POLITECNICO DI TORINO
Repository ISTITUZIONALE

The Growing Curvilinear Component Analysis (GCCA) neural network

Original
The Growing Curvilinear Component Analysis (GCCA) neural network / Cirrincione, G., Randazzo, V., Pasero, E.. - In: NEURAL NETWORKS. - ISSN 0893-6080. - ELETTRONICO. - 103:(2018), pp. 108-117. [10.1016/j.neunet.2018.03.017]

Availability:
This version is available at: 11583/2706354 since: 2021-04-06T17:40:32Z

Publisher:
Elsevier Ltd

Published
DOI:10.1016/j.neunet.2018.03.017

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Elsevier postprint/Author's Accepted Manuscript

© 2018. This manuscript version is made available under the CC-BY-NC-ND 4.0 license http://creativecommons.org/licenses/by-nc-nd/4.0/.The final authenticated version is available online at: http://dx.doi.org/10.1016/j.neunet.2018.03.017

(Article begins on next page)

26 October 2026

The Growing Curvilinear Component Analysis (GCCA) Neural Network

Giansalvo Cirrincione[a], Vincenzo Randazzo[b] and Eros Pasero[b]

[a]*University of Picardie Jules Verne, Amiens, FR and with the University of South Pacific, Suva, FJ. E-mail: giansalvo.cirrincione@u-picardie.fr*
[b]*Department of Electronics and Telecommunications, Politecnico di Torino, Turin, IT. E-mail: {vincenzo.randazzo, eros.pasero}@polito.it*

Abstract

Big high dimensional data is becoming a challenging field of research. There exist a lot of techniques which infer information. However, because of the curse of dimensionality, a necessary step is the dimensionality reduction (DR) of the information. DR can be performed by linear and nonlinear algorithms. In general, linear algorithms are faster because of less computational burden. A related problem is dealing with time-varying high dimensional data, where the time dependence is due to nonstationary data distribution. Data stream algorithms are not able to project in lower dimensional spaces. Indeed, only linear projections, like principal component analysis (PCA), are used in real time while nonlinear techniques need the whole database (offline). The Growing Curvilinear Component Analysis (GCCA) neural network addresses this problem; it has a self-organized incremental architecture adapting to the changing data distribution and performs simultaneously the data quantization and projection by using CCA, a nonlinear distance-preserving reduction technique. This is achieved by introducing the idea of "seed", pair of neurons which colonize the input domain, and "bridge", a novel kind of edge in the manifold graph, which signals the data non-stationarity. Some artificial examples and a real application are given, with a comparison with other existing techniques.

Keywords: curvilinear component analysis; dimensionality reduction; neural network; non-stationary data; soft competitive learning; bridge

1. Introduction

DATA mining is more and more facing the extraction of meaningful information from big data (e.g. from internet), which is often very high dimensional. For both visualization and automatic purposes, their dimensionality has to be reduced. This is also important in order to learn the data manifold, which, in general, is lower dimensional than the original data. Dimensionality reduction (DR) also mitigates the curse of dimensionality: e.g., it eases classification, analysis and compression of high-dimensional data.

Most DR techniques work offline, i.e. they require a static database (batch) of data, whose dimensionality is reduced. They can be divided into linear and nonlinear techniques, the latter being in general slower, but more accurate in real world scenarios. See for an overview [1].

However, the possibility of using a DR technique working in real time is very important, because it allows not only having a projection after only the presentation of few data (i.e. a very fast projection response), but also tracking non-stationary data distributions (e.g. time-varying data manifolds). This can be applied, for example, to all applications of real time pattern recognition, where the data reduction step plays a very important role: fault diagnosis, novelty detection, intrusion detection for alarm systems, speech, face and text recognition, computer vision and scene analysis and so on.

Working in real time requires a data stream, a continuous input for the DR algorithms, which are defined as on-line or, sometimes, incremental (synonym of non-batch). They require, in general, data drawn from a stationary distribution. The fastest algorithms are linear and use the Principal Component Analysis (PCA) by means of linear neural networks, like the Generalized Hebbian Algorithm (GHA, [2]), the Adaptive Principal-component Extractor (APEX, [3]) and the incremental PCA (candid covariance-free CCIPCA [4]).

Nonlinear DR techniques are not suitable for online applications. Many efforts have been tried in order to speed-up these algorithms: updating the structure information (graph), new data prediction, embedding updating. However, these incremental versions (e.g. iterative LLE, [5], incremental Laplacian eigenmaps [6], incremental Hessian LLE [7]) require too a cumbersome computational burden and are useless in real time applications.

Neural networks can also be used for data projection. In general, they are trained offline and used in real time (recall

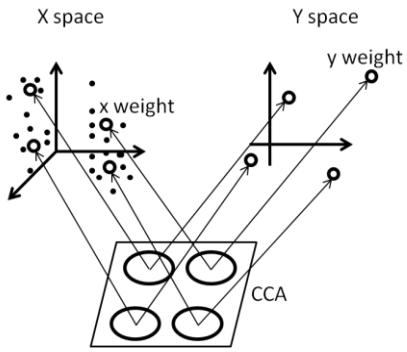

Fig. 1 The CCA neural network and its two weight layers

phase). In this case, they work only for stationary data and can be better considered as implicit models of the embedding. Radial basis functions and multilayer perceptrons work well for this purpose (out-of-sample techniques). However, their adaptivity can be exploited either by creating ad hoc architectures and error functions or by using self-organizing maps (SOM) and variants [9]. The former comprises multilayer perceptrons trained on a precomputed Sammon's mapping or with a backpropagation rule based on the Sammon's technique and an unsupervised architecture (SAMANN, [10]). These techniques require the stationarity of their training set. The same problem affects the deep neural autoencoders [11], which are trained for modelling the data reduction. They are multilayer feed-forward neural networks with an odd number of hidden layers and shared weights between the top and bottom layers; sigmoid activation functions are generally used (except in the middle layer, where a linear activation function is usually employed). The main weakness of autoencoders is that their training may converge very slowly, especially in cases where the input and target dimensionality are very high (since this yields a high number of weights in the network). In addition, they are limited by the presence of local optima in the objective function.

The latter family of neural networks comprises the self-organizing feature maps (SOM [12]) and its incremental variants. SOM is inherently a feature mapper with fixed topology (which is also its limit). Its variants have no topology (neural gas, NG [13]) or a variable topology and pave the way to pure incremental networks like growing neural gas (GNG, [14]). These networks, in conjunction with the Competitive Hebbian Rule (CHR, [15]), create a graph representing the manifold, which is the first step for most DR techniques. NG plus CHR is called Topology representing network (TRN, [16]). The approach is called TRNMap [17] if the DR technique is a multidimensional scaling (MDS); it is called RBF-NDR [18] if the projection is modeled by an RBF with an error function based on geodesic and Euclidean distances: in both cases, the projection follows the graph estimation, which results in the impossibility to track changes in real time. If the graph is computed by GNG, then the DR can be computed by OVI-NG [19], if Euclidean distances are used, and GNLG-NG [20] if geodesic distances replace Euclidean distances. However, from the point of view of real time applications, only the former is interesting, because it estimates, in the same time, the graph updating and its projection.

For data drawn from a nonstationary distribution (nonstationary data stream), as it is the case for fault and pre-fault diagnosis and system modeling, the above-cited techniques basically fail. For instance, the methods based on geodesic distances always require a connected graph. If the distribution changes abruptly (jump), they cannot track anymore. Apart the linear techniques, like PCA, which, for their speed, can be adapted to this problem, but are forcedly approximations in case of nonlinearities, only the variant of SOM, called DSOM [21], can be used. It exploits 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. Unfortunately, it is a forgetting network, in the sense that it forgets the past information and only tracks the last changes. This is a serious problem, especially in case the past inputs would carry useful information. There are also neural networks, like SOINN and its variants [22], which record the whole life of the process to be modelled (life-long learning), but are not able to project the information into a lower dimensional space. Nonetheless, they can be considered as a preprocessing clustering stage before the dimensionality reduction step. As a consequence, DR tools can be chosen according to the interest in either a network tailored only on the last data (forgetting network) or the whole (life-long) story, if, for instance, the data can repeat in the future. However, this preprocessing step is outside the scope of this paper, which is centered on the dimensionality reduction step. The same observation can be repeated for the data stream clustering methods [23]. 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 [24]) of the Growing Neural Gas neural network, hierarchical stream methods, like

BIRCH [25] and ClusTree [26], partitioning stream methods, like CluStream [27], and density-based stream methods, like DenStream [28] and SOStream [29], which is inspired by SOM. No technique takes the dimensionality reduction step into account, which is necessary in case of high dimensional streams. Only in [30] this problem is considered by applying a variant of the deep autoencoder, called the Variational Autoencoder (VAE [31]) for DR. There is a significant improvement in the clustering accuracy of high dimensional datasets, but, not all clustering algorithms benefit in the same way from DR. Additionally, regardless of the clustering algorithm, no relevant improvement in the purity of the clusters can be obtained. However, the theory underlying this DR technique is completely different from the classical techniques and is not taken into account in this paper.

Recently, an ad hoc architecture, somewhat in the GNG based category, has been proposed (onCCA, [32]), which addresses this problem by using an incremental quantization synchronously with a fast projection based on the Curvilinear Component Analysis (CCA, [33], [34]). This neural network requires an initial architecture provided by a fast offline CCA.

The growing CCA (GCCA, [35], [36]) neural network is an improved version of onCCA, which, by using the new idea of seed, does not need an initial CCA architecture. It also uses the principle of bridges in order to detect changes in the data stream.

This paper is an extensive description and analysis of GCCA. After the presentation of the traditional (offline) CCA in Sec. II, Sec. III presents some online algorithms such as OVI-NG and DSOM. Sec. IV introduces the new algorithm and discusses both its basic ideas and the influence of its user-dependent parameters. Sec. V shows the results of a few simulations on artificial problems and a real application. Sec. VI yields the conclusions.

2. The curvilinear component analysis

One of the most important non-linear techniques for dimensionality reduction is the Curvilinear Component Analysis (CCA, [33], [34]), which is a non-convex technique based on weighted distances. It derives from the Sammon mapping [1], but improves it because of its properties of unfolding data and extrapolation. CCA is a self-organizing neural network (see Fig.1), which performs the quantization of a data training set (input space, say X) for estimating the corresponding non-linear projection into a lower dimensional space (latent space, say Y). Two weights are attached to each neuron. The first one has the dimensionality of the X space and is here called X-weight: it quantizes the input data. The second one, here called Y-weight, is placed in the latent space and represents the projection of the X-weight. In a sense, each neuron can be considered as a correspondence between a vector and its projection. The input vector quantization can be performed in several ways, by using, for instance, classical neural unsupervised techniques. The CCA projection, which is the core of the algorithm, works as follows. For each pair of different weight vectors in the X space (data space), a between-point distance D_{ij}, calculated as $D_{ij} = \|x_i - x_j\|$. The objective is to constraint the distance L_{ij} of the associated Y-weights in the latent space, computed as $L_{ij} = \|y_i - y_j\|$, to be equal to 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 metric function F_λ, which penalizes the long distances but preserves local topology, by using a user-dependent parameter λ. In its simplest form, it is given by:

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

That is a step function for constraining only the under threshold between-point distances L_{ij}. A smoother way of favoring short distances stems from the use of the *right Bregman divergence* [34]:

$$F_\lambda(L_{ij}) = \ell^{-\frac{L_{ij}}{\lambda}} \tag{2}$$

whose asymmetry yields a better projection.

For each pair *i, j* of *N* neurons, the CCA error function is given by:

$$E_{CCA} = \frac{1}{2}\sum_{i=1}^{N} E_{CCA}^{i} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(D_{ij} - L_{ij}\right)^2 F_\lambda\left(L_{ij}\right) \tag{3}$$

This error allows to unfold strongly non-linear or closed structures. Defining as $\mathbf{y}_j$ the weight of the *j*-th projecting neuron, the stochastic gradient algorithm for minimizing (3) follows:

$$\mathbf{y}_j(t+1) = \mathbf{y}_j(t) + \alpha\left(D_{ij} - L_{ij}\right)F_\lambda\left(L_{ij}\right)\frac{\mathbf{y}_j(t) - \mathbf{y}_i(t)}{L_{ij}} \tag{4}$$

where $\boldsymbol{\alpha}$ is the learning rate.

The computational complexity of CCA is *O (N)*. CCA has good extrapolation (projection of points outside the input data domain). As pointed out in [37], the CCA projection has also a high trustworthiness.

However, there are some limits. For instance, although the accuracy of the projected topology in the latent space does not change, the projected map is not positionally invariant with regard to each new CCA execution. Indeed, it changes because it is only constrained by distance preservation. An invariant CCA is presented in [38].

3. Some Online Algorithms

As a comparison with GCCA, the OVI-NG and DSOM methods are here sketched. The former has some similarities with GCCA, but is not suitable for a nonstationary environment. The latter is simpler, but has no temporal memory.

3.1 OVI-NG

By defining a neural gas quantization equipped by the Competitive Hebbian Rule (CHR) for the creation of connections, the online visualization neural gas (OVI-NG, [19]) is a nonlinear projection method which combines TRN with the CCA projection rule (4). Quantization and projection are performed simultaneously (the authors define it as *on line* technique). The quantization is not incremental: it requires to define in advance the number of neurons (units) and the corresponding initial conditions. The CCA rule (4) uses the right Bregman divergence (2) and is tailored to offline problems: indeed, each parameter of the network is time-dependent and follows a decreasing exponential law in order to converge to a solution. As a consequence, twelve user-dependent parameters are required: the initial and final values of the lifetime of the connections, the NG learning rate, the neighborhood widths in input and output space and the CCA learning rate; the number of neurons; the maximum time for the temporal schedules.

This technique is not well suited to non-stationary data, because of its dependence on time (decreasing exponentially). Also, the predefined architecture does not allow to track the underlying distribution. However, by using constant parameters, it can mimic the forgetting behavior of DSOM. In this case, it still requires the initial conditions for the weights and six parameters. Nevertheless, OVI-NG is here used for comparison because it uses the same formula (4) as GCCA (but using the right Bregman divergence) and has some points in common with it.

As a consequence not to be incremental, there is no neuron pruning; indeed, CHL is only used for visualization purposes.

In [19] it is claimed OVI-NG can visualize discontinuities in the map thanks to CHR (under the assumption of trustworthiness of the network), by showing only the connections among non-neighboring units. However, CHR is not able to track non-stationarities, because it is not time directional (time anisotropy), as it will be clear in the analysis of GCCA bridges.

3.2 The Dynamic Self-Organizing Maps

Self-organizing maps (SOM) are composed of neurons equipped with pairs of weights, like CCA. The X-weights quantize the X-space, while the Y-weights are constrained to a fixed topology (predefined grid). The quantization law is given by:

$$x_i(t+1) = x_i(t) + \varepsilon(t)h_\sigma(t,i,i_{win})(\xi - x_i(t)) \tag{5}$$

where ξ is the input vector in X-space and i_{win} is the index of the closest neuron to the input (given a metric); the function $\varepsilon(t)$ is called learning rate and, in general, follows a decreasing law; h_σ is the neighborhood function and depends on the norm of the distance between the Y-weights of the neuron i and the winner *win*; it is often expressed as a Gaussian law:

$$h_\sigma(t,i,i_{win}) = \ell^{-\frac{\left\|y_i(t) - y_{i_{win}}(t)\right\|^2}{2\sigma(t)^2}} \tag{6}$$

where $\sigma(t)$ is a geometrically decreasing law (for tuning the quantization from coarse to fine).

This version of SOM is not able to track nonstationary data; indeed, when $\varepsilon(t)$ and $\sigma(t)$ are too low, the map does not

evolve anymore. A possible solution, which is considered in this paper, is the use of constant laws. However, it loses the possibility of fine-tuning the quantization because of this lack of flexibility.

The only good variant of SOM for solving this problem is given by the dynamic SOM (DSOM, [21]). The learning law (5) is changed as:

$$x_i(t+1) = x_i(t) + \varepsilon \| \xi - x_i(t) \| h_n(i, i_{win}, \xi)(\xi - x_i(t)) \tag{7}$$

where

$$h_n(i, i_{win}, \xi) = \ell^{-\frac{1}{\eta^2} \frac{\| y_i(t) - y_{i_{win}}(t) \|^2}{\| \xi - x_{i_{win}}(t) \|^2}} \tag{8}$$

being ε and η the constant learning rate and elasticity (or plasticity) parameter, respectively. If $\xi = x_{i_{win}}$, then $h_n(i, i_{win}, \xi) = 0$ and DSOM works like having time-invariant, but *local dependent* parameters. If an X-vector is close enough to data, there is no need for others to learn anything (the winner can represent the data); if there is no X-vector close enough to the data, any X-weight learns the data according to its own distance to it. The first fact prevents DSOM from fitting the *magnification law* (the quantization does not capture the data density): what is actually mapped by DSOM is the structure or *support* of the distribution rather than the density. The second one implies new data (from a changing environment) *attract* all X-weights. As a consequence, their positions (in input and output space) change and represent only the new information (*memoryless* network). Indeed, the goal of DSOM is only the representation of the new environment. It tracks each successive distribution with a short transient error correlated to the distribution change. Both *constant* SOM and DSOM require only two parameters. However, their convergence and stability are not proved (the same is true for SOM). In online learning, DSOM stability seems unachievable.

4 The GCCA Description

The growing CCA (GCCA) is a neural network whose number of neurons is determined by the quantization of the input space. Each neuron has two weights: the first in the data space (X-weight) is used for representing the input distribution, the second in the latent space (Y-weight) yields the corresponding projection. Each neuron is equipped with a local threshold which represents its receptive field in data space. It is estimated as the distance in X-space between the neuron and its farthest neighbor (neighbors are defined by the graph) and is used for determining the novelty of input data. Neurons are connected by links which define the manifold topology. The original concepts are the idea of seed and bridge. By considering that GCCA is an incremental network, it requires a number of initial neurons to start. As most neural networks of this kind, GCCA starts with a pair of neurons whose weights are, in general, random, which is here called seed. The concept of seed is also used in order to represent a change in the distribution, in the sense that these two neurons colonize the non-stationary inputs, i.e. they are the first neurons representing the change in data. Seeds are created by the neuron doubling. The bridge is a qualitatively different link, which indicates a non-stationarity of the input. Hence, there are two types of links: edges, created by CHL, and bridges. GCCA is incremental both in the sense that it can increase or decrease (pruning by age) the number of neurons and that the quantization and the projection work simultaneously. The learning rule is the soft competitive learning (SCL, [32]) except in neuron doubling, which requires the hard competitive learning (HCL, [32]). The projection is based on (4), which, because of the choice of (1), implies the idea of λ-neurons.

4.1 The Algorithm

The initial structure of GCCA is a seed, i.e. a pair of neurons connected by a link, which only in this case is an edge. The X-weights are random. However, a good choice is the use of two randomly drawn inputs. The associated Y-weights can be chosen randomly, but it is better that one projection is the zero vector, for normalization purposes.

The basic iteration, represented in the flowchart of Fig. 2, starts at the presentation of a new data, say $x_0 \in X$.

All neurons are sorted according to the Euclidean distances between x_0 and their X-weights. The neuron with the shortest distance (d_1) is the first winner w_1, the neuron with the second shortest distance is the second winner w_2 and so on. If d_1 is higher than the scalar threshold of the neuron w_1 (*novelty test*), a new neuron is created. Otherwise, there is a weight adaptation and a linking phase.

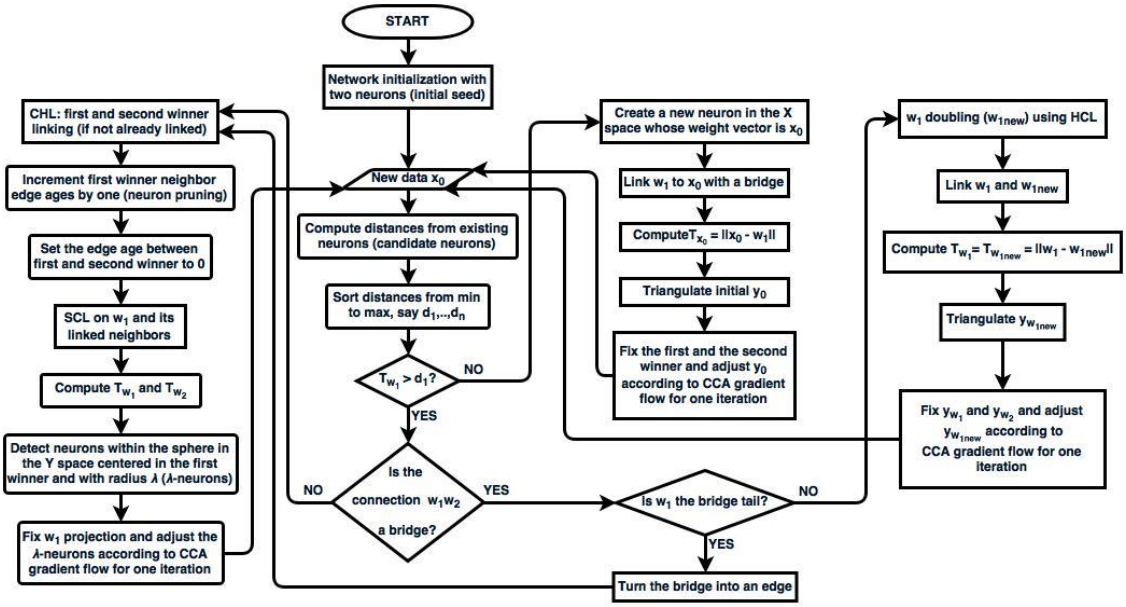

Fig. 2 The GCCA flowchart

Neuron creation. A new neuron is created whose X-weight vector is given by x_0. The winner w_1 and the new neuron are linked by a bridge (this link does not respect CHL). The new neuron threshold is set equal to d_1. The associated projection (Y-weight) in latent space requires two steps:

1. Determination of the initial values of the projection (y_0): a *triangulation* (see Appendix) inspired by [39] is used, in which the winner and second winner projections are the centers of two circles (in the first two dimensions of the latent space), whose radii are the distances in data space from the input data, respectively. There are two intersections and the initial two components are chosen as the farthest from the third winner projection. If the latent space is more than two-dimensional, the other components are chosen randomly.
2. One or several CCA iterations (4) in which the first and second winner projections are considered as fixed, in order to estimate the new y_0 (*extrapolation*).

Adaptation, linking and doubling. If a new neuron is not created, it is checked if the winner, whose X-weight is x_{-1}, and the second winner, whose X-weight is x_{-2}, are connected by a bridge.

1. If there is no bridge:
   a. these two neurons are linked by an edge (whose age is set to zero) and the same age procedure as in [13] is used as follows. The age of all other links emanating from 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, X-weights are adapted by using SCL [14]: x_{-1} and its direct topological neighbors (i.e. only those connected by one edge) are moved towards x_0 by fractions α_1 and α_n, respectively, of the total distance according to:

$$\Delta x_{-i} = \alpha_1(x_0 - x_{-i}) \qquad \text{i = 1} \qquad (9a)$$

$$\Delta x_{-i} = \alpha_n(x_0 - x_{-i}) \qquad \text{otherwise} \qquad (9b)$$

   and the thresholds of the winner and second winner are recomputed.

   b. The neurons whose Y-weights are within the sphere of radius λ centered in the first winner are determined, say λ-*neurons* (*topological constraint*). One or several CCA iterations (4), in which the first winner projection is fixed, are done for estimating the new projections of the λ-neurons (*interpolation*).
2. If it is a bridge, it is checked if the winner is the bridge tail; in this case, the bridge becomes an edge and step 1 is done. Otherwise, a seed is created by means of the neuron doubling (see Fig.3) as follows:

a. A virtual adaptation of the X-weight of the winner is estimated by HCL (only (9a) is used) and considered as the X-weight of a new neuron (doubling).

b. The winner and the new neuron are linked (age set to zero) and their thresholds are set equal to their Euclidean distance.

c. The initial projection of the new neuron (Y-weight) is estimated by the same *triangulation* as before.

d. One or several CCA iterations (4) are performed. The projections of the two neurons of the bridge (i.e. the first and the second winner) are considered as fixed, in order to estimate the final projection of the new neuron (*extrapolation*).

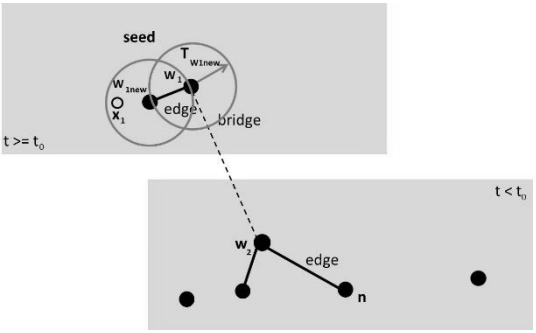

Fig. 3 Neuron Doubling

4.2 Choice of Parameters

The algorithm requires very few user-dependent parameters, in order to suit it to the problem at hand. They can be divided in two groups: one group for the CCA projection and the other one for the soft competitive learning (SCL). The CCA projection requires the following parameters:

- α learning rate: it influences the Y-projection through (4). A low value implies a more rigid displacement of the projected points. On the contrary, a high value has to be preferred in case of a fast changing data distribution, which can be automatically detected by introducing some measure of the non-stationarity (e.g. rate of change of the density and length of bridges). If the non-stationarity decreases, the parameter can be lowered to improve the accuracy of the projection.
- λ : it is the most important parameter of the network; it depends on the data manifold geometry. If the manifold is linear, λ can be set to ∞, because all distances in the X-space can be respected in the Y-space. On the contrary, a strong non-linearity implies a low value of λ, which means that distances can be preserved only locally.
  The selection of this parameter is also crucial because a too small value could imply a collection of local projections without any coordination. The accurate setting of λ is the way GCCA creates its global projection. As a consequence of all these problems, the original CCA requires its manual setting by visual inspection of a particular diagram, named *dy-dx*, which plots the pairs of corresponding distances in X- and Y-space.
  Another possible solution is the use of multiple λ's, one for each neuron: in this way λ can be made local and its tuning can be automatized by using the pairs of corresponding distances in X- and Y-space which express the projection accuracy. These values can also be considered as an alternative representation of the data manifold local curvature.

A last consideration on the CCA projection: the original CCA projection (4) is applied for several iterations (until convergence). This is possible because data are stationary and the technique is batch. On the contrary, GCCA is driven by changing data, which forbid this use of (4). As a consequence, (4) is iterated each time the neuron (which represents the projection) is taken into account. Indeed, when the neuron is created (new data or doubled neuron), (4) is computed for the first time (in extrapolation mode). All the subsequent iterations are performed in interpolation mode each time the associated neuron is labeled as λ-neuron.

The soft competitive learning and pruning of GCCA require the following parameters:

- α_l and α_n : they are used in the same way as for other neural networks, like GNG. In particular, $\alpha_l >> \alpha_n$. The case $\alpha_n = 0$ corresponds to Hard Competitive Learning (HCL). These quantities can be set constant, as in DSOM, in order to react to changes in the input distribution. This is the choice for this paper. However, it is

possible to apply a local α_1 parameter to each neuron, endowed with a decreasing exponential law. If, from one side this allows a better refinement of the quantization, on the other side, this rigidity can be avoided in presence of non-stationarity, which is detected by means of the bridges. For each new neuron (unlike DSOM, GCCA is incremental), the associated α_1 is set to the initial value. Even this initial value can be set equal to the length of the corresponding bridge. Indeed, for big jumps in the distribution, α_1 is given a larger value, which implies a more flexible network.

- age_{max} : it is a threshold for the pruning of the network. A low value implies a more flexible network, capable of better tracking the input distribution. It can be computed automatically by estimating the rate of change of the number of bridges. If this rate is higher, there is more novelty in data and the threshold is decreased in order to better react to the change.

Resuming, only four user-dependent parameters are required in case of HCL. They are five for SCL. However, the above-cited considerations show the possibility of a completely automatic GCCA.

4.3 Analysis of Bridges

Bridges are fundamental in tracking nonstationary data. They are links between a neuron and a new data (new neuron). As a consequence, they point to the change in data. They have two basic characteristics: the length and the density. A long bridge, whose new neuron has doubled, represents an effective change in the input distribution; instead, if the new neuron has no edges, it represents an outlier. The density yields further insight in the time-varying distribution:

1. In case of abrupt change in the input distribution (jump), there are a few long bridges.
2. In case of smooth displacement of data, the density of bridges is proportional to the displacement speed of the distribution.
3. In case of very slow displacement, only the border (frontier of the distribution domain) neurons win the competition and move in average in the direction of the displacement. The other neurons are static. Very slow displacement implies no bridges. Bridges appear only if the learning rate of SCL is not constant, but depends inversely on the number of wins; however, this rate must be reset after a certain number of iterations, in order to avoid the braking effect on the border neurons (effect similar to DSOM).

The geometry of bridges can be deepened, but it is outside the scope of the paper.

5 Examples

The behavior of the GCCA neural network is now illustrated both for artificially examples and, in the next subsection, in a real world experiment, with regard to the above described DSOM and OVI-NG, which are the only non-linear algorithms for non-stationary problems.

The first example is a simple illustration of the basic difference between DSOM and GCCA. It deals with a data uniform distribution whose domain jumps three times (from NW to NE, then from NE to SW and, finally, from SW to SE).

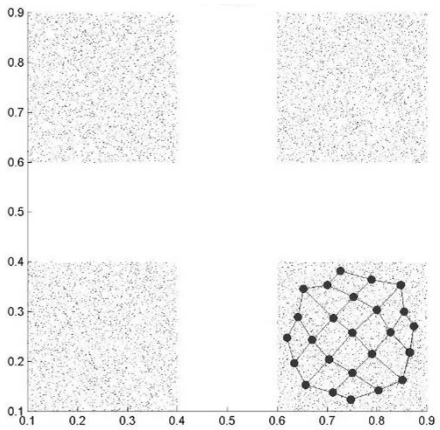

Fig. 4. Four squares: DSOM case. Data are represented by points, neurons by full-circles and edges by segments.

Fig. 4 shows the result for DSOM ($\varepsilon = 1.5$, $\eta = 5.5$). The neurons and edges only represent the last domain of the distribution. The previous information is completely lost. In this sense, the DSOM quantization is memoryless. The same phenomenon can be seen for other unsupervised neural networks, like SOM, when using constant parameters. DSOM detects the jump by means of the migration of its neurons towards the new domain, because it does not have additional resources because the number of neurons is constant (and fixed in advance).

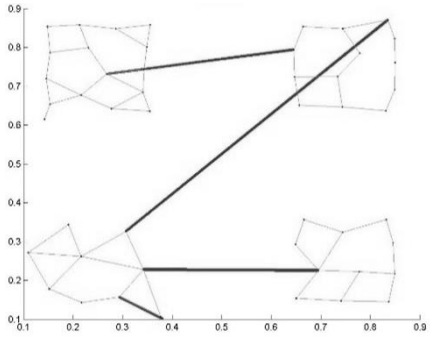

Fig. 5. Four squares: GCCA case. Edges are represented by thin segments, neurons by small points and bridges by thick segments.

Fig. 5 illustrates the behavior of GCCA on the same example. The parameters are set to: $\alpha = 0.005$, $\lambda = 2$, $\alpha_1 = 0.01$, $\alpha_n = 0.0001$, $age_{max} = 8$. Unlike DSOM, the resulting quantization memorizes the previous positions of the distribution

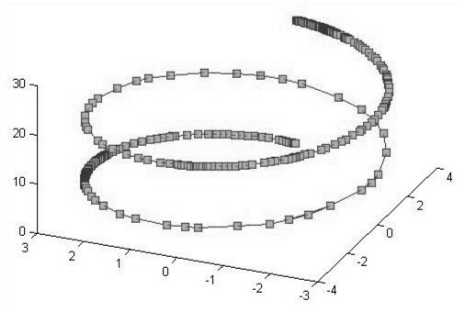

Fig.6. 3-D Spiral Distribution: GCCA vector quantization

(the grid deformations are due to the small number of data presentations before each jump). The advantage of being incremental gives the possibility of representing the different domains without need of defining the number of neurons in advance. Pruning only works in the current domain, because in the previous domains there are no more winners, which implies that the age of the corresponding edges does not change. This *freezes* the previous quantization. The presence of single bridges alerts for a jump (abrupt change) in the distribution. In case of a smoother change, more bridges would be created.

Fig. 6 shows a static unidimensional manifold representing a 3-D spiral distribution. The parameters of GCCA are set to: $\alpha = 0.001$, $\lambda = 20$, $\alpha_1 = 0.4$, $\alpha_n = 0.1$, $age_{max} = 2$. The projection results are shown in Fig. 7. It can be deduced that the quantization spans the input domain uniformly and the projection unfolds data correctly. This result shows GCCA has the same performance as CCA when the distribution is stationary. In this sense, we can consider GCCA as an extension of CCA to non-stationary distributions.

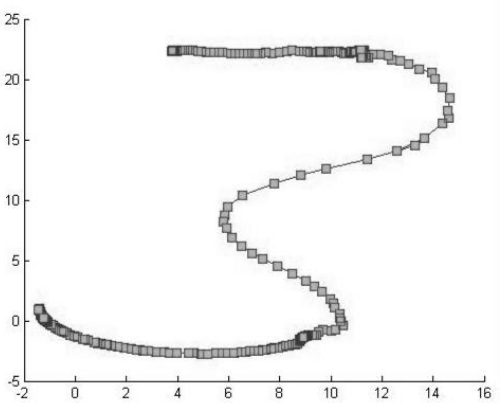

Fig.7. 3-D Spiral Distribution: 2-D GCCA projection

5.1 A. Fault Diagnostics Application

The OVI-NG (with constant parameters) and GCCA neural networks have been tested in the diagnosis of degradation and faults of the bearing of an asynchronous motor. Real data [40] are drawn from an experimental platform (PRONOSTIA, [41]) that enables accelerated degradation of bearings under constant and/or variable operating conditions, while gathering online health monitoring data (rotating speed, load force, temperature, vibration). The main objective of PRONOSTIA is to provide real experimental data that characterize the degradation of ball bearings along their whole operational life (until their total failure).

The dataset contains 2155 5-dimensional vectors whose components correspond to statistical features extracted by measurements drawn from four vibration transducers of the motor. In particular, this test deals with a nonstationary framework which evolves from the healthy state to a double fault occurring in the inner-race of a bearing and in the ball of another bearing.

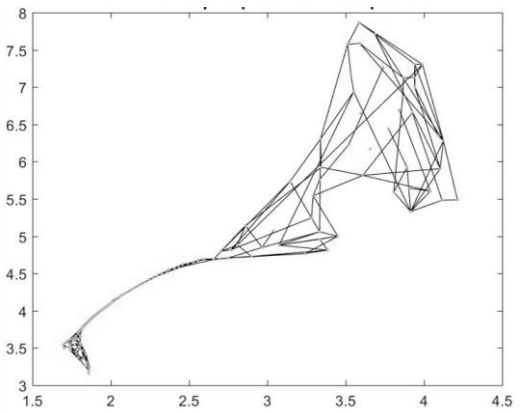

Fig.8. Bearing life tracking: OVI-NG projection

The best OVI-NG neural network is composed of 150 neurons; the CCA and NG learning rates are equal to 0.25; the neighborhood widths in the input and output spaces are set to 35 and 70, respectively. The projection is shown in Fig. 8.

The best GCCA neural network uses the following parameters: $\alpha = 0.01$, $\lambda = 1.6$, $\alpha_1 = 0.05$, $\alpha_n = 0.005$, $age_{max} = 2$. Fig. 9 illustrates the projection.

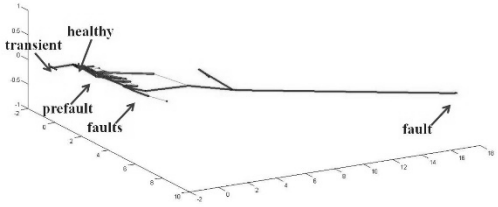

Fig. 9. Bearing life tracking: GCCA projection

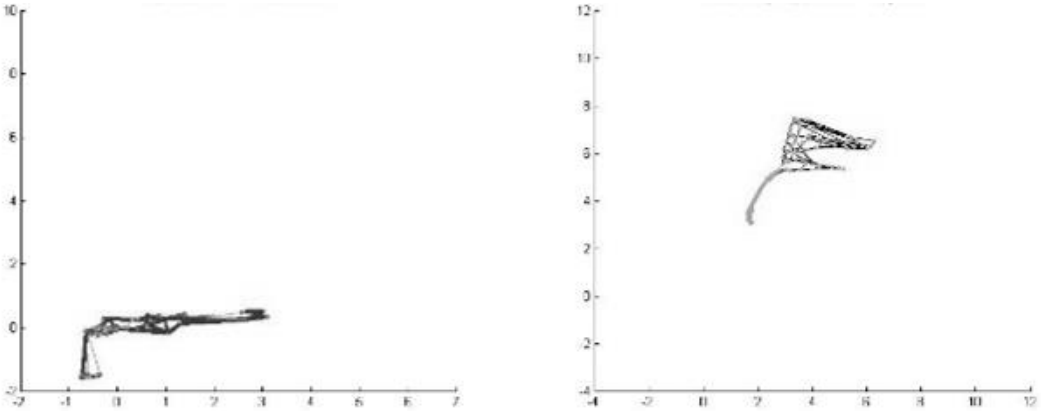

Fig. 10. Bearing life tracking: GCCA (left), OVI-NG (right) comparison at the fault onset

The GCCA learns the chain behavior and tracks it, by adapting in real time the data projection. Fig. 9 shows the motor lifecycle, from the initial transient phase, through the healthy state, towards, first, a pre-fault (characterized by an increasing bridge density), and, finally, the two faults which are clearly identified in the figure by the longer bridges.

As apparent in Figs. 8 and 9, the dataset represent the all history of the bearing. After a short transient, the bearing is in its own healthy state. Then, two possible faults happen. Both neural networks represent well the first two phases. However, they behave differently at the onset of the fault (see Fig. 10). OVI-NG starts to oscillate, while GCCA increases the number of bridges. This difference influences the final representation of these two networks: unlike OVI-NG, which only underlies oscillations, GCCA shows two branches associated to the two faults. The advantage of GCCA versus OVI-NG does not only rely on its representation properties, but also on the accuracy of the quantization and projection. Indeed, for assessing this property, the trustworthiness and continuity measures [42], [43] are used to compare the mapping quality.

A projection (map) onto an output space is said to be *trustworthy* if the set of k nearest neighbors of a point in the map are also close by in the original space. Let $U_k(i)$ be the set of data samples that are in the neighborhood of the *i-th* point in the map but not in the original space. The measure of trustworthiness of the visualization, M_1, is defined as:

$$M_1(k) = 1 - A(k)\sum_{i=1}^{N}\sum_{j\in U_k(i)}(r(x_i, x_j) - k) \tag{10}$$

where N is the total number of neurons, $A(k) = 2/(Nk \times (2N - 3k - 1))$, and $r(x_i, x_j)$ is the neuron ranking in input space.

A projection onto an output space is said to be *continuous* if the set of k closest neighbors of a point in the original space are also close by in the output space. Let $V_k(i)$ be the set of data samples that are in the neighborhood of the *i-th* point in the original space but not in the map. The measure of continuity of the visualization, M_2, is defined as:

$$M_2(k) = 1 - A(k)\sum_{i=1}^{N}\sum_{j\in V_k(i)}(s(y_i, y_j) - k) \tag{11}$$

where $s(y_i, y_j)$ is the neuron ranking in output space.

In order to track these indices in time, because of the non-stationary nature of the problem two plots for trustworthiness (Fig. 11) and continuity (Fig. 12) are given with regard not only to k, as usual, but also to time.

With regard to trustworthiness, the behavior of both networks is very accurate and similar until the onset of the first fault. However, after the fault onsets there is big loss in the projection quality of OVI-NG, which is recovered only after a certain number of iterations. This confirms the previous analysis about the advantage of the bridge representation of the fault with regard to the oscillating behavior of OVI-NG.

A similar analysis can be repeated for the case of continuity index, with the difference that in the healthy state the index for GCCA is better. Probable it is due to the incremental nature of GCCA.

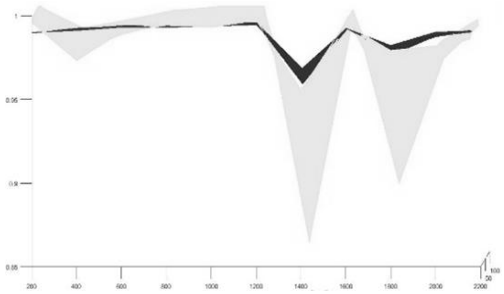

Fig.11. Trustworthiness as a function of k and time: OVI-NG (grey), GCCA (black)

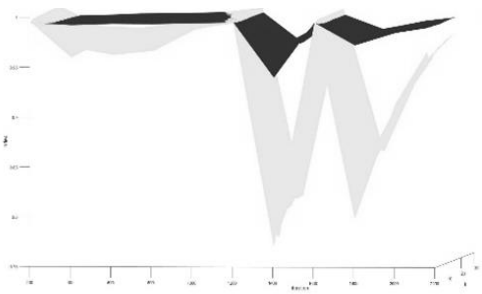

Fig.12. Continuity as a function of k and time: OVI-NG (grey), GCCA (black)

6 CONCLUSION

Only a few networks, in general linear, are able to track a non-stationary input distribution and to project it in a lower dimensional space. In this paper nonlinear techniques, like DSOM and OVI-NG have been taken into account, by choosing their parameters as time-constant.

The GCCA neural network is a novel incremental technique which has been devised as an extension of CCA in order to learn and project a time-varying manifold. The algorithm is based on three key ideas: the first is the seed, a pair of neurons which colonizes (start of the new vectorization) a change in the input distribution domain; the second is the bridge, which not only allows the visualization of data changes, but also can discriminate the outliers and yields the possibility (by its geometry and density) to infer more information about the non-stationarity; the third is the locality of the projection, given by the selection of the λ-*neurons* for the CCA iterations. The global coherence of the projection is obtained by modulating λ. It is basically dependent on very few user-dependent parameters, which can be automatically estimated by the algorithm itself.

Unlike nonlinear techniques with constant parameters (e.g. DSOM) which are memoryless, both OVI-NG and GCCA represent the whole story of the process at hand. However, the incremental property of GCCA allows a better representation because more neurons can be used. Indeed, OVI-NG has not been conceived for nonstationary problems, even if it derives from CCA and is considered as working in real time (online). In the examples, GCCA has shown a better performance in terms of trustworthiness and continuity than OVI-NG, but its main advantage is given by the presence of bridges, which allow a deep analysis of the non-stationarity.

Future work will deal with the implementation of an automatic GCCA (i.e., no user-dependent parameters needed), a deeper analysis of bridges and a minor change in the computation of the short distances for approximating the geodesic distances. Novel applications of GCCA are dealing with fault diagnosis of electrical machines (using current and voltage signals) and analysis of colorectal cancer by using DNA microarrays (it is a non-stationary problem because of the change in volume of the tumor).

REFERENCES

[1] L. Van der Maaten, E. Postma and H. Van der Herik, Dimensionality Reduction: a Comparative Review, TiCC TR 2009-005, Delft University of Technology, 2009.

[2] T.D. Sanger, Optimal Unsupervised Learning in a Single-Layer Neural Network, Neural Networks, vol. 2, pp. 459-473, 1989.

[3] K.I. Diamantaras and S.Y. Kung, Principal Component Neural Networks: Theory and Applications, Wiley and Sons, 1996.

[4] Weng J., Zhang Y. and Hwang W.S., Candid covariance-free incremental principal components analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25 (8), pp. 1034-1040, 2003.

[5] Kouropteva O., Okun O. and Pietikainen M., Incremental locally linear embedding, Pattern Recognition, vol. 38, pp. 1764-1767, 2005.

[6] Jia P., Yin J., Huang X. and Hu D., Incremental Laplacian Eigenmaps by preserving adjacent information between data points, Pattern Recognition Letters, vol. 30, pp. 1457-1463, 2009.

[7] Li H., Jiang H., Barrio R., Lia X., Cheng L. and Su F., Incremental manifold learning by spectral embedding methods, Pattern Recognition Letters, vol. 32, pp. 1447-1455, 2011.

[8] D. de Ridder and R. Duin, Sammon's mapping using neural networks: a comparison, Pattern Recognition Letters, vol. 18, pp. 1307-1316, 1997.

[9] Qiang X., Cheng G. and Li Z., A Survey of Some Classic Self-organizing Maps with Incremental Learning, 2nd Int. Conf. on Signal Processing Systems (ICSPS), pp. 804-809, 2010.

[10] J. Mao and A.K. Jain, Artificial neural networks for feature extraction and multivariate data projection, IEEE Transactions on Neural Networks, vol. 6 (2), pp. 296-317, 1995.

[11] G. E. Hinton and R. R. Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks, Science, vol. 313, pp. 504-507, 2006.

[12] Kohonen T, Schroeder MR, Huang TS, (eds). Self-Organizing Maps, 3rd edn. Secaucus, NJ, USA: Springer; 2001

[13] T. Martinetz and K. Schulten , A "neural gas" network learns topologies, Artificial Neural Networks, Elsevier, pp. 397–402, 1991.

[14] B. Fritzke, A Growing Neural Gas Network Learns Topologies, in Advances in Neural Information Processing System, 7, MIT Press, pp. 625-632, 1995.

[15] R. White, Competitive Hebbian Learning: Algorithm and Demonstations, Neural Networks, vol. 5, no. 2, pp. 261-275, 1992.

[16] T. Martinetz and K. Schulten , Topology representing networks, Neural Networks, vol. 7 (3), pp. 507-522, 1994.

[17] Vathy-Fogarassy A., Kiss A. and J. Abonyi, Topology Representing Network Map – A New Tool for Visualization of High-Dimensional Data, in Transactions on Computational Science I, Vol. 4750 of the series Lecture Notes in Computer Science pp 61-84, 2008.

[18] V. Tomenko, Online dimensionality reduction using competitive learning and Radial Basis Function network, Neural Networks, vol. 24, pp. 501-511, 2011.

[19] P. Estevez and C. Figueroa, Online data visualization using the neural gas network, Neural Networks, vol. 19, pp. 923-934, 2006.

[20] Estevez P., Chong A., Held C. and Perez C., Nonlinear projection using geodesic distances and the neural gas network, Lect. Notes Comput. Sci., 4131, pp. 464-473, 2006.

[21] N. P. Rougier and Y.Boniface, Dynamic Self-Organising Map, Neurocomputing, Elsevier, 74 (11), pp.1840-1847, 2011.

[22] F. Shen, O. Tomotaka and O. Hasegawa, An enhanced self-organizing incremental neural network for online unsupervised learning, *Neural Networks,* vol. 20, pages 893–903, 2007

[23] M. Ghesmoune, M. Lebbah and H. Azzag, State-of-the-art on clustering data streams, Big Data Analytics, 1:13; 2016.

[24] M. Ghesmoune, H. Azzag and M. Lebbah. G-stream: Growing neural gas over data stream. In Neural Information Processing-21st International Conference, ICONIP, Kuching, Malaysia, p.207-214, 2014.

[25] T. Zhang, R. Ramakrishnan and M. Livny. BIRCH: an efficient data clustering method for very large databases. In SIGMOD Conference, New York, ACM, p.103-114, 1996.

[26] P. Kranen, I. Assent, C. Baldauf and T. Seidl. The ClusTree indexing microclusters for anytime stream mining. Knowledge Inf. Syst., 29(2), p.249-272, 2011.

[27] C.C. Aggarwal, T.J. Watson, R. Ctr, J. Han, J. Wang and P.S. Yu. A framework for clustering evolving data streams . In VLDB. Berlin: VLDB Endowment, p.81-92, 2003.

[28] Cao F, Ester M, Qian W, Zhou A. Density-based clustering over an evolving data stream with noise. In: SDM. SIAM;, p. 328–39, 2006.

[29] Isaksson C, Dunham MH, Hahsler M. SOStream: Self organizing density-based clustering over data stream. In: MLDM. Berlin: Springer Berlin Heidelberg;., p. 264–78, 2012.

[30] A. Hontabat, and M. Rising. Clustering non-stationary data streams with online deep learning reduction. Bachelor's thesis, Department of Computer Science and Engineering, Chalmers University of Technology,, University of Gothenburg, Göteborg, Sweden 2016.

[31] D. Kingma and M. Welling. "Auto-Encoding Variational Bayes". Proceedings of the 2nd International Conference on Learning Representations. Banff, Canada, 2014.

[32] G. Cirrincione, J. Hérault and V. Randazzo, The on-line curvilinear component analysis (onCCA) for real-time data reduction, International Joint Conference on Neural Networks (IJCNN), pp. 157-165, 2015.

[33] P. Demartines and J. Hérault, Curvilinear Component Analysis: A Self-Organizing Neural Network for Nonlinear Mapping of Data Sets, IEEE Transaction on Neural Networks, vol. 8, no. 1, pp. 148-154, 1997.

[34] J. Sun, M. Crowe and C. Fyfe, Curvilinear Components Analysis and Bregman Divergences, in proceedings of the European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning (ESANN). pp.81-86, Bruges (Belgium), 2010.

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

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

[37] Venna, J., & Kaski, S. (2005). Local multidimensional scaling with controlled tradeoff between trustworthiness and continuity. In Proceedings of the workshop on self-organizing maps (pp. 695–702).

[38] G. Cirrincione, J. Ortega, H. Henao, M. Prieto and A. Espinosa, Bearing Faults Detection by a Novel Condition Monitoring Scheme based on Statistical-Time Features and Neural Networks, IEEE Transactions On Industrial Electronics, vol. 60, no. 8, pp. 3398-3407, 2012.

[39] Karbauskaitė, R., Dzemyda, G.: Multidimensional data projection algorithms saving calculations of distances, Information Technology and Control, vol.35, no.1, pp. 57-61, 2006.

[40] FEMTO Bearing Data Set, Nasa prognostic data repository, http://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository

[41] P. Nectoux, R. Gouriveau, K. Medjaher, E. Ramasso, B. Morello, N. Zerhouni, C. Varnier. PRONOSTIA: An Experimental Platform for Bearings Accelerated Life Test. IEEE International Conference on Prognostics and Health Management, Denver, CO, USA, 2012

[42] Kaski, S., Nikkilä, J., Oja, M., Venna, J., Törönen, P., & Castrén, E. (2003). Trustworthiness and metrics in visualizing similarity of gene expression. BMC Bioinformatics, 4, 48.

[43] Venna, J., & Kaski, S. (2005). Local multidimensional scaling with controlled tradeoff between trustworthiness and continuity. In Proceedings of the workshop on self-organizing maps (pp. 695–702).

APPENDIX - THE TRIANGULATION METHOD

Given two points $P_i(x_i, y_i)$ and $P_j(x_j, y_j)$, whose distance $d_{ij} = d$ is known. Find the point $P_k(x_k, y_k)$ whose position is forced to respect the two distances $d_{ik} = r_i$ and $d_{jk} = r_j$. Actually, $P_i(x_i, y_i)$ and $P_j(x_j, y_j)$ are associated to the first and second neighbors in the high dimensional space of the data whose projection $P_k(x_k, y_k)$ is sought.

Algorithm :

1. If $d < r_i + r_j$ (intersecting circles)

$$\delta = \frac{1}{4}\sqrt{(d + r_i + r_j)(d + r_i - r_j)(d - r_i + r_j)(-d + r_i + r_k)}$$

$$x_k^{\,1} = \frac{x_i + x_j}{2} + \frac{(x_j - x_i)(r_i^{\,2} - r_j^{\,2})}{2d^2} + 2\frac{y_i - y_j}{d^2}\delta$$

$$x_k^{\,2} = \frac{x_i + x_j}{2} + \frac{(x_j - x_i)(r_i^{\,2} - r_j^{\,2})}{2d^2} - 2\frac{y_i - y_j}{d^2}\delta$$

$$y_k^{\,1} = \frac{y_i + y_j}{2} + \frac{(y_j - y_i)(r_i^{\,2} - r_j^{\,2})}{2d^2} - 2\frac{x_i - x_j}{d^2}\delta$$

$$y_k^{\,2} = \frac{y_i + y_j}{2} + \frac{(y_j - y_i)(r_i^{\,2} - r_j^{\,2})}{2d^2} + 2\frac{x_i - x_j}{d^2}\delta$$

Choose either $P_k^{\,1}(x_k^{\,1}, y_k^{\,1})$ or $P_k^{\,2}(x_k^{\,2}, y_k^{\,2})$ using a third projected point, say $P_l(x_l, y_l)$, whose associated high dimensional vector is the third nearest neighbor to the associated high dimensional vector of P_k .

$$P_k = P_k^{\,s} \text{ where } s = \arg\min_r \left\| P_k^{\,r} - P_l \right\| \text{ interpolation}$$

$$P_k = P_k^{\,s} \text{ where } s = \arg\max_r \left\| P_k^{\,r} - P_l \right\| \text{ extrapolation}$$

2. If $r_i > d + r_j$

$$\delta = \frac{r_i - r_j - d}{2}$$

a. If $x_i \neq x_j$

$$x_k = x_i + \frac{(x_j - x_i)(r_i - \delta)}{d}$$

$$y_k = y_i + \frac{(x_k - x_i)(y_j - y_i)}{x_j - x_i}$$

b. If $x_i = x_j$ and $y_i > y_j$

$$x_k = x_i$$

$$y_k = y_j - r_j - \delta$$

c. If $x_i = x_j$ and $y_i < y_j$

$$x_k = x_i$$

$$y_k = y_j + r_j + \delta$$

d. If $x_i = x_j$ and $y_i = y_j$ (concentric circles)

Consider $P_l(x_l, y_l)$, whose associated high dimensional vector is the third nearest neighbor to the associated high dimensional vector of P_k .

$$\rho = r_j + \frac{r_i - r_j}{2}$$

$$y_k{}^1 = y_i + \sqrt{\frac{\rho^2 (y_l - y_i)^2}{(y_l - y_i)^2 + (x_l - x_i)^2}}$$

$$y_k{}^2 = y_i - \sqrt{\frac{\rho^2 (y_l - y_i)^2}{(y_l - y_i)^2 + (x_l - x_i)^2}}$$

$$x_k{}^1 = x_i + \frac{(x_l - x_i)(y_k{}^1 - y_i)}{y_l - y_i}$$

$$x_k{}^2 = x_i + \frac{(x_l - x_i)(y_k{}^2 - y_i)}{y_l - y_i}$$

Choose either $P_k{}^1(x_k{}^1, y_k{}^1)$ or $P_k{}^2(x_k{}^2, y_k{}^2)$ as the closest (interpolation) or the farthest (extrapolation) to the third projected point, say $P_l(x_l, y_l)$, if $y_i = y_j \neq y_l$.

e. If $x_i = x_j$ and $y_i = y_j = y_l$ and $x_l \geq x_i$

$$y_k = y_i$$

$$x_k = x_j + r_j + \frac{r_i - r_j}{2}$$

f. If $x_i = x_j$ and $y_i = y_j = y_l$ and $x_l < x_i$

$$y_k = y_i$$

$$x_k = x_j - r_j - \frac{r_i - r_j}{2}$$

3. If $r_j > d + r_i$
   replace *i* with *j* and vice versa; use the eqs. of step 2.

4. If $d \geq r_i + r_j$ (non-intersecting or tangent circles)

a. If $y_i \neq y_j$

$$\rho = \frac{d + r_i - r_j}{2}$$

$$y_k{}^1 = y_i + \sqrt{\frac{\rho^2 (y_j - y_i)^2}{(y_j - y_i)^2 + (x_j - x_i)^2}}$$

$$y_k{}^2 = y_i - \sqrt{\frac{\rho^2 (y_j - y_i)^2}{(y_j - y_i)^2 + (x_j - x_i)^2}}$$

$$x_k{}^1 = x_i + \frac{(x_j - x_i)(y_k{}^1 - y_i)}{y_j - y_i}$$

$$x_k{}^2 = x_i + \frac{(x_j - x_i)(y_k{}^2 - y_i)}{y_j - y_i}$$

Choose either $P_k{}^1(x_k{}^1, y_k{}^1)$ or $P_k{}^2(x_k{}^2, y_k{}^2)$ as the closest (interpolation) or the farthest (extrapolation) to the second projected point, say $P_j(x_j, y_j)$.

.

b. If $y_i = y_j$ and $x_i < x_j$

$$y_k = y_i$$

$$x_k = x_i + r_i + \frac{d - r_i - r_j}{2}$$

c. If $y_i = y_j$ and $x_i > x_j$

$$y_k = y_i$$

$$x_k = x_j + r_j + \frac{d - r_i - r_j}{2}$$

Dr. Giansalvo Cirrincione is Associate Professor in Control Systems at the University of Picardie Jules Verne (UPJV), Amiens (France) and member of the LTI (Laboratory of Novel Technologies) in the same University. He graduated at the Politecnico di Torino and then got his PhD at the Institut Polytechnique de Grenoble in France in 1998. In 1999, he was a post-doc for a year at the Catholic University of Leuven in Belgium at the Department of Electrical Engineering (SISTA). He is Adjunct Associate Professor at the University of the South Pacific and has been visiting professor in several countries (Italy, United Kingdom, Fiji, Turkey, Spain). He is author of more than 100 publications, 33 of which on international peer-reviewed ISI journals, and one book about linear regression. His current research interests include: neural networks, orthogonal regression, data analysis, computer vision, pattern recognition, system identification, diagnosis of nonlinear system, and electrical drives. He is a Senior Member IEEE.

Vincenzo Randazzo is Ph.D. student at the Politecnico of Turin on the topic "Offline and online neural networks applied to pattern recognition of big data and IoT (Internet of Things) sensors data". He graduated at Università degli Studi di Palermo in Computer Engineering. His works have been accepted at IJCNN2015 and WIRN2016, WIRN2017. His current research interests include: neural networks, data analysis, pattern recognition, diagnosis of nonlinear system, electrical drives, IoT, medical and biomedical applications, in particular, electrocardiogram (ECG). He is the treasurer of the IEEE student branch of the Politecnico di Torino.

Prof. Eros G. Pasero is Professor of Electronic System Engineering in the Department of Electronics and Telecommunications and the head of the Neuronica Lab at the Politecnico of Turin. He is now the President of SIREN, the Italian Society for Neural Networks and the General Chairman of WIRN2017, the international Italian workshop for artificial neural networks. Prof. Pasero interests lie in Artificial Neural Networks and Electronic Sensors. Hardware neurons and synapses are studied for neuromorphic approaches; neural software applications are applied to real life proof of concepts. Innovative wired and wireless sensors are also developed for biomedical, environmental and automotive applications. Data coming from sensors are post processed by means of artificial neural networks. He is a IEEE Member.