Growing Curvilinear Component Analysis (GCCA) for Dimensionality Reduction of Nonstationary Data

Giansalvo Cirrincione[1], Vincenzo Randazzo[2] and Eros Pasero[2]

[1]University of Picardie Jules Verne, Lab. LTI, Amiens, France
exin@u-picardie.fr
[2]Politecnico di Torino, DET, Turin, Italy
{vincenzo.randazzo,eros.pasero}@polito.it

Abstract. 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). Their incremental variants do no work properly. The onCCA neural network addresses this problem; it is incremental and performs simultaneously the data quantization and projection by using the Curvilinear Component Analysis (CCA), a distance-preserving reduction technique. However, onCCA requires an initial architecture, provided by a small offline CCA. This paper presents a variant of onCCA, called growing CCA (GCCA), which has a self-organized incremental architecture adapting to the nonstationary data distribution. This is achieved by introducing the ideas of "seeds", pairs of neurons which colonize the input domain, and "bridge", a different kind of edge in the manifold graph, which signal the data nonstationarity. Some examples from artificial problems and a real application are given.

Keywords: dimensionality reduction; curvilinear component analysis; online algorithm; neural network; vector quantization; projection; seed; bridge.

1 Introduction

Data mining is ever increasingly facing the extraction of meaningful information from big data (e.g. from internet), which are 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 helps 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, but also tracking non-stationary

data distributions (e.g. time-varying data manifolds). This can be used, for example, for 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, 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 for 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] and the incremental PCA (candid covariance-free CCIPCA [3]). 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, [4]) 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 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 by either creating ad hoc architectures and error functions or using self-organizing maps (SOM) and variants. 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, [5]). These techniques require the stationarity of their training set. The latter family of neural networks comprises the self-organizing feature maps (SOM) 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 [6]) or a variable topology and pave the way to pure incremental networks like growing neural gas (GNG, [7]). These networks, in conjunction with the Competitive Hebbian Learning (CHL, [8]), create a graph representing the manifold, which is the first step for most DR techniques. NG plus CHL is called Topology representing network (TRN, [9]). The approach is called TRNMap [10] if the DR technique is a multidimensional scaling (MDS); here 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 [11], if Euclidean distances are used, and GNLG-NG [12] 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, as it is the case for fault and prefault 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. Recently, an ad hoc architecture has been proposed (onCCA, [13]), which tracks nonstationarity by using an incremental quantization synchronously with a fast projection based on the Curvilinear Component Analysis (CCA, [22]). It requires an initial architecture provided by a fast offline CCA.

The purpose of this paper is the presentation of an improved version of onCCA, here called growing CCA (GCCA), 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. After the presentation of the traditional (offline) CCA in Sec. 2, Sec. 3 introduces the new algorithm and discusses both its basic ideas and the influence of its user-dependent parameters. Sec. 4 shows the results of a few simulations on artificial and real problems. Sec. 5 presents the conclusions.

2 The curvilinear component analysis

One of the most important nonlinear techniques for dimensionality reduction is the Curvilinear Component Analysis (CCA, [14]), 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. It 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 distance function, which, in its simplest form, is the following:

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

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

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

Defining as $\boldsymbol{y}(j)$ the weight of the j-th projecting neuron, the stochastic gradient algorithm for minimizing (2) follows:

$$\mathbf{y}(j) \leftarrow \mathbf{y}(j) + \alpha(D_{ij} - L_{ij}) F_{\lambda}(L_{ij}) \frac{\mathbf{y}(j) - \mathbf{y}(i)}{L_{ij}} \tag{3}$$

where α is the learning rate.

3 The GROWING CCA (GCCA)

The growing CCA 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. The neurons are connected by links which define the manifold topology. The original concepts are the idea of seed and bridge. The seed is a pair of neurons, which (except in the network initialization) colonize the nonstationary input distribution, in the sense that they are the first neurons representing the change in data. Seeds are created by the neuron doubling explained in Fig. 1. The bridge is a qualitatively different link, which indicates a nonstationarity of the input. Hence, there are two types of links: edges, created by CHL, and bridges. Each neuron is equipped with a 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. GCCA is incremental both in the sense that it can increase or decrease (pruning by age) the number of neurons and the quantization and the projection work simultaneously. The learning rule is the soft competitive learning (SCL, [13]) except in neuron doubling, which requires the hard competitive learning (HCL, [13]). The projection is based on (3), which, as a consequence of the choice of (1), implies the idea of λ-neurons.

3.1 The Algorithm

The initial structure of GCCA is a seed, i.e. a pair of neurons. 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. 1, 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 winner. If its distance is higher than the scalar threshold of the neuron (novelty test), a new neuron is created. Otherwise, there is a weight adaptation and a linking phase.

Neuron creation. The X-weight vector is given by x_0. The winner and the new neuron are linked by a bridge (this link does not respect CHL). The new neuron threshold is 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* inspired by [15] 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 (3) 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.

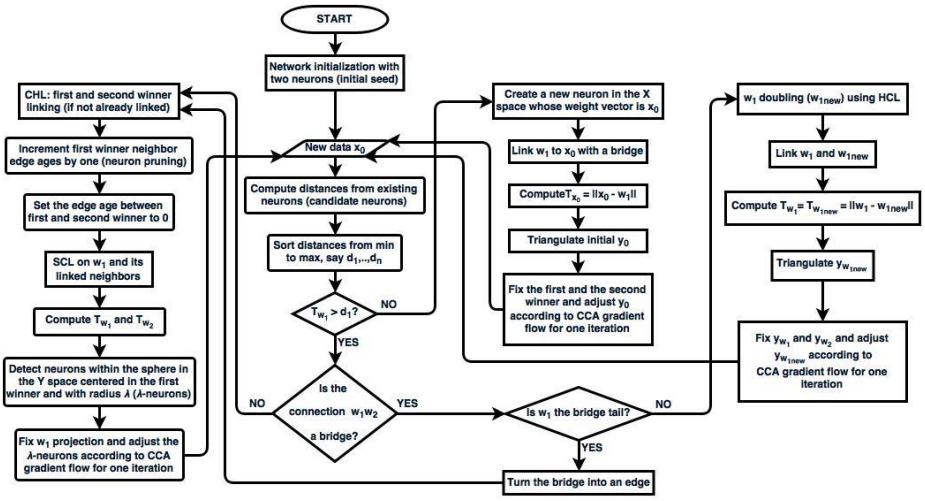

Fig. 1. The GCCA flowchart

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 in [13] is used as follows. The age of all other links emanating from the *winner* is incremented by one; if a link age is greater than the *agemax* scalar parameter, it is eliminated. If a neuron remains without links, it is removed (*pruning*). X-weights are adapted by using SCL [13]: x_{-1} and its direct topological neighbors are moved towards x_0 by fractions α_1 and α_n, respectively, of the total distance

$$\Delta x_{-i} = \alpha_1 (x_0 - x_{-i}) \qquad \text{i} = 1 \qquad (4a)$$

$$\Delta x_{-i} = \alpha_n (x_0 - x_{-i}) \qquad \text{otherwise} \qquad (4b)$$

and the thresholds of the winner and second winner are recomputed. Then 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 (3), 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 step 1 is done and the bridge becomes an edge. Otherwise, a seed is created by means of the neuron doubling:
   (a) A virtual adaptation of the X-weight of the winner is estimated by HCL (only (4a) 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 computed (it corresponds 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 (3) in which the projections of the two neurons of the bridge are considered as fixed, in order to estimate the final projection of the new neuron (*extrapolation*).

3.2 Considerations

The algorithm requires very few user-dependent parameters. They are needed for the CCA projection, the competitive learning and the pruning. The CCA projection requires the learning rate α and the λ parameter, which determines the choice of the neurons for the projection step. The selection of this parameter is very important, because a too small value could imply a collection of local projections without any coordination. Indeed, the accurate setting of λ is the way GCCA creates its global projection. Instead, the network is not very sensitive to the choice of the number of iterations for each projection. The neuron pruning requires setting the value of *edge-max*, i.e. the maximum value of the age before pruning: a too low value implies a smaller number of neurons. The constant learning parameters α_1 for the *first winner* (for CHL and HCL, see (4a)) and α_n: constant learning rate for the *first winner neighbors* (for CHL, see (4b)) are needed for the X-weight adaptation.

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. In case of abrupt change in the input distribution (jump), there are a few long bridges. In case of smooth displacement of data, the density of bridges is proportional to the displacement speed of the distribution. 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.

4 Examples

Two examples, showing a two-dimensional projection (for visualization) follow: the first one deals with a static unidimensional manifold embedded in the three-dimensional space, the second one, instead, with nonstationary data in a fault diagnosis.

All the simulations have been implemented on MATLAB®. The first deals with data drawn uniformly from a spiral distribution of 30000 noiseless points (see Fig. 2 left). The parameters of GCCA are the following: ρ=0.07, α_1=0.4, α_n=0.1, agemax=2, λ_i=20, λ_f=0.6, epochs$_i$=5, epochs$_f$=1, α_{cca}=0.001. The results are shown in Fig. 2 right after 30000 instants. It can be deduced that the quantization spans the input domain uniformly and the projection unfolds data correctly.

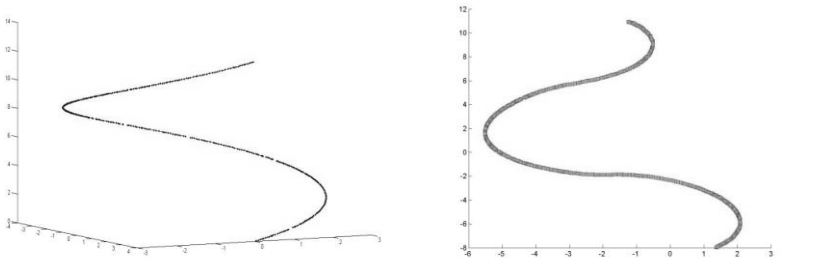

Fig. 2. 3D-Spiral (left); 2D projection (right)

The second example deals with a more challenging problem: data drawn from a dataset coming from the bearing failure diagnostic and prognostic platform [16], which provides access to accelerated bearing degradation tests. Here, the dataset contains 2155 5-dimensional vectors whose components correspond to statistical features extracted by measurements drawn from four vibration transducers installed in a kinematic chain of an electrical 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. The parameters of GCCA are the following: ρ=0.01, α_1=0.05, α_n=0.005, agemax=2, λ_i=20, λ_f=0.6, epochs$_i$=1, epochs$_f$=1, α_{cca}=0.01. The GCCA learns the chain behavior and tracks it, by adapting in real time the data projection. Fig. 3 shows the motor life-cycle, from the initial transient phase, through the healthy state, towards, first, a prefault (characterized by an increasing bridges density), and, finally, the two faults which are clearly identified in the figure by the longer bridges.

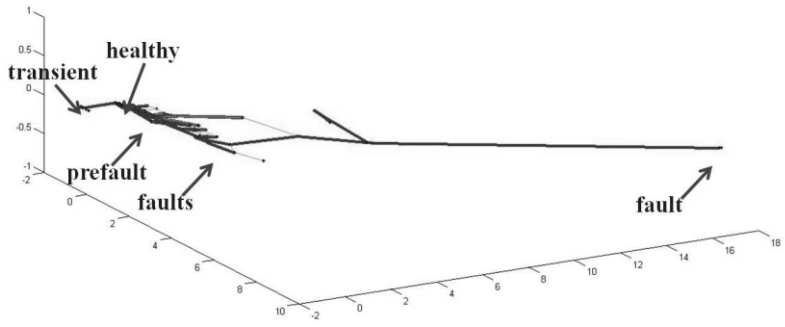

Fig. 3. GCCA edges and bridges for the bearing diagnostic experiment

5 Conclusion

The GCCA neural network is the only method able to track a nonstationary input distribution and to project it in a lower dimensional space. In a sense, GCCA learns 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 discriminates the outliers and yields the possibility (by its geometry and density) to infer more information about the nonstationarity; 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.

Future work will deal with the implementation in this network of other projection techniques, a deeper analysis of bridges and a minor change in the computation of the short distances for approximating the geodesic distances.

Acknowledgments

This work has been partly supported by OPLON Italian MIUR project.

References

1. Van der Maaten, L., Postma, E., Van der Herik, H.: *Dimensionality Reduction: a Comparative Review*. TiCC TR 2009-005, Delft University of Technology, 2009.
2. Sanger, T.D.: *Optimal Unsupervised Learning in a Single-Layer Neural Network*, Neural Networks, vol. 2, pp. 459-473, 1989.

3. Weng, J., Zhang, Y., 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.
4. Kouropteva, O., Okun, O., Pietikainen, M.: *Incremental locally linear embedding*, Pattern Recognition, vol. 38, pp. 1764-1767, 2005.
5. De Ridder, D., Duin, R.: *Sammon's mapping using neural networks: a comparison*, Pattern Recognition Letters, vol. 18, pp. 1307-1316, 1997.
6. Martinetz, T., Schulten, K.: *A "neural gas" network learns topologies*, Artificial Neural Networks, Elsevier, pp. 397–402, 1991.
7. Fritzke, B.: *A Growing Neural Gas Network Learns Topologies*, in Advances in Neural Information Processing System, 7, MIT Press, pp. 625-632, 1995.
8. White, R.: *Competitive Hebbian Learning: Algorithm and Demonstations*, Neural Networks, vol. 5, no. 2, pp. 261-275, 1992.
9. Martinetz, T., Schulten, K.: *Topology representing networks*, Neural Networks, vol. 7 (3), pp. 507-522, 1994.
10. Vathy-Fogarassy, A., Kiss, A., Abonyi, J.: *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.
11. Estevez, P., Figueroa, C.: *Online data visualization using the neural gas network*, Neural Networks, vol. 19, pp. 923-934, 2006.
12. Estevez, P., Chong, A., Held, C., Perez, C.: *Nonlinear projection using geodesic distances and the neural gas network*, Lect. Notes Comput. Sci., 4131, pp. 464-473, 2006.
13. Cirrincione, G., Hérault, J., Randazzo, V.: *The on-line curvilinear component analysis (onCCA) for real-time data reduction*, International Joint Conference on Neural Networks (IJCNN), pp. 157-165, 2015.
14. Demartines, P., Hérault, J.: *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.
15. Karbauskaitė, R., Dzemyda, G.: *Multidimensional data projection algorithms saving calculations of distances*, Information Technology and Control, vol.35, no.1, pp. 57-61, 2006.
16. Nasa prognostic data repository, `http://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository`