Curvilinear Component Analysis: a Self-Organizing Neural Network for Nonlinear Mapping of Data Sets

Pierre Demartines and Jeanny Hérault

Abstract— **We present a new strategy called "Curvilinear Component Analysis" (CCA), for dimensionality reduction and representation of multidimensional data sets. The principle of CCA is a self-organized neural network performing two tasks:** *Vector Quantization* **(VQ) of the submanifold in the data set (input space) and** *nonlinear Projection* **(P) of these quantizing vectors towards an output space, providing a revealing unfolding of the submanifold. After learning, the network has the ability to continuously map any new point from one space into another: forward mapping of new points in the input space, or backward mapping of an arbitrary position in the output space.**

Keywords— **Dimension Reduction, Self-Organizing Neural Network, Nonlinear Mapping, Interactive Data Exploration.**

I. Introduction

This algorithm was proposed (for example in [4] and [5]) as an improvement to the Kohonen Self-Organizing Maps (SOM) [12]; the output is not a fixed lattice but a continuous space able to take the shape of the submanifold. As it turns out, the Projection part of CCA is similar in its goal to other nonlinear mapping methods, such as Multidimensional Scaling (MDS) [19] and Sammon's Nonlinear Mapping (NLM) [18], in that it minimizes a cost function based on inter-point distances in both input and output spaces (for a review of such algorithms, see [2], [20], [14]). However, CCA significantly outperforms these algorithms in several aspects:

1. The use of a *new cost function*, able to unfold strongly nonlinear or even closed structures, and which allows a selection of the scale at which the unfolding of the submanifold has to take place.
2. Significant *speedup* due to an original method of minimization.
3. *Interactivity*, as a result of this speed, in which the user has control over the minimized function itself, principally on the *scale* at which the distances have to be preferably respected.
4. The user is helped in this task by a *representation of the mapping quality*, called "$dy - dx$", which is the joint distribution of input and output distances.

We observe experimentally that this human control gives "more revealing" results than the ones obtained by automatic methods or fixed cost functions, and does so in a shorter period of time. This is of course subjective, but not more than the concept itself of a "good mapping", and as an excellent illustration it is shown in [10] that the optimal solutions provided by different topological mappings can be radically different.

Institut National Polytechnique de Grenoble, Laboratoire de Traitement d'Images et de Reconnaissance des Formes, F-38031 Grenoble, France. Email: herault@tirf.inpg.fr, demartin@tirf.inpg.fr.

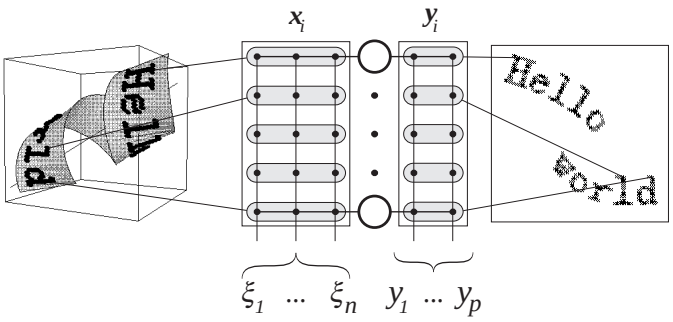

Fig. 1. Network structure of CCA and an example of its function: here, a 2-D mapping of a text initially folded onto a 3-D "flypaper". First, the unfolding of the flypaper has been learnt, and then the text has been projected using the mapping built between input and output spaces.

The purpose of CCA is primarily to give a *revealing representation* of data in low dimension, preparing a basis for further clustering and classification. We claim that this kind of representation helps to *understand* the structure of the data set and therefore to select the appropriate techniques for further automatic processing. For example, in the case of clustering, it has been shown [20] that human analysts helped by mapping techniques significantly outperform automatic clustering methods.

However, CCA can be more than a visualization technique, as shown in many applications described in [4] and [6].

II. Learning algorithm

Quantization and nonlinear mapping are separately performed by two layers of connections, as illustrated in fig. 1.

Each of the N neurons has two weight vectors. Input vectors $\{\boldsymbol{x}_i \ ; i = 1, \dots, N\}$ are n-dimensional, while corresponding output vectors $\{\boldsymbol{y}_i\}$ are p-dimensional ($p \leq n$).

The input vectors $\boldsymbol{x}_i$ are forced to become prototypes of the distribution using any of the several existing VQ methods (see for example [1], [9], [5], [4]). However, since we are interested to grasp the *submanifold* of the distribution rather than the density along it, we may choose a method which regularly quantizes the space covered by the data, regardless of the density. Such a method is the "Competitive Learning with Regularization" we introduced in [4].

The output layer must build a nonlinear mapping of the input vectors. In order to do so, Euclidean distances between $\boldsymbol{x}_i$'s: $X_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ are considered. Corresponding distances in the output space are $Y_{ij} = d(\boldsymbol{y}_i, \boldsymbol{y}_j)$. The goal is to force Y_{ij} to match X_{ij} for each possible pair (i, j). Since a perfect matching is not possible at all scales when manifold "unfolding" is needed to reduce the dimension

from n to p, a weighting function $F(Y_{ij}, \lambda_y)$ is introduced, yielding the quadratic cost function:

$$E = \frac{1}{2} \sum_i \sum_{j \neq i} (X_{ij} - Y_{ij})^2 F(Y_{ij}, \lambda_y). \quad (1)$$

Generally, $F(Y_{ij}, \lambda_y)$ is chosen as a bounded and monotically decreasing function, in order to favour local topology conservation (like in SOM). Decreasing exponential, sigmoid, or Lorentz function are all suitable choices. In the simulations we report here, we used the simple step function: $F(Y_{ij}, \lambda_y) = \begin{cases} 1 & \text{if } Y_{ij} \leq \lambda_y \\ 0 & \text{if } Y_{ij} > \lambda_y \end{cases}$. Key points:

1. As in SOM, the weighting is done as a function of *output* distances Y_{ij}, and not input ones like in most implementations of NLM. This *recursive* solution is the only one to unfold strongly folded data.
2. $F(Y_{ij}, \lambda_y)$ should be bounded. A function like $1/Y_{ij}$, mentioned as a possible variant for NLM [2], is not appropriate because of too strong influence of small Y_{ij}'s.
3. $\lambda_y(t)$ generally evolves with the time. It can be a fixed schedule, like the neighborhood in SOM, but
4. λ_y can also be controlled by the user, allowing an *interactive* selection of the scale at which the unfolding takes place.

The minimization of the cost function (1) with respect to the $\boldsymbol{y}_i$'s is done with a new procedure that we introduce below. Compared to other methods such as the stochastic gradient descent ($\Delta \boldsymbol{y}_i \approx -\boldsymbol{\nabla}_i E$) or the steepest gradient descent:

1. It saves a tremendous amount of computing time (see fig. 4).
2. While the average of the updates is proportional to the opposite of the gradient of E, it can temporarily produce increases of E. This allows the algorithm to eventually escape from local minima[1] of E, and in practice we observe lower final cost by contrast with gradient methods (see fig. 4).

Instead of moving one vector $\boldsymbol{y}_i$ according to the sum of every other $\boldsymbol{y}_j$'s influences, as it would be the case with a stochastic gradient descent, we temporarily *pin* one $\boldsymbol{y}_i$ and move all the other $\boldsymbol{y}_j$ around, without regard to interactions amongst the $\boldsymbol{y}_j$. If we rewrite (1) as a sum of partial costs:

$$E = \frac{1}{2} \sum_i \sum_{j \neq i} E_{ij}, \quad (2)$$

then the proposed rule is, given i (randomly chosen):

$$\Delta \boldsymbol{y}_j(i) = \alpha(t) \boldsymbol{\nabla}_i E_{ij} = -\alpha(t) \boldsymbol{\nabla}_j E_{ij}. \quad (3)$$

Similarly to usual stochastic gradient methods, $\alpha(t)$ decreases with the time, for example $\alpha(t) = \alpha_0/(1+t)$.

Considering a quantized version of $F(Y_{ij}, \lambda_y)$ (that is, with $\partial F / \partial Y_{ij} = 0$), we obtain the simple expression, very easy to understand and which works well in most cases:

$$\boxed{\Delta \boldsymbol{y}_j = \alpha(t) F(Y_{ij}, \lambda_y)(X_{ij} - Y_{ij}) \frac{\boldsymbol{y}_j - \boldsymbol{y}_i}{Y_{ij}} \quad \forall j \neq i.} \quad (4)$$

From the computational point of view, this rule is much lighter than a stochastic gradient. For an adaptation cycle of all units (except i), the complexity is only in $O(N)$ instead of $O(N^2)$. It is not necessary to compute all the $N(N-1)/2$ distances in both input and output spaces, but only the distances from unit i to the others. This speedup is illustrated in fig. 4.

Let us consider first the cost variation ΔE, given i:

$$\Delta E(i) = \sum_{j \neq i} (\boldsymbol{\nabla}_j E)^T (\Delta \boldsymbol{y}_j) \quad (5)$$

$$= \sum_{j \neq i} \left(\sum_{p \neq j} \boldsymbol{\nabla}_j E_{pj} \right)^T (-\alpha(t) \boldsymbol{\nabla}_j E_{ij})$$

$$= -\alpha(t) \sum_{j \neq i} \left[\| \boldsymbol{\nabla}_j E_{ij} \|^2 + \sum_{p \neq i,j} (\boldsymbol{\nabla}_j E_{pj})^T (\boldsymbol{\nabla}_j E_{ij}) \right] \quad (6)$$

The second term under the sum can happen to be negative, yielding a possible temporary increase of E. However, the expectation $\widehat{\Delta \boldsymbol{y}_j}$ of the move of a particular y_j with respect to all possible choices of i is:

$$\widehat{\Delta \boldsymbol{y}_j} = -\alpha \frac{1}{N} \sum_i \boldsymbol{\nabla}_j E_{ij} = -\alpha \frac{1}{N} \boldsymbol{\nabla}_j E, \quad (7)$$

which, put together with (5), yields $\widehat{\Delta E} \leq 0$. As a consequence, the cost E can momentarily increase, but it decreases on average. This behavior is illustrated in fig. 4. It is observed that a much deeper minimum is reached in contrast to stochastic gradient or steepest descent.

This optimized rule is very efficient: for a network of a thousand of neurons ($N = 1000$) and with a linear manifold, generally about fifty iterations are enough to reach a perfectly organized output state ($|X_{ij} - Y_{ij}| \leq 10^{-6}$ of the maximum distance X_{ij} which is roughly the diameter of the data set). When the submanifold is not linear (thus an unfolding has to be done), it takes more steps to converge (some thousands), but it is still much faster (in CPU time) than a stochastic gradient descent.

III. "$dy - dx$" REPRESENTATION

In order to check the topology preservation of Kohonen SOM, we have proposed in [3] a representation which is called "$dy - dx$". It consists in the joint distribution of weight distances and grid distances between pairs of neurons: for each possible pair, one plots a point at position $[dy, dx]$, where dy is the physical distance between the neurons (the distance on the grid) and dx the distance between their weight vectors. For a well topology-preserving mapping, dy should be proportional to dx, at least for small dy's. This representation is directly usable in the framework of CCA: dx simply becomes X_{ij} and dy becomes Y_{ij};

[1] but we do not prove the convergence toward a global minimum

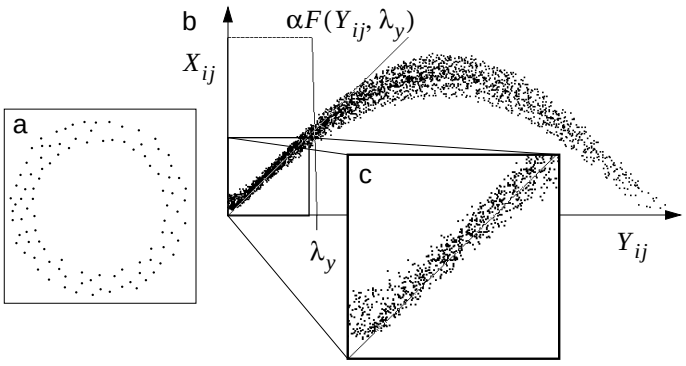

Fig. 2. Illustration of "$dy - dx$" representation, showing two different aspects of CCA: local projection and unfolding. The data is spread (because of some noise) around a circular submanifold (a). A nonlinear mapping toward a one-dimensional space is found by CCA. The result (not shown here) is the average circle, split somewhere and unfolded to lie on the one-dimensional output space. (b): Global aspect of "$dy - dx$" representation, showing the unfolding effect. (c): Zoom around the origin, showing the local projection effect.

this actually corresponds to the "inter-point distances reconstruction plot" of Shepard and Carroll [19].

It is relatively easy to analyze a mapping by means of this representation and to take the relevant action, for example concerning the neighborhood parameter λ_y. For this reason we plot the function $\alpha F(Y_{ij}, \lambda_y)$ directly on the $dy - dx$ representation. The user can continuously change α and λ_y with the mouse.

Interpretation (see fig. 2):

1. Correct matches ($Y_{ij} \approx X_{ij}$) give points close to the line corresponding to the identity function. If the dimension reduction from n to p can be done by linear projection, then the mapping is easily found and the complete "$dy - dx$" characteristic lies on the identity function. That generally suggests that one could try a mapping with a smaller output dimension, where unfolding would be necessary.
2. In this representation, a locally correct mapping is shown by a distribution close to the identity function, near the origin. On the other hand, unfolding is revealed by bent and spread "$dy - dx$" characteristic where $Y_{ij} > X_{ij}$ in average.
3. Close to the origin, points that are above this line ($Y_{ij} < X_{ij}$) can be interpreted as a local projection (see fig. 2.c).

This evolution of $\lambda_y(t)$ can be driven by an automatic schedule, like the neighborhood radius in SOM for example. However, we have noticed that, by tuning $\lambda_y(t)$ by hand, one can dramatically improve the accuracy of the mapping and realize a trade-off between global unfolding and local projection qualities, only by observing the shape of "$dy - dx$" representation.

The choice of another parameter has been hidden so far: the output dimension. Ideally, it should be the dimension of the submanifold. This can be roughly estimated by fractal dimension analysis of the data set. However, noise in the data and the small number of available points often make this fractal analysis difficult or impracticable. In particular, due to the discrete nature of data sets, one has to select a *scale* of observation (the limit used in the mathematical formulation of fractal dimensions is not applicable).

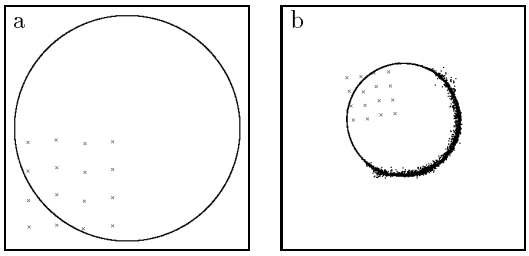

Fig. 3. Projection of a circle with a CCA network having learned a small square. (a) Input space, with the weights quantizing the learned distribution (small square), and the test distribution of circular shape. (b) Output space, with output learned weights, and the interpolation/extrapolation of the circle. The error of extrapolation (here emphasized for purpose of visualization) is of order of 0.1% of the diametra of the circle.

We usually proceed as follow: a rough estimation of fractal dimension is taken as a starting point for p. Then, we tune the dimension considering the "$dy - dx$" representation, as explained above, which has been observed much more accurate and informative.

IV. Continuous mapping and backward mapping

The relation $\boldsymbol{x} \rightarrow \boldsymbol{y}$ is quantized by N prototypes ($\boldsymbol{x}_i \rightarrow \boldsymbol{y}_i$). To obtain the continuous mapping $\boldsymbol{y}_0$ of any supplementary point $\boldsymbol{x}_0$ from the input distribution, considered as a temporary new prototype, the same cost function (1) as for the learning will be minimized, but only with regard to the new corresponding $\boldsymbol{y}_0$. Thus, instead of moving each vector with respect to each other, only one point $\boldsymbol{y}_0$ is adapted according to a simple stochastic gradient descent, while all the others are kept fixed. Therefore, this point $\boldsymbol{y}_0$ is searched with respect to the $\boldsymbol{y}_i$'s in function of the measured distances X_{i0} between $\boldsymbol{x}_0$ and the $\boldsymbol{x}_i$'s. It is actually a local mapping, and the initialization of $\boldsymbol{y}_0$ is made with the few first neighbors.

This procedure gives very accurate interpolation, but also good extrapolation, which is a particularly unusual feature for artificial neural networks, and which can truly be called "generalization". It is illustrated in fig. 3. It is homogeneous with the learning algorithm and does not need other parameters (such as radii or local Jacobian matrices). However, each point of the continuous mapping needs several adaptation steps, and the parameters schedule during this "sub-convergence" is empiric up to now.

This continuous mapping is invertible: *backward mapping* can be obtained by simply swapping input and output weights and spaces and using the same algorithm.

V. Comparison with other algorithms

- Classical MDS [21] and PCA [7] both find the axes of maximum variance of input data and represent them by a linear projection onto a subspace of reduced dimension [15]. In the case of Euclidean distances, the cost function to be minimized is $E = \sum_i \sum_{j \neq i} (X_{ij}^2 - Y_{ij}^2)$ under constraint that $Y_{ij} \leq X_{ij}$

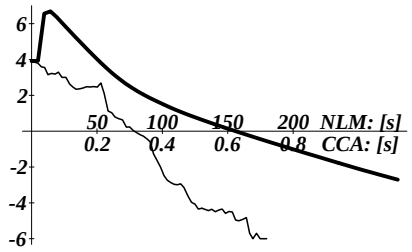

Fig. 4. Comparison of the cost minimization for CCA and NLM. The input space is quantized by 256 points. The submanifold is simply a square, not folded. It is then possible to plot the common part of the costs of both algorithms: $E_{lin} = \frac{1}{2}\sum_i \sum_{j\neq i}(X_{ij} - Y_{ij})^2$. The vertical axis is logarithmic. The horizontal axis represents the CPU time in seconds on a SUNSparc 10 mod 41. In order to represent these plots on the same graph, it is necessary to adopt a different scale for the two algorithms. NLM: final time = 4mn40s; final $E = 0.0025$; after a violent expansion due to $1/X_{ij}$, the cost decreases slowly and smoothly. CCA: final time = 0.72s; final $E < 10^{-6}$; the costs decreases on average, but increases occasionally, to finally reach quicker a much lower minimum. Note that the complete convergence of CCA shown here is achieved before the first iteration of NLM has been completed.

(because of projection). The minimum is quadratic in the neighborhood of the solution. If non Euclidean distances are considered, then the cost function is $E = \sum_i \sum_{j\neq i}(\boldsymbol{x}_i^T \boldsymbol{x}_j - \boldsymbol{y}_i^T \boldsymbol{y}_j)^2$, which means a biquadratic minimum, hence very sensitive to noise, remains a linear approximation of nonlinear data structure.

- Non metric MDS [13] can take into account nonlinear data structures. The cost function implies rank orders of the inter-point distances rather than distances themselves:

$$S = \frac{\sum_i \sum_{j\neq i} (\mathrm{rank}(X_{ij}) - \mathrm{rank}(Y_{ij}))^2}{\sum_i \sum_{j\neq i} \mathrm{rank}(X_{ij})} \qquad (8)$$

The minimum is quadratic. The topology fails to be correctly represented because of the quantization errors introduced by rank ordering instead of true continuous distances.

- Shepard's Non-Linear Multidimensional Scaling has the closest performances to CCA. However, due to a very complex cost function:

$$\kappa = \sum_i \sum_{j\neq i} \frac{X_{ij}^2}{Y_{ij}^4} \Bigg/ \left(\sum_i \sum_{j\neq i} \frac{1}{Y_{ij}^2} \right)^2 , \qquad (9)$$

it is computationally very demanding. Moreover, this cost function suffers from three other drawbacks: as the number of data points increases, 1) the global minimum gets flatter, leading to higher sensitivity to noise; 2) local minima get sharper and deeper and the walls between them get higher, so it is very difficult to escape from them; 3) the cost function rapidly vanishes to zero when some output distances are large. All these considerations lead to severe difficulties to find a solution, especially concerning the choice of initial conditions.

- Sammon's NLM. Here, the cost function resembles CCA's one:

$$E = \frac{1}{c} \sum_i \sum_{j<i} (X_{ij} - Y_{ij})^2 \frac{1}{X_{ij}} \qquad (10)$$

(with $c = \sum_i \sum_{j<i} X_{ij}$ a normalization constant). The $F(Y_{ij}, \lambda_y)$ of CCA is replaced in $1/X_{ij}$. This means that short range distances of *input space* are favored, so the unfolding is very difficult to obtain (see fig. 5), depending on particular problems and on the initial configuration. Besides, points that are close in the input space, yielding $X_{ij} \approx 0$, disturb badly the cost function.
Finally, due to the minimization algorithm, the NLM has a complexity of $O(N^2)$ instead of $O(N)$ for CCA. This is illustrated in fig. 4.

- Kohonen SOM often have been thought to perform non linear mapping, but when they succeed at that, it is only by chance: they perform a vector quantization *under the constraint of a predefined neighborhood between neurons*. Hence, they map a discretized grid of given shape to some *unknown* input distribution regardless to the actual shape of the submanifold. In contrast, CCA automatically finds the appropriate shape of the submanifold: the neurons "search" a suitable position in the output space such that the local input topology is preserved as well as the global shape of the submanifold.
- Growing Cell Structures [8], Neural Gas [16], and other attempts to escape from the crisp grid of Kohonen SOM generally lose the concept of an output or representation space. Hence, the submanifold is well captured, but no method is provided to represent it. Generally, one can only consider local information as local dimension, local connectivity, and so on. However, since they are very fast VQ methods, they can be used for the input layer of CCA.
- The *generalization property* is often not or badly achieved by most networks whose target is to continuously link an output with an input space, and which generally perform interpolation only. For example, with Radial Basis Functions (RBF) networks, or some SOM implementations or with "Counter-Propagation network" ([11]), this interpolation is made by computation of a center of gravity weighted by kernel functions. Since these kernels are strictly positive (generally Gaussian), extrapolation is impossible. Thus, since the quantization process does not place any vector at the distribution boundaries, there is a stripe around the weight vectors which is not correctly mapped.

VI. Artificial examples

In this section, we illustrate the main features of CCA (speed and unfolding) with three synthetic examples

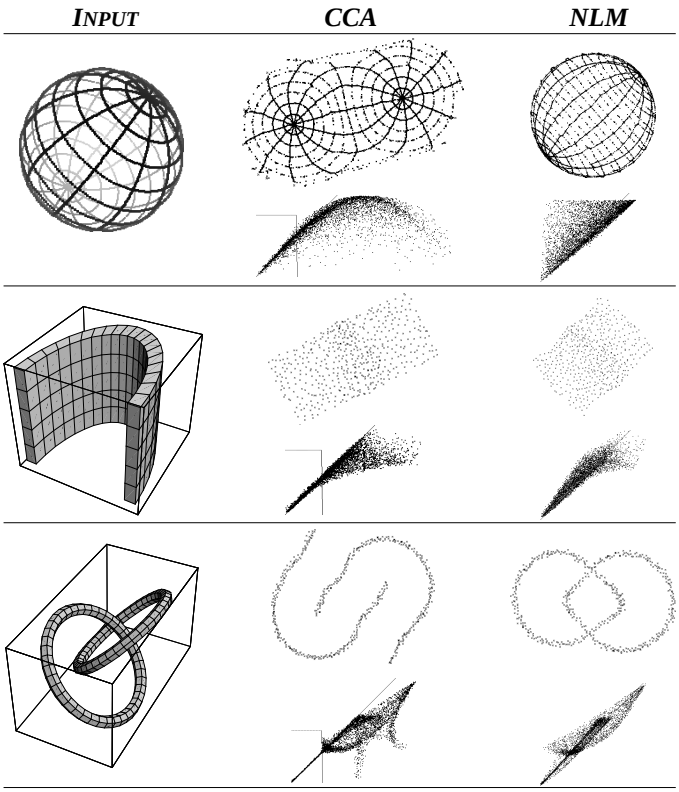

Fig. 5. Three synthetic examples, and the respective solutions of CCA and NLM (discussion in the text). The "$dy - dx$" representation is provided with the output space. Note also the final position of λ_y, revealed by the step function shown in superposition to '$dy - dx$" plots for CCA. Every distribution was quantized with 500 points. Each run is stopped when no visual improvement is achieved. For each case, several runs have been done, and the shortest time is retained. Sphere: CCA 8s, NLM 5mn41s; U-fold: CCA 6s, NLM 13mn47s; 2-rings: CCA 9s, NLM 13mn3s.

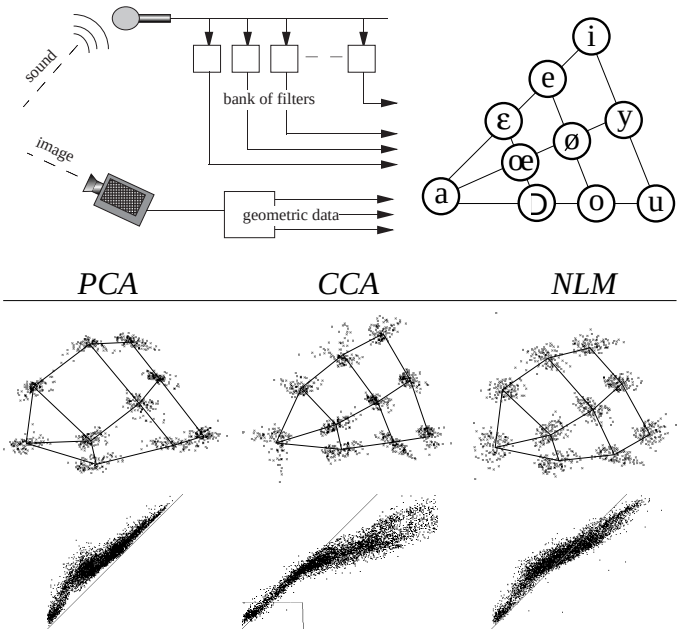

Fig. 6. Experience of audio and visual data fusion. First row: acquisition system and theoretical "vocalic triangle". Second row: reconstruction of the vocalic triangle in a two-dimensional space, by three different methods. See text for discussion.

(fig. 5). For comparison, the solution found by NLM (version implementing the steepest gradient descent) is also given.

Here for CCA, both α and λ_y have been interactively tuned by the user during the run. First line: NLM is unable to unfold the sphere (there is not much difference from a linear projection), whereas CCA provides a good mapping. Second line: the NLM reveals correctly the submanifold, but with the curvilinear parameter shrunk. CCA unfolds correctly this data set, without this shrinking effect. Third line: in order to untangle the two rings, CCA has to break them to be able to put them on the plane. Note the perfect topological conservation for $Y_{ij} < \lambda_y$, visible on "$dy - dx$". The NLM is unable to extricate the rings.

VII. Real-world example: Phoneme representation

The problem we address here concerns the field of speech analysis. One of the classical techniques for the processing of vowels is to feed the vocal signal into a bank of band-pass filters in order to classify the various configurations of energies at the output of these filters. Of course, the ambient acoustic noise is an important source of errors for the classifier. A recent method [17] consists in taking supplementary information, of visual kind (shape of the mouth), in order to reduce the effect of noise. This is the principle of data fusion.

In our studied case, the acoustic signal is analyzed by a bank of 20 audio band-pass filters, and in the mean time, three visual parameters are recorded: width, height and surface of the speaker's mouth. The data have been kindly provided by Institut de la Communication Parlée, Grenoble, France. Fig. 6 shows the data acquisition system, the theoretical "vocalic triangle", and the low-dimensional view provided by three different algorithms: PCA, CCA and NLM. On these representations, lines between the centroids of neighbor clusters have been added for visual comparison. The number of points quantizing the input space is 1000.

The three projections, at a first look, seem very similar. This is due to the fact the submanifold of the vocalic triangle in the 23-dimensional input space is not too strongly folded. However, a closer analysis reveals:

1. the clusters provided by the PCA are not too much scattered, but they occasionally mix with their neighbors.
2. with CCA, the clusters are better separated (the minimum interclass distance is the largest of the three methods). The "$dy - dx$" representation also reveals that the unfolding is the strongest for CCA. Typical time for the run: 40 seconds.
3. the NLM also gives a good result: the clusters are also linearly separable. However, 1– the minimum interclass distance is smaller than for CCA, 2– the cluster diameters are larger, 3– several points are mapped completely wrongly, even outside of the picture, and it has not been possible, in spite of several runs, to avoid

this problem, 4– each run took about 30 minutes of CPU.

VIII. Conclusion

CCA is a technique of data representation which borrows both the ideas of multivariate data analysis and the principles of self-organizing neural networks. In principle, once the intrinsic dimension of the submanifold spanned by the data has been found, it provides an interactive and fast nonlinear mapping useful for data exploration. The advantages of this method are:

1. unfolding of the submanifold, even in strongly nonlinear cases,
2. local projection of the high-dimensional residual noise or low-variance components on the image of the submanifold,
3. due to a special adaptation rule, the algorithm is very fast while experimentally shown to reach deeper minima than simple gradient methods,
4. thanks to this speed, the user can interactively select the range of distances preferably preserved,
5. after learning, the same algorithm applies to continuous mapping and is capable of accurate interpolation and extrapolation,
6. this continuous mapping can also be used in reverse mode in order to map the output space into the input submanifold, a property which can be useful for further data mining considerations,
7. CCA is recursive and data-driven. Associated to a continuously adaptive VQ scheme, it can be easily adapted to provide pursuit abilities in the case of non-stationary data.

Up to now CCA has been successfully applied to various difficult non-linear problems of data representation in the frameworks of process surveillance, sensor fusion and generation of metric spaces from non metric cost functions ([4], [6]). As for many of the algorithms of the same kind, theoretical proofs of convergence are not available. Only experimental data and intuitive considerations have been derived. All that can be said at the moment is that CCA overcomes many of the drawbacks of other nonlinear mapping algorithms and that it explicitly supports the concept of "data unfolding" which we consider to be one of the first steps for the understanding of nonlinear data structure.

References

[1] A. Ahalt, A. K. Krishnamurthy, Chen P., and D. E. Melton. Competitive learning algorithms for vector quantization. *Neural Networks*, 3:277–290, 1990.

[2] Y. Chien. *Interactive Pattern Recognition*. Marcel Dekker, Inc., New York, 1978.

[3] P. Demartines. Mesures d'organisation du réseau de Kohonen. In M. Cottrell, editor, *Congrès Satellite du Congrès Européen de Mathématiques: Aspects Théoriques des Réseaux de Neurones*, 1992.

[4] P. Demartines. *Analyse de données par réseaux de neurones auto-organisés*. PhD thesis, Institut National Polytechnique de Grenoble, 1994.

[5] P. Demartines and J. Hérault. CCA: "Curvilinear Component Analysis". In *GRETSI'95*, Juan-les-pins, France, September 1995.

[6] P. Demartines and J. Hérault. The Curvilinear Component Analysis. Technical Report TR-96-038, International Computer Science Institute, September 1996.

[7] E. Diday, J. Lemaire, P. Pouget, and F. Testu. *Eléments d'analyse de données*. Dunod, Paris, 1983.

[8] B. Fritzke. Let it grow — self-organizing feature maps with problem dependent cell structure. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, volume 1, pages 403–408, North-Holland, 1991. Elsevier Science Publishers.

[9] A. Gersho and Robert M. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, London, 1992.

[10] G. J. Goodhill, S. Finch, and T. J. Sejnowski. Quantifying neighbourhood preservation in topographic mappings. Technical Report INC-9505, Institute for Neural Computation, November 1995.

[11] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the theory of neural computation*, volume 1 of *Santa Fe Institute Lecture Notes*. Addison-Wesley Publishing Company, 1991.

[12] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 3rd edition, 1989.

[13] J. B. Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29:115–129, June 1964.

[14] J. Mao and A. K. Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transaction on Neural Networks*, 6(2):296–317, March 1995.

[15] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, London, 1979.

[16] T. Martinetz and K. Schulten. A neural gas network learns topologies. In T. Kohonen *et al.*, editor, *IEEE International Conference on Artificial Neural Networks, Espoo, Finland*, volume 1, pages 397–407. Elsevier, 1991.

[17] J. Robert-Ribes, J.L. Schwarz, and P. Escudier. A comparison of models for fusion of the auditory and the visual sensors in speech perception. *Artificial Intelligence Review*, 9(4-5).

[18] J. W. Sammon. A nonlinear mapping algorithm for data structure analysis. *IEEE Trans. Computers*, C-18(5):401–409, 1969.

[19] R. N. Shepard and J. D. Carroll. Parametric representation of nonlinear data structures. In P. R. Krishnaiah, editor, *International Symposium on Multivariate Analysis*, pages 561–592. Academic Press, 1965.

[20] W. Siedlecki, K. Siedlecka, and J. Sklansky. Experiments on mapping techniques for exploratory pattern analysis. *Pattern Recognition*, 21(5):431–438.

[21] W.S. Torgerson. Multidimensional scaling, i: theory and method. *Psychometrika*, 17:401–419, 1952.