Dealing with time-varying high dimensional data is a big problem for real time pattern recognition. Non-stationary topological representation can be addressed in two ways, according to the application: life-long modeling or by forgetting the past. The G-EXIN neural network addresses this problem by using life-long learning. It uses an anisotropic convex polytope, which, models the shape of the neuron neighborhood, and employs a novel kind of edge, called bridge, which carries information on the extent of the distribution time change. In order to take into account the high dimensionality of data, a novel neural network, named GCCA, which embeds G-EXIN as the basic quantization tool, allows a real-time non-linear dimensionality reduction based on the Curvilinear Component Analysis. If, instead, a hierarchical tree is requested for the interpretation of data clustering, the new network GH-EXIN can be used. It uses G-EXIN for the clustering of each tree node dataset.

This chapter illustrates the basic ideas of this family of neural networks and shows their performance by means of synthetic and real experiments.

Keywords: bridge, convex polytope, curvilinear component analysis, dimensionality reduction, fault diagnosis, hierarchical clustering, non-stationary data, projection, real-time pattern recognition, seed, unsupervised neural network, vector quantization.

Open problems in cluster analysis and vector quantization

The topological representation of data is an important challenge for unsupervised neural networks. They build a covering of the data manifold in form of a directed acyclic graph (DAG), in order to fill the input space. However, above all for high

dimensional data, the covering is prone to the problem of the curse of dimensionality, which requires, in general, a large number of neural units. The nodes of the graph are given by the weight vectors of the neurons and the edges, if present, by their connections. The weight estimation, in several cases, implies the minimization of an error function based on some error (e.g. vector quantization, VQ). In other cases, only the iterative technique is given. In general, VQ is performed by using competitive learning (neural units compete for representing the input data): it can be either hard (HCL, e.g. LBG [1] and k-means [2]) or soft (SCL, e.g. neural gas [3] and Self Organizing Maps, SOM, [4]). In HCL only the winning neuron (the closest to the input in terms of weight distance) changes its weight vector. For this reason, it is also known winner-take-all. Instead, in SCL, a.k.a. winner-take-most, both the winner and its neighbors adapt their weights. This approach needs a definition of neighborhood, which requires a network topology, as a graph, whose edges are in general found by means of the Competitive Hebbian Rule (CHR [5]), as in the Topology Representing Network [6], or by back-projecting a fixed grid as in SOM.

Incremental or growing neural networks do not require a prior choice of the architecture, which is, instead, determined by the data (data-driven). All these techniques need a *novelty test* in order to decide when a new neuron has to be created. All tests demand, in general, a model representing the portion of input space explained by each unit. This model is, in general, a hypersphere, because it is as simplest as possible: only a scalar hyperparameter, its radius, is needed. All existing algorithms determine, in a way or another, this threshold. It can be set a user-dependent global parameter (IGNG [7]), or it can be automatically and locally estimated. The single-layer Enhanced Self-Organizing Incremental Neural Network (ESOINN [8]) uses a threshold for each neuron, which is defined as the largest distance from its neighbors. Furthermore, in AING [9], it is given by the sum of distances from the neuron to its data-points, plus the sum of weighted distances from its neighboring neurons, averaged on the total number of the considered distances. In both cases, the influence region of the neuron depends on the extension of its neighborhood, but not on its shape. An exhaustive description can be found in [10]. However, this simple model is isotropic, in the sense that it does not take into account the orientation of the vector connecting the new data to the winner, but only its norm. Hence, it does not consider the topology of the manifold of data of the winner Voronoi set. The use of an anisotropic criterion should be justified by the need of representing in more detail the data manifold.

Data manifolds can be stationary or time-changing (i.e., non-stationary). It should be important to have a neural network able to automatically detect the data evolution. Tracking non-stationary data distributions is an important goal. This is requested by applications like real time pattern recognition: fault diagnosis, novelty detection, intrusion detection alarm systems, speech, face and text recognition, computer vision and scene analysis and so on. The existing neural solutions tackle this

problem by means of different approaches, depending both on their architecture and on the application at hand. These techniques can be mainly classified into two categories: forgetting and life-long learning networks. The first class comprises the networks with a fixed number of neurons (not incremental). Indeed, they cannot track a varying input distribution without losing the past representation (given by the old weight vectors). Furthermore, if the distribution changes abruptly (jump), they cannot track it anymore. They are used if only the most recent representation is of interest. The fastest techniques of this class are linear, like the principal component analysis (PCA) networks. However, they are not suited for non-linear problems. In this case, the best non-linear network is a variant of SOM, called DSOM [11], which is based on some changes of the SOM learning law in order to avoid a quantization proportional to the data distribution density. However, what is more interesting is the use of constant parameters (learning rate, elasticity) instead of time-decreasing ones. As a consequence, DSOM is able to promptly react to changing inputs, at the expense of forgetting the past information. Indeed, it only tracks the last changes. Forgetting networks are not suited in case the past inputs carry useful information.

Life-long learning addresses the fundamental issue of how a learning system can adapt to new information without corrupting or forgetting previously learned information, the so-called Stability-Plasticity Dilemma [12]. It should have the ability of repeatedly training a network using new data without destroying the old nodes. For this reason, they must have the capability to increase the number of neurons in order to track the distribution (the previous neurons become dead units but represent past knowledge). This kind of networks, like SOINN and its variants [8], record the whole life of the process to be modelled. The precursor is the Growing Neural Gas (GNG [13]), but it is not well suited for these problems because the instant of new node insertion is predefined by a user-dependent parameter. However, its variant GNG-U [14] is a forgetting network, which uses local utility variables to estimate the probability density of data in order to delete nodes in regions of low density.

The same observation can be repeated for the data stream clustering methods [15]. There exist techniques which can be categorized according to the nature of their underlying clustering approach, as: GNG based methods, which are incremental versions (e.g., G-Stream [16]) of the Growing Neural Gas neural network, hierarchical stream methods, like BIRCH [17] and ClusTree [18], partitioning stream methods, like CluStream [19], and density-based stream methods, like DenStream [20] and SOStream [21], which is inspired by SOM.

The first neuron layers of online Curvilinear Component Analysis (onCCA) [22] and Growing Curvilinear Component Analysis (GCCA) [23, 24, 25, 26], use the same threshold as ESOINN, but introduce the new idea of bridge, i.e. a directed interneuron connection, which signals the presence of a possible change in the data distribution. Bridges carry information about the extent of the time change by means of its length and density and allow the outlier detection.

G-EXIN

G-EXIN [27] is an online, self-organizing, incremental neural network whose number of neurons is determined by the quantization of the input space. It uses seeds to colonize a new region of the input space, and two distinct types of links (edges and bridges), to track data non-stationarity. Each neuron is equipped with a weight vector to quantize the input space and with a threshold to represent the average shape of its region of influence. In addition, it employs a new anisotropic threshold idea, based on the shape (convex hull) of neuron neighborhood to better match data topology. G-EXIN is incremental, i.e. it can increase or decrease (pruning by age) the number of neurons. It is also online: data taken directly from the input stream are fed only once to the network. The training is never stopped, and the networks keeps adapting itself to each new datum, that is, it is stochastic in nature.

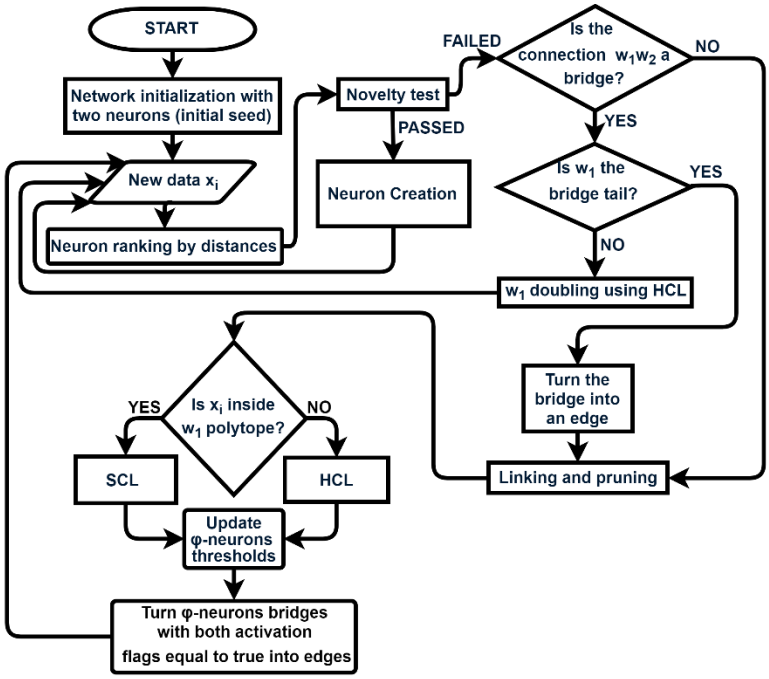

Fig. 1 G-EXIN flowchart

The G-EXIN Algorithm

The starting structure of G-EXIN is a seed (couple of neurons connected through a link) based on the first two data.

Then, each time a new datum, say $x_i \in X$, is extracted from the input stream, it is fed to the network and the training algorithm, described in Fig. 1, is performed. All neurons are sorted according to the Euclidean distances d_i between x_i and their weights. The neuron with the shortest distance (d_1) is the *first winner*, say w_1; the one with the second shortest distance (d_2) is the *second winner*, say w_2, the third one w_3, and so on. Then, the novelty test between the new data x_i and w_1 is performed. If x_i passes it, a new neuron is created; otherwise, it follows the weight adaptation, linking and doubling phase.

Novelty test. An input data x_i is considered *novel* w.r.t. the neuron γ if it satisfies two conditions: their distance d is greater than the neuron local threshold T_γ and x_i is outside the neighborhood of γ, say NG_γ.

T_γ provides the minimal resolution of the test. Indeed, if a lower threshold is not given, there is the potential risk of a too large amount of neurons. The choice of this minimum implies neighbor neurons are not too close, which results in a apriori granularity (resolution). T_γ represents the radius of a hypersphere centered on the neuron. It is given by the average of the distances between γ and its topological neighbors according to:

$$T_\gamma = \frac{1}{|NG_\gamma|} \sum_{w_i \in NG_\gamma} \|\gamma - w_i\| \tag{1}$$

The neighborhood NG_γ can be represented in different ways. However, if we want to respect its geometry and, in the same time, to avoid complicating too much the model, a good compromise is the convex hull (bounded convex polytope) of the weight vector of neuron γ and the weights of its topological neighbors. Indeed, it is simple linear approach that considers not only the neighbors, but also the directionality of the corresponding edges, which implies to take into account the anisotropy of the region of influence. In this context, neurons connected through bridges are excluded, only those connected through edges are taken into account.

Depending on the network configuration, two scenarios can occur:

1) γ has less than two topological neighbors, then, it is impossible to build the convex hull. In this case, for the novelty detection, only the isotropic hypersphere centered on γ and with radius T_γ is used. If the input data x_i is outside the sphere, then the novelty test is passed, otherwise, it is failed.

2) γ has at least two topological neighbors then, for the novelty detection, a more sophisticated strategy is adopted. First, the convex hull of γ and its topological neurons is built. Then, if d is sufficiently big (i.e. greater than T_γ) the isotropic hypersphere with radius T_γ is replaced by the following simple and time-efficient anisotropic test to determine if x_i belongs or not to the NG_γ region. The difference vectors δ_i between x_i and NG_γ weight vectors and their sum vector $\psi = \Sigma\, \delta_i$ are computed. If all the

scalar products between δ_i and ψ have the same sign (null products are ignored), then x_i is outside the polytope. Otherwise, x_i is inside the polytope.

Neuron creation. If x_i passes the novelty test, a new neuron, whose weight vector is given by x_i, is created. w_1 is linked to x_i by a bridge and their *activation flags* are set to *false*. Finally, T_{x_i} is set equal to d_1.

Adaptation, linking and doubling. If x_i fails the novelty test, it is checked if the first winner, whose weight is w_1, and the second winner, whose weight is w_2, are connected by a bridge:

1. *If there is no bridge*, these two neurons are linked by an edge (whose age is set to zero) and the same age procedure as onCCA is used as follows. The age of all other links of NG_{w_1} is incremented by one; if a link age is greater than the age_{max} scalar parameter, it is eliminated. If a neuron remains without links, it is removed (*pruning*). Then:
   a) *if x_i is inside NG_{w_1}* (i.e. inside the convex hull), x_i neighbor neuron weights are adapted according to the Soft Competitive Learning (SCL):

$$\Delta w_i = \alpha_1(w_i - x_i) \qquad \mathrm{i} = 1 \tag{2a}$$
$$\Delta w_i = \alpha_n(w_i - x_i) \quad \text{otherwise} \tag{2b}$$

   where $\alpha_1 = \frac{\alpha}{N_i}$ as in k-means [18] and $\alpha_n = \alpha * \exp(-\frac{(w_i - x_i)^2}{2\sigma^2})$. Here, N_i is the total number of times w_i has been the first winner and, α and σ are two user-dependent parameters.
   b) *if x_i is outside NG_{w_1}*, only (2a) is used (Hard Competitive Learning, HCL).

   Next, for all the neurons that have been moved, i.e. whose weight vector has changed, say *φ-neurons*, their thresholds are recomputed, and their *activation flags* are set to *true*.
   Finally, all the *φ-neurons* bridges, both ingoing and outgoing, are checked and all those which have both neurons at their ends with *activation flags* equal to true become edges.

2. *If there is a bridge*, it is checked if w_1 is the bridge tail; in this case, step 1 is performed and the bridge becomes an edge. Otherwise, a seed is created by means of the neuron doubling:
   a) a virtual adaptation of the w_1 weight is estimated by HCL (only (2a) is used) and considered as the weight of a new neuron (doubling).

b) w_1 and the new neuron are linked with an edge (age set to zero) and their thresholds are computed (they correspond to their Euclidean distance).

Growing Curvilinear Component Analysis (GCCA)

Dealing with time-varying high dimensional data is a big problem for real time pattern recognition. Only linear projections, like principal component analysis, are used in real time while nonlinear techniques need the whole database (offline). On the contrary, when working in real time requires a data stream, that is, a continuous input, the algorithm needs to be defined as online. This is the case, for example, of fault and pre-fault diagnosis and system modeling.

The techniques and the concepts presented above can be applied to different scenarios and applications. For instance, they can be used to perform an online quantization and dimensionality reduction (DR) of the input data, such as in the Growing Curvilinear Component Analysis (GCCA) neural network.

GCCA, whose flowchart is shown in Fig. 2, has a self-organized incremental (pruning by age) architecture, which adapts to the nonstationary data distribution. It performs simultaneously the data quantization and projection. The former is based on G-EXIN in the sense that it exploits the same techniques, such as seeds and bridges, to perform an online clustering of the input space. Seeds are pairs of neurons which colonize the input domain, bridges are a different kind of edge in the manifold graph, signaling the data non-stationarity. The input projection is done using the Curvilinear Component Analysis (CCA), a distance-preserving reduction technique, here called *offline CCA*.

Data projection is a tool used frequently as a preprocessing stage; therefore, in a scenario such as that one characterized by an input fast-changing data stream (e.g. fault and pre-fault diagnosis), it needs to be as fast as possible. For this reason, the use of convex polytopes has been avoided and the novelty test is based only on the isotropic hypersphere whose radius is locally computed as the average of the distances from a neuron and its neighbors. The remaining has been designed as in G-EXIN with the difference that each neuron is equipped with two weight-vector, one in the input space X and one in the projected space Y. Moreover, an additional hyperparameter, λ, is needed, as in CCA, to tune the projection mechanism.

The projection works as follows. For each pair of different weight vectors in the X space (input space), a between-point distance D_{ij} is calculated as $D_{ij} = \|x_i - x_j\|$. At the same time, the distance L_{ij} of the associated Y-weights in the latent space, is computed as $L_{ij} = \|y_i - y_j\|$. CCA aims to project data such that $L_{ij} = D_{ij}$. Obviously, this is possible only if all input data lay on a linear manifold. In order to face this problem, CCA defines a distance function, which, in its simplest form, is the following:

$$F_{\lambda}(L_{ij}) = \begin{cases} 0 & \text{if } \lambda < L_{ij} \\ 1 & \text{if } \lambda \geq L_{ij} \end{cases} \tag{3}$$

That is a step function for constraining only the under threshold between-point distances L_{ij}. In this way, the CCA favors short distances, which implies local distance preservation.

Defining as $\boldsymbol{y}(j)$ the weight of the j-th projecting neuron in the Y space, the stochastic gradient algorithm for minimizing the error function follows:

$$\mathbf{y}(j) \leftarrow \mathbf{y}(j) + \alpha(D_{ij} - L_{ij}) F_{\lambda}(L_{ij}) \frac{\mathbf{y}(j) - \mathbf{y}(i)}{L_{ij}} \tag{4}$$

where α is the learning rate.

Each time a datum fails the *novelty test* a new neuron is created. As in G-EXIN, its weight vector in the input space X is the datum itself. To determine the weight in the latent space, i.e. the Y-weight, a two-step procedure is applied. First, the starting projection (y_0) is estimated using the *triangulation* technique defined in [23]. To compute y_0 the winner and second winner projections are used as the centers of the two circles, whose radii are the distances in data space from the input data, respectively. The circles intersect in two points, the farthest from the third winner projection is chosen as the initial y_0. Then, y_0 is refined with one or several CCA iterations (4), in which the first and second winner projections are considered as fixed (*extrapolation*).

The same projecting algorithm is applied in case of neuron doubling. In this case, the new neuron to be considered as input is w_{1new}, that is the unit just born from the first winner w_1 doubling.

On the other side, if the datum fails the novelty test, the CHL and the SCL techniques are applied. Due to the weight updates of SCL, the first winner and its neighbors' distances D_{ij} change. Hence, the projections of the neuron whose distances from w_1 have to be updated. The CCA rule (4) is used but in an opposite way (*interpolation*). The first winner projection is fixed and the other neuron projection are moved accordingly to (4).

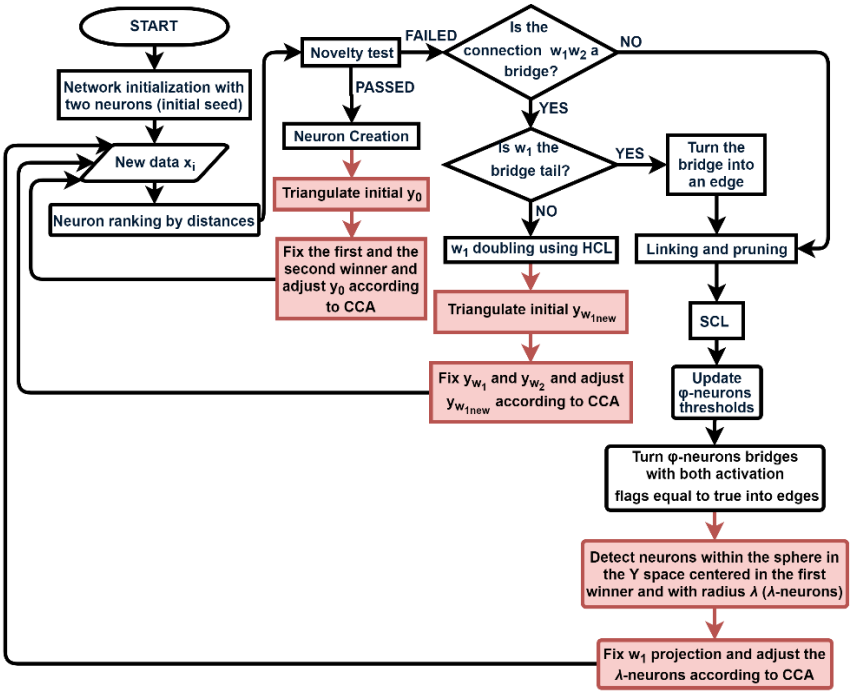

Fig. 2 GCCA flowchart: black blocks deal with G-EXIN quantization while red ones, specifically, with GCCA projection

GH-EXIN

Hierarchical clustering is an important technique to retrieve multi-resolution information from data. It creates a tree of clusters, which corresponds to different resolution of data analysis. Generally, e.g. in data mining, the outcome is a richer information if compared with plain clustering.

The growing hierarchical GH-EXIN [28], [29] neural network builds a hierarchical tree based on a stationary variant (i.e. without bridges) of G-EXIN, called sG-EXIN. As before, the network is both incremental (data-driven) and self-organized. It is a top-down, divisive technique, in which all data start in a single cluster and, then, splits are done recursively until all clusters satisfy certain conditions.

The algorithm starts from a single root node, which has associated fictitiously the whole dataset; then, using vertical and horizontal growths, it builds a hierarchical tree. Vertical growth refers to the addition of further layers to leaf nodes until a higher resolution is needed; it always implies the creation of a seed, i.e. a pair of neurons, which represents the starting structure of a new sG-EXIN neural network. On the other side, horizontal growth is the process of adding further neurons to the seed. This characteristic is important in order to be able to create complex hierarchical structures; indeed, without it, it would be possible to build only binary trees. This process is performed by the neuron creation mechanism during the sG-EXIN

training. As G-EXIN, GH-EXIN uses convex hull to define neuron neighborhood, which implies the anisotropic region of influence for the horizontal growth. In addition, upon time, it performs outlier detection and, when needed, it reallocates their data by using a novel simultaneous approach on all the leaves.

The GH-EXIN training algorithm starts, as already mentioned, from a single root node whose Voronoi set is the whole input dataset. It is considered as the initial father node. A father neuron Ψ is the basis for a further growth of the tree; indeed, new leaves are created (vertical growth), whose father is Ψ and whose Voronoi sets are a partition (i.e. a grouping of a set's elements into non-empty subsets, whose intersection is the empty set) of the Ψ one. More specifically, for each father neuron Ψ, which does not satisfy the vertical growth stop criterion, a new seed is created as in G-EXIN and, then, an sG-EXIN neural network is trained using the father Voronoi set as training set. The neurons yielded by the training, which defines a so-called *neural unit*, became the sons of Ψ in the tree determining a partition of its Voronoi set. If the resulting network does not satisfy the horizontal growth stop criterion, the training is repeated for further epochs (i.e. presentation of the whole Ψ dataset) until the criterion is fulfilled.

At the end of each training epoch, if a neuron remains unconnected (no neighbors) or is still lonely, it is pruned, but the associated data are analyzed and possibly reassigned as explained later in this section.

At the end of each horizontal growth, the *topology abstraction check* is performed to search for connected components within the graph of the resulting neural unit. If more than one connected component is detected, the algorithm tries to extract an abstract representation of data; at this purpose, each connected component, representing a cluster of data, is associated with a novel abstract neuron, which becomes the father node of the connected component neurons, determining a double simultaneous vertical growth. As weight vectors of the abstract neurons are used the centroids of the clusters they represent.

Then, each leaf, in the same level of the hierarchy of Ψ, that does not satisfy the vertical growth stop criterion, is considered as a father node and the growth algorithm is repeated, until no more leaves are available in that specific level.

Finally, the overall above procedure is repeated on all the leaves of the novel, deeper level yielded from the previous vertical growth; therefore, the tree can keep growing until the needed resolution is reached, that is, until the vertical growth stop criterion is satisfied for all the leaves of the tree.

It is worth to be noticed that such mechanism allows a simultaneous vertical and horizontal growth; indeed, due to node creation (seed) below a father an additional level is added to the tree (i.e. vertical growth) and, at the same time, thanks to sG-EXIN training, several nodes are added to the same level (i.e. horizontal growth).

The novelty test (Semi-Isotropic Region of Influence), the weights update (SCL) and the pruning mechanism (pruning by age) are the same as in G-EXIN. The difference is that GH-EXIN is based on sG-EXIN which, as stated above, does not have bridges; as a consequence, each time a new neuron is created along the GH-EXIN training process, it is created as a *lonely neuron*, that is a neuron with no

edges. Then, in the next iterations connections may be created according to the Competitive Hebbian Rule; if, at the end of the epoch, the neuron is still lonely, it will be removed according to the pruning rule.

When a neuron is removed, its Voronoi set data remain orphans and are labelled as *potential outliers* to be checked at the end of each epoch; for each potential outlier x, i.e. each datum, GH-EXIN seeks a possible new candidate among all leaf nodes. If the closest neuron w among the remaining, i.e. the new winner, belongs to the same neural unit of x but the datum is outside its region of influence (the hypersphere and the convex-hull), x is not reassigned; otherwise, if x is within a winner region of influence within the same neural unit or in case the winner belongs to another neural unit, it is reassigned to the winner Voronoi.

The growth stop criteria are used to drive, in an adaptive way, the quantization process; for this reason, they are both based on the H index, which, depends on the application at hand, and it is used to measure clusters heterogeneity and purity, i.e. their quality. For the horizontal growth, the idea is to check if the H average estimated value of the neurons of the neural unit being built falls below a percentage of the value of the father node. On the other side, in the vertical growth stop criterion, a global user-dependent threshold is used for H; at the same time, to avoid too small, meaningless clusters, a min_{card} parameter is used to establish the minimum cardinality of Voronoi sets, i.e. the maximum meaningful resolution.

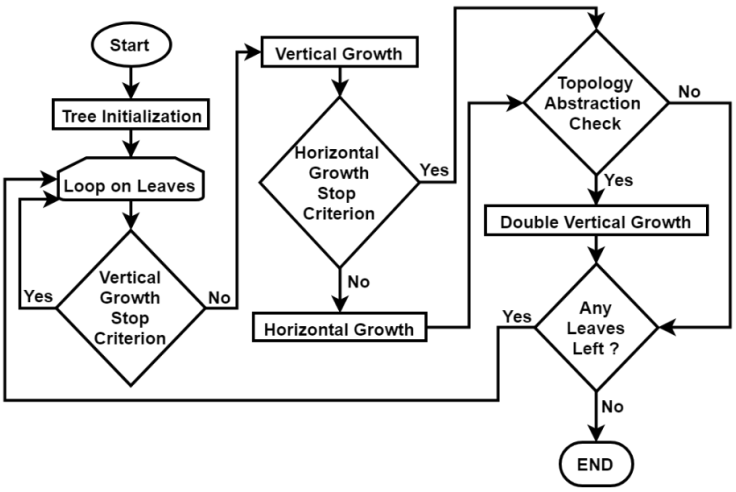

Fig. 3 GH-EXIN flowchart

Experiments

The performance of the above-mentioned neural networks has been tested on both synthetic and real experiments. The aim has been to check their clustering capabilities and to assess their specific abilities (e.g. projection).

G-EXIN

The first experiment deals with data drawn uniformly from a 5000-points square distribution, which, after an initial steady state (stationary phase), starts to move vertically (non-stationary phase). Indeed, in the beginning, the network is trained with data randomly extracted (without repetition) from the 5000-points square. Then, after the presentation of the whole training set, the (support of the) distribution starts to move monotonically, with constant velocity, along the y-axis in the positive direction. The results of G-EXIN ($age_{max} = 2$, $\alpha = 1$, $\sigma = 0.03$) are presented in Fig. 4 and Fig. 5 both for the stationary and non-stationary phases, respectively. Firstly, the network is able to properly quantize the input distribution even along its borders; then, it is able to fully understand the data evolution over time and to track it after the end of the steady state. The importance of the density of bridges as a signal of non-stationarity is also revealed in Fig. 6, which shows how the number of bridges changes in time. In particular, the growth is linear, which is a consequence of the constant velocity of the distribution. G-EXIN correctly judges the data stream as drawn by a single distribution with fully connected support, thanks to its links (i.e., edges and bridges). Fig. 5 also shows G-EXIN performs life-long learning, in the sense that previous quantization is not forgotten.

Resuming, the use of different, specific, anisotropic, links has been proved to be an appropriate solution to track non-stationary input changes into the input distribution.

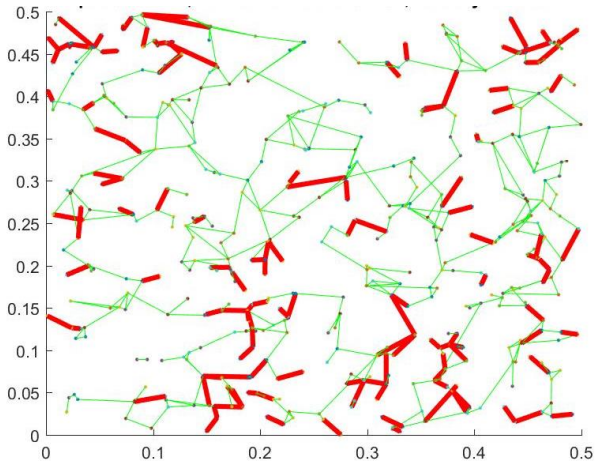

Fig. 4 G-EXIN: vertical moving square, stationary phase. Neurons (circles) and their links: edges (green), bridges (red).

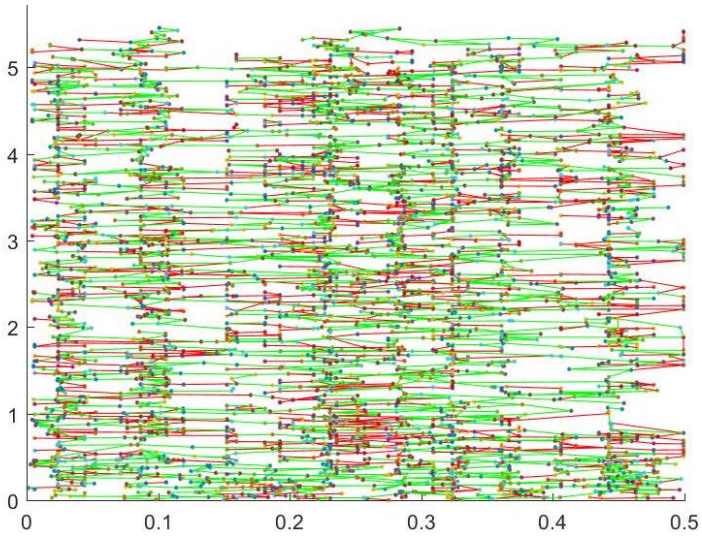

Fig. 5 G-EXIN: vertical moving square, non-stationary phase. Neurons (circles) and their links: edges (green), bridges (red).

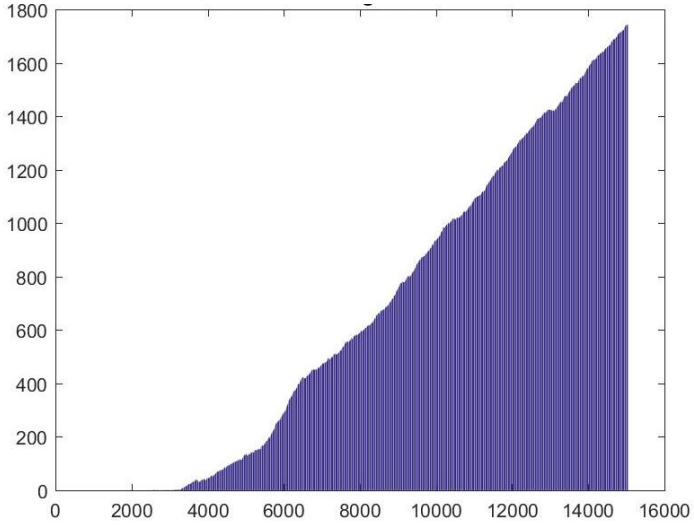

Fig. 6 G-EXIN: vertical moving square, number of bridges (Y-axis) over time (X-axis)

The second experiment deals with data drawn uniformly from a 5000-points square distribution whose support changes abruptly (jump) three times (from NW to NE, then from NE to SW and, finally, from SW to SE), in order to test on abrupt changes. Fig. 7 shows the results of G-EXIN (age_{max} = 9, α = 1, σ = 0.06) on such dataset, where neuron weights are represented as small dots and links as green

(edges) and red segments (bridges); the same color is used for all neurons because the network does not perform any classification task.

Not only G-EXIN learns the data topology and preserves all the information without forgetting the previous history, as in the previous experiment, but it is able to track an abrupt change in the distribution by means of a single, long, bridge. The length of the bridges is proportional to the extent of the distribution change.

Fig. 7 also shows the G-EXIN graph is able to represent well the borders of the squares because of its anisotropic threshold. On the contrary, this is not possible with a simpler isotropic technique.

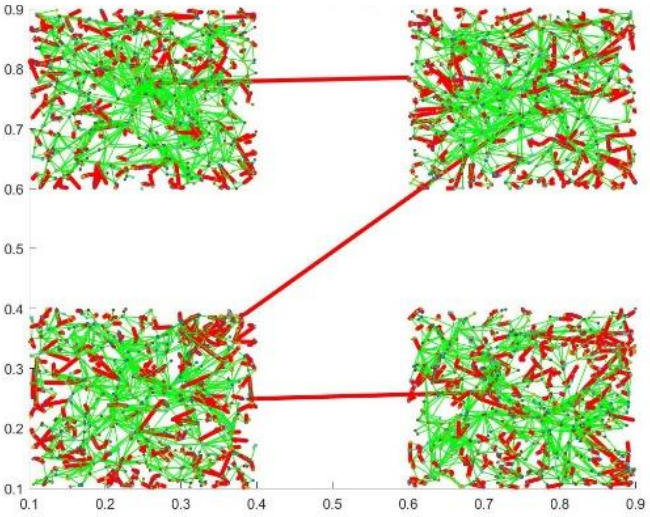

Fig. 7 G-EXIN: three jumps moving square. Neurons (circles) and their links: edges (green), bridges (red).

The third experiment deals with a more challenging problem: data drawn from a dataset coming from the bearing failure diagnostic and prognostic platform [30], which provides access to accelerated bearing degradation test. In particular, the test is based on a non-stationary framework that evolves from an initial transient to its healthy state to a double fault. Fig. 8 shows G-EXIN ($age_{max} = 3$, $\alpha = 0.2$, $\sigma = 0.01$) on the experiment dataset during the complete life of the bearing: the initial transient, the healthy state and the following deterioration (the structure and color legenda are the same as in the previous figures). The transient phase is visible as the small cluster in the bottom left part of the figure. Then, the long vertical bridge signals the onset of the healthy state, which is represented as the central region made neurons connected by green and red edges. Finally, on the right and upper of this region there is the formation of longer and longer bridges which detect the deterioration of the bearing.

Resuming, all these experiments have shown that G-EXIN is able to fully track the non-stationarity by means of bridges, whose length and density carry information on the extent of the non-stationarity of the data distribution.

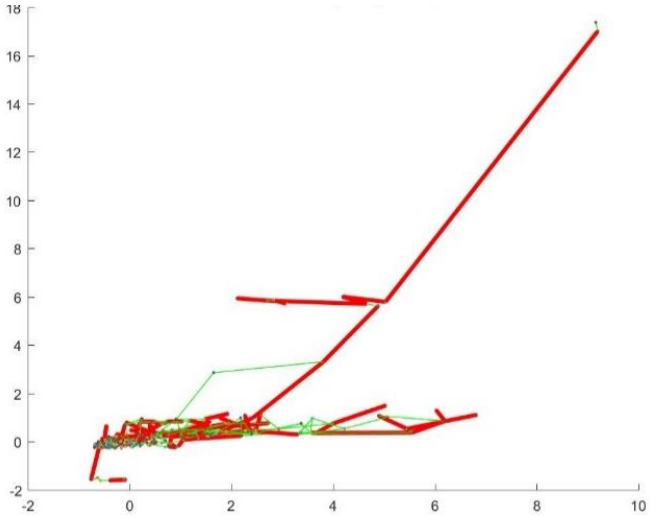

Fig. 8 G-EXIN: bearing fault experiment. Neurons (circles) and their links: edges (green), bridges (red).

GCCA

The simulation for GCCA deals with a more challenging synthetic problem: data drawn from a uniform distribution whose domain is given by two interlocked rings (see Fig. 9 upper left). Using a batch of 1400 data the projection of the *offline CCA* has been computed, by using a number of *epochs* equal to 10 and λ equal to 1. Fig. 9 lower left shows that the *offline CCA* correctly unfolds data (the rings are separated). GCCA has then been applied to the same problem. The following parameters have been chosen: $age_{max} = 2$, $\alpha = 1$, $\sigma = 0.03$, $\lambda = 0.05$. Fig. 9 upper right shows the result of the input space quantization together with the initial dataset. Fig. 9 lower right yields the GCCA projection. There is a good unfolding (separation) in both the projections; however, it is evident from Fig. 9 that GCCA online projection, based on a single epoch, performs as good as the offline CCA, which, on the contrary, needs 10 presentations, i.e. *epochs*, of the training set.

In order to check the robustness of GCCA to white noise, an additional experiment has been made, starting from the same training set, but adding a Gaussian noise of zero mean and standard deviation set to 0.1. Fig. 10 top left shows the resulting noisy distribution. The parameters are the same as in the previous experiment. Fig. 10 top right yields the X-weight quantization of GCCA. Fig. 10 bottom left and bottom right show the results of *offline CCA* and GCCA, respectively. It can be observed not only the robustness of GCCA, but also the better accuracy of its projection w.r.t. the *offline CCA*, trained on a batch composed of the same data presented to GCCA.

From the previous simulations and logical considerations, some conclusions about the features of GCCA can be drawn. It retains the same properties of the *of-*

fline CCA, that is the topological preservation of smallest distances and the unfolding of data. The adaptive features allow non-stationary data to be tracked by means of the quantization and the corresponding projection. Finally, GCCA is inherently robust to noise.

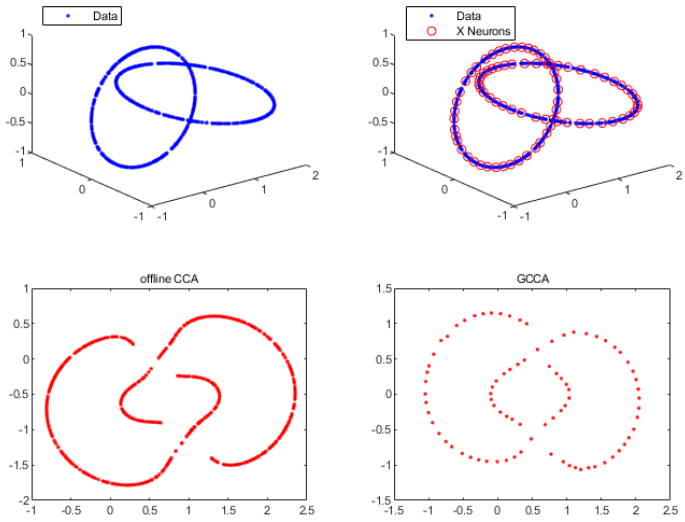

Fig. 9 GCCA: interlocked rings - no noise

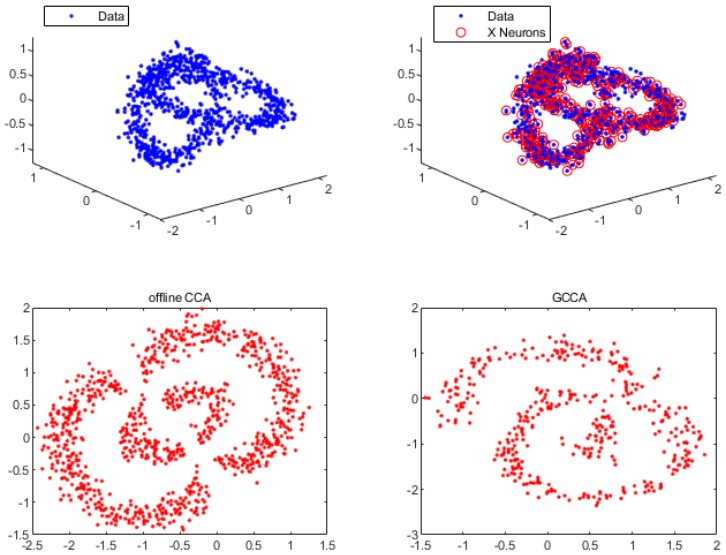

Fig. 10 GCCA: interlocked rings - Gaussian noise

GH-EXIN

Considering that GH-EXIN has been conceived for hierarchical clustering, a dataset composed of two Gaussian mixture models has been devised: the first model is made of three Gaussians, the second one of four Gaussians, as shown in Fig. 11.

The results, visualized in Fig. 12 and Fig. 13, clearly show that GH-EXIN (H_{max} = 0.001, H_{perc} = 0.9, $\alpha_{\gamma 0}$ = 0.5, α_{i0} = 0.05, age_{max} = 5, min_{card} = 300) builds the correct hierarchy (the tree is visualized in Fig. 14): two nodes in the first layer (level), which represent the two clusters, and as many leaves as Gaussians in the second layer, which represent the mixtures. Neurons are also positioned correctly w.r.t. the centers of the Gaussians.

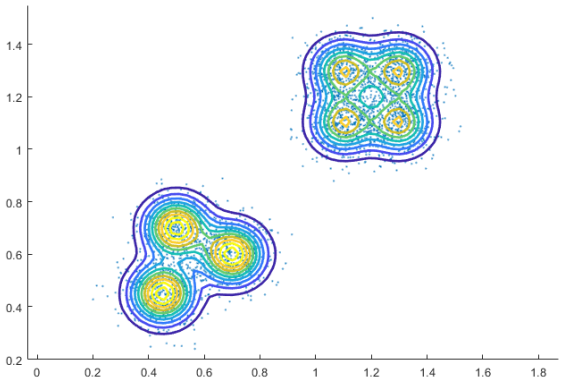

Fig. 11 GH-EXIN: Gaussian dataset. Data (blue points) and contours

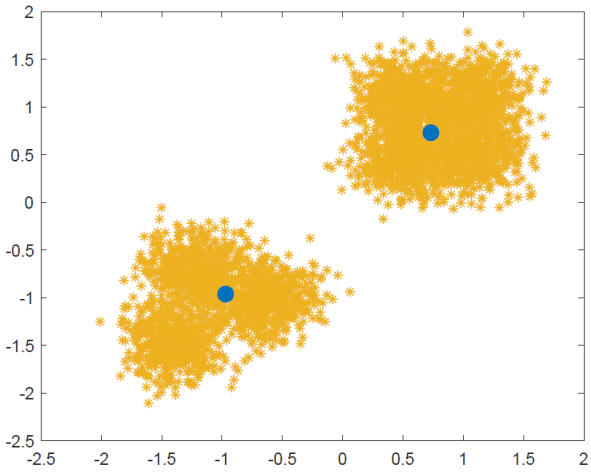

Fig. 12 GH-EXIN: Gaussian dataset, first level of the hierarchy. Data (yellow points) and neurons (blue points).

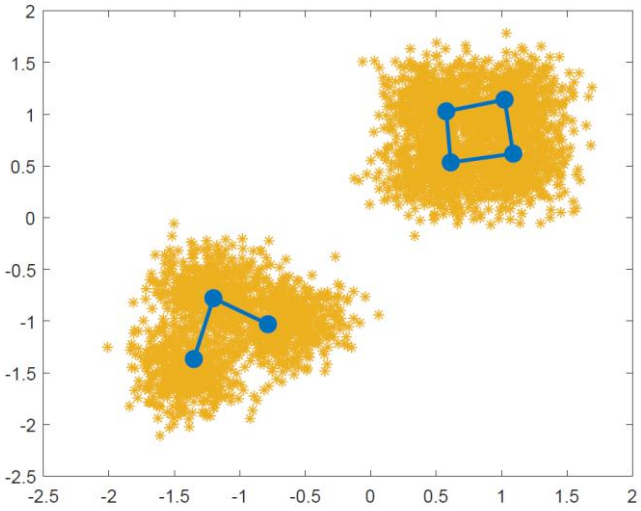

Fig. 13 GH-EXIN: Gaussian dataset, second level of the hierarchy. Data (yellow points) and neurons (blue points).

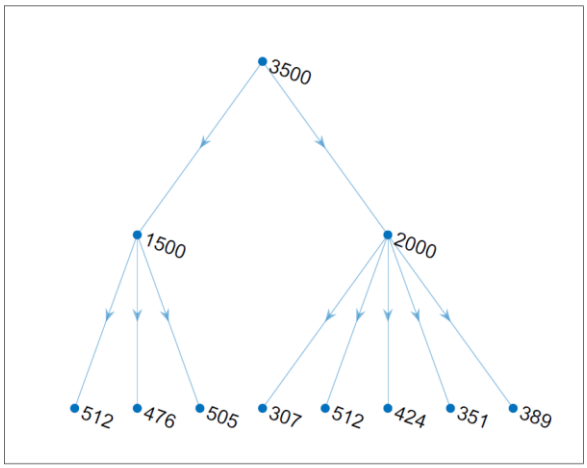

Fig. 14 GH-EXIN: Gaussian dataset, final tree and cardinality of nodes and leaves

Conclusions

This chapter addresses the problem of inferring information from unlabeled data drawn from stationary or non-stationary distributions. At this aim, a family of novel unsupervised neural networks has been introduced. The basic ideas are implemented in the G-EXIN neural network, which is the basic tool of the family. The

other neural networks, GCCA and GH-EXIN, are extensions of G-EXIN, for dimensionality reduction and hierarchical clustering, respectively. All these networks exploit new peculiar tools: bridges, which are links for detecting changes in the data distribution; anisotropic threshold for taking into account the shape of the distribution; seed and associated neuron doubling for the *colonization* of new distributions; soft-competitive learning with the use of a Gaussian to represent the winner neighborhood.

The experiments show these neural networks work well both for synthetic and real experiments. In particular, they perform long-life learning, build a quantization of the input space, represent the data topology with edges and the non-stationarity with bridges, perform the CCA non-linear dimensionality reduction with an accuracy comparable to the offline CCA, yield the correct tree in case of hierarchical clustering. These are fast algorithms that require only a few user-dependent parameters.

Future work will deal with the search of new *automatic* variants, which self-calibrate their parameters, and more challenging applications.

References

[1] Y. Linde, A. Buzo e R. Gray, «An algorithm for vector quantizer design,» *IEEE Transactions on Communication,* vol. 28, pp. 84-95, 1980.

[2] J. MacQueen, «Some methods for classification and analysis of multivariate observations,» in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley (USA), 1967.

[3] T. Martinetz e K. Schulten, «A "neural-gas" network learns topologies,» *Artificial Neural Networks,* pp. 397-402, 1991.

[4] T. Kohonen, «Self-organized formation of topologically correct feature maps,» *Biological Cybernetics,* vol. 43, pp. 59-69, 1982.

[5] R. H. White, «Competitive hebbian learning: Algorithm and demonstrations,» *Neural Networks,* vol. 20, n. 2, pp. 261-275, 1992.

[6] T. Martinetz e K. Schulten, «Topology representing networks,» *Neural Networks,* vol. 7, n. 3, pp. 507-522, 1994.

[7] Y. Prudent e A. Ennaji, «An incremental growing neural gas learns topologies,» in *Proceedings of the IEEE International Joint Conference on Neural Networks*, Motréal, Quebec, Canada, 2005.

[8] S. Furao, T. Ogurab and O. Hasegawab, "An enhanced self-organizing incremental neural," *Neural Networks,* vol. 20, p. 893–903, 2007.

[9] M. Bouguelia, B. Y. e B. A., «An adaptive incremental clustering method based on the growing neural gas algorithm,» in *2nd International Conference on Pattern Recognition Applications and Methods ICPRAM 2013*, Barcelona, (Spain), 2013.

[10] M. Bouguelia, B. Y. e B. A., «Online Unsupervised Neural-Gas Learning Method for Infinite,» *Pattern Recognition Applications and Methods,* p. 57–70, 2015.

[11] N. P. Rougier e Y. Boniface, «Dynamic self-organizing map,» *Neurocomputing,* vol. 74, n. 11, pp. 1840-1847, 2011.

[12] G. Carpenter e S. Grossberg, «The ART of adaptive pattern recognition by a self-organizing neural network,» *IEEE Computer Society,* vol. 21, p. 77–88, 1988.

[13] B. Fritzke, «A growing neural gas network learns topologies,» *Advances in Neural Information Processing System,* vol. 7, pp. 625-632, 1995.

[14] B. Fritzke, «A self-organizing network that can follow non-stationary distributions,» in *Proceedings of ICANN 97, International Conference on Artificial Neural Networks*, Lausanne, Switzerland, 1997.

[15] M. Ghesmoune, M. Lebbah e H. Azzag, «State-of-the-art on clustering data streams,» *Big Data Analytics,* pp. 1-13, 2016.

[16] M. Ghesmoune, H. Azzag e M. Lebbah, «G-stream: Growing neural gas over data stream,» in *In Neural Information Processing, 21st International Conference, ICONIP*, Kuching, Malaysia, 2014.

[17] T. Zhang, R. Ramakrishnan e M. Livny, «BIRCH: an efficient data clustering method for very large databases,» in *SIGMOD Conference*, New York, 1996.

[18] P. Kranen, I. Assent, C. Baldauf e T. Seidl, «The ClusTree indexing microclusters for anytime stream mining,» *Knowledge Inf. Syst.,* vol. 29, n. 2, pp. 249-272, 2011.

[19] C. Aggarwal, T. Watson, R. Ctr, J. Han, J. Wang e P. Yu, «A framework for clustering evolving data streams,» in *VLDB2003 Proceedings of the VLDB Endowment*, Berlin, 2003.

[20] F. Cao, M. Ester, W. Qian e A. Zhou, «Density-based clustering over an evolving data stream with noise,» in *SIAM International Conference on Data Mining (SDM06)*, Maryland, 2006.

[21] C. Isaksson, M. Dunham e M. Hahsler, «SOStream: Self organizing density-based clustering over data stream,» in *8th International Conference on Machine Learning and Data Mining MLDM 2012*, Berlin, 2012.

[22] G. Cirrincione, J. Hérault e V. Randazzo, «The on-line Curvilinear Component Analysis (onCCA) for real-time data reduction,» in *Proceedings of the IEEE International Joint Conference on Neural Networks*, Killarney (Ireland), 2015.

[23] G. Cirrincione, V. Randazzo and E. Pasero, "Growing Curvilinear Component Analysis (GCCA) for Dimensionality Reduction of Nonstationary Data," in *Multidisciplinary Approaches to Neural Computing*, Springer International Publishing, 2018, pp. 151-160.

[24] R. Kumar, V. Randazzo, G. Cirrincione, M. Cirrincione e E. Pasero, «Analysis of stator faults in induction machines using Growing Curvilinear Component Analysis,» in *International Conference on Electrical Machines and Systems ICEMS2017*, Sydney (Australia), 2017.

[25] G. Cirrincione, V. Randazzo e E. Pasero, «The Growing Curvilinear Component Analysis (GCCA) neural network,» *Neural Networks,* p. 108–117, 2018.

[26] G. Cirrincione, V. Randazzo, R. Kumar, M. Cirrincione e E. Pasero, «Growing Curvilinear Component Analysis (GCCA) for Stator Fault Detection in Induction Machines,» in *Neural Approaches to Dynamics of Signal Exchanges*, Springer International Publishing, 2019.

[27] V. Randazzo, G. Cirrincione, G. Ciravegna e E. Pasero, «Nonstationary Topological Learning with Bridges and Convex Polytopes: the G-EXIN Neural Network,» in *2018 International Joint Conference on Neural Networks (IJCNN)*, Rio de Janeiro, 2018.

[28] P. Barbiero, A. Bertotti, G. Ciravegna, G. Cirrincione, E. Pasero e E. Piccolo, «Unsupervised Gene Identification in Colorectal Cancer,» in *Quantifying and Processing Biomedical and Behavioral Signals*, Springer International Publishing, 2018, p. 219–227.

[29] P. Barbiero, G. Ciravegna, E. Piccolo, G. Cirrincione, M. Cirrincione e A. Bertotti, «Neural biclustering in gene expression analysis,» in *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, 2017.

[30] N. A. R. Center, «FEMTO Bearing Data Set, NASA Ames Prognostics Data Repository,» [Online]. Available: http://ti.arc.nasa.gov/project/prognostic-data-repository.