CCA: "Curvilinear Component Analysis"

Pierre Demartines* and Jeanny Hérault

INPG, Labo. TIRF, 46 av. Félix-Viallet, F-38031 Grenoble, France
(*): now with CSEM, Maladière 71, CH-2007 Neuchâtel, Switzerland
e-mail: demartin@tirf.inpg.fr, herault@tirf.inpg.fr

RÉSUMÉ

Résumé : L'ACC est un réseau de neurones auto-organisé qui donne une carte de la sous-variété d'un nuage de données en grandes dimensions non linéairement dépendantes. Le principe est de construire une relation entre un espace d'entrée (les données) et un espace de sortie (la carte) au moyen d'un ensemble de neurones ayant chacun deux vecteurs-poids : un pour l'entrée et l'autre pour la sortie. Après avoir quantifié la distribution par les vecteurs d'entrée, les distances entre ces vecteurs sont copiées dans l'espace de sortie, tout en favorisant les petites distances de sortie. On obtient alors le dépliage de la variété des données avec réduction de dimension. Après apprentissage, le même algorithme peut être utilisé pour projeter continûment n'importe quel point de la distribution, avec d'excellentes caractéristiques en interpolation et en extrapolation. L'ACC peut être employée dans plusieurs domaines comme la fusion de données, l'appariement de graphes, l'analyse et la surveillance de procédés industriels, la détection de pannes dans des machines, la cartographie de concepts et le routage adaptatif en télécommunications.

ABSTRACT

Abstract : CCA is a self-organizing neural network which gives a revealing low-dimensional mapping of the submanifold of a high-dimensional and non linearly related data set. The principle is to build a relation between an input space (data) and an output space (the expected mapping) through a set of neurons, each having two weight vectors: one for the input and the other one for the output. After driving the input vectors to a vector quantization of the input data set, the distances between input vectors are copied in the output space, while favouring short-range output distances. Then, one obtains the unfolding of the data submanifold together with a dimension reduction. After learning, the same projection algorithm can be used to map continuously any point of the distribution, leading to excellent interpolation and extrapolation properties, which is an original result. CCA can be used in several domains such as data fusion, graph matching, industrial process monitoring or analysis, faults detection in devices, concept mapping and adaptive routing in telecommunications.

1 Introduction

The Kohonen's Self-Organizing Maps (SOM) are a kind of artificial neural network historically inspired by sensory maps found in biology ([17, 11, 12]). They are well known for their ability to provide a data mapping where both sample number and dimensionality are reduced, and can be viewed as a non linear extension of Principal Component Analysis (PCA) [2, 15, 14]. However, SOM have a major drawback: the mapping is done toward a grid of neurons whose shape is *a priori* fixed and may not comply with the one of the data submanifold, leading to confuse mapping. The grid is generally rectangular or hexagonal, and for most applications (especially with high-dimensional data space), it is even chosen as a square because of the lack of further knowledge about the real shape of the data submanifold.

We have proposed a new model, initially called VQP neural network ("*Vector Quantization and Projection*", [8, 7]) which overcomes this drawback. The function looks like that of Kohonen network, that is, a vector quantization in addition with a topologically correct mapping (at least locally). However, the principle

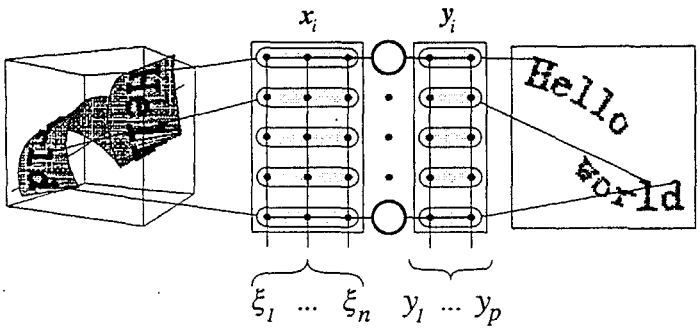

Figure 1: VQP network structure and an illustration of its function. Here, a 2-D mapping of a text which is folded onto a 3-D "flypaper" is done. In essence, this function consists in revealing the underlying submanifold of a data set through unfolding and dimension reduction, leading to the concept of "Curvilinear Component Analysis".

is completely different: instead of performing a vector quantization *under the constraint of a predefined neighborhood between neurons*, quantization and mapping functions are separately performed by two layers of connections (figure 1).

The input vectors are forced to become prototypes of the distribution by means of any of the several existing

methods of vector quantization. Each neuron has also a vector pointing towards position within a continuous output space the dimension of which is the only predefined characteristic. At the beginning, output vectors are randomly initialized. Then, the learning algorithm consists to move them with respect to each other in order to reproduce as much as possible the configuration of the input vectors. These output positions are totally free in the space (conversely to SOM where they are locked onto a grid). Therefore, a much larger variety of distributions can be mapped while avoiding dead units.

This "Curvilinear Component Analysis" (CCA) is a useful method for redundant and non linear data structure representation. It provides revealing curvilinear views of even strongly folded structures whereas Principal Component Analysis (PCA) or other linear methods fail to give such a suitable information. For example, figure 1 shows this ability to map and represent folded structures.

2 Algorithm

Consider N neurons whose n-dimensional input vectors $\{\boldsymbol{x}_i\ ; i = 1, \ldots, N\}$ quantize the input distribution (see for example [1, 9, 8]). Their p-dimensional output vectors $\{\boldsymbol{y}_i\}$ should copy the topology of the $\boldsymbol{x}_i$'s. In order to do that, 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 this is not possible when manifold "unfolding" is needed for reducing the dimension from n to p, the matching of short-range output distances is favoured by a monotically decreasing weighting function of Y_{ij}'s. Then we derive an energy function to be minimized:

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

with $F(Y_{ij}) \geq 0$ a decreasing function of Y, in order to favour local topology conservation.

The minimization is not made through a strict gradient descent. We prefer a simple new procedure that is equivalent *in average*, but which is much quicker and allows to escape from casual local minima of energy.

In a conventional stochastic gradient descent step, the output vector $\boldsymbol{y}_i$ of a randomly selected neuron i would be adapted accordingly to:

$$\Delta \boldsymbol{y}_i = -\alpha \nabla_i E = \alpha \sum_{j \neq i} G(X_{ij}, Y_{ij})(\boldsymbol{y}_i - \boldsymbol{y}_j), \qquad (2)$$

where $\nabla_i E$ denotes the gradient of E with respect to $\boldsymbol{y}_i$, and

$$G(X_{ij}, Y_{ij}) = \frac{X_{ij} - Y_{ij}}{Y_{ij}} \left[2F(Y_{ij}) - (X_{ij} - Y_{ij}) F'(Y_{ij}) \right].$$

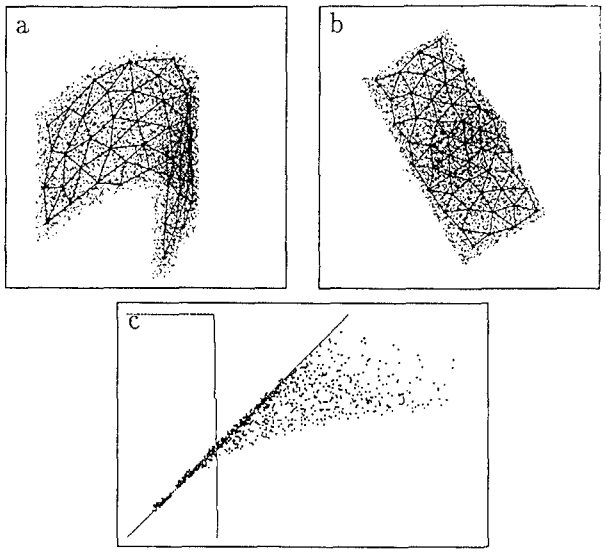

Figure 2: Mapping of a folded distribution. (a): input distribution and input weights. (b): output weights and continuous projection of the input distribution. (c): $dy - dx$ representation, showing that the mapping is very accurate only locally: for small distances Y_{ij}, X_{ij} are strongly non linear: for large output distances, Y_{ij} and X_{ij} are not well correlated. However, the topology is locally preserved.

(2) is a sum of radial attractions or repulsions due to all neurons $j \neq i$. This rule, that we call "*intraverted gradient*" suffers of several drawbacks. Only one neuron is adapted at a time, thus the adaptation of all neurons is heavy (complexity in order $O(N^2)$ for updating each neuron once). It is slow, because the sum produces a "average effect" that leads units to have small movements. Moreover, it can fall into local minima.

In contrast, a step of the empiric procedure that we use (and that we call "*extraverted gradient*") consists in randomly selecting a neuron i, momentarily freezing its output weight vectors $\boldsymbol{y}_i$, and radially moving all $\boldsymbol{y}_{j \neq i}$ regardless of the interactions among them. Considering a stepwise version of F (that is, with $F' = 0$), we finally obtain the simple rule:

$$\boxed{\Delta \boldsymbol{y}_j = \alpha F(Y_{ij}) \frac{X_{ij} - Y_{ij}}{Y_{ij}} (\boldsymbol{y}_j - \boldsymbol{y}_i) \quad \forall j \neq i.} \qquad (3)$$

Though all the $\boldsymbol{y}_j$ (but i) are moved at each step, this procedure has only a complexity of order $O(N)$, whereas the equivalent stochastic gradient descent is of $O(N^2)$. Despite of possible momentarily energy increases, the mean of the updates is proportional to the opposite of the energy gradient ($\mathrm{E}(\Delta \boldsymbol{y}_j) = -\frac{\alpha}{N} \nabla_j E$), therefore the energy decreases in average [7]. This simple and empiric rule is very efficient: for a network of a thousand of neurons ($N = 1000$), generally only about fifty iterations are sufficient to reach a perfectly organized output state ($E \leq 10^{-6}$ of the max distance Xij), when the submanifold is linear. When the submanifold is not linear, thus an unfolding has to be done, it takes

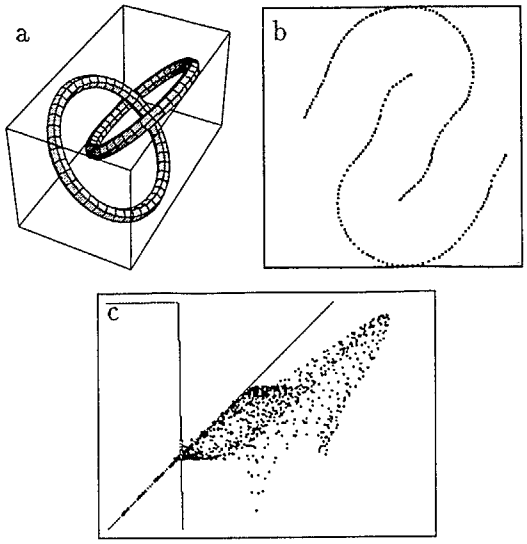

Figure 3: Mapping of two interlaced rings. The input space is in 3 dimensions. The output space (constraint to 2 dimensions) gives the mapping found: the network cut the rings in order to put them on the plane.

more steps to converge (some thousands), but remains much faster than a stochastic gradient.

$F(Y) = \exp(-Y/\lambda_y)$ or even $F(Y) = \mathbf{1}(Y - \lambda_y)$ (step function) can be used as weighting function in (1). The "neighborhood radius" λ_y, as well as the adaptation factor α are decreasing with time, like in usual Kohonen's SOM.

This algorithm compares favourably to existing non linear algorithms, such as the Sammon's Non Linear Mapping algorithm (NLM) [16] which is N times slower (in CPU time) and which fails to find good representations of strongly folded structures (because the weighting of energy terms depends on input distances (X_{ij}) only).

3 Output dimension and "$dy - dx$" representation

The output space is continuous and the mapping automatically takes the relevant shape. However, its dimension remains, as for Kohonen maps, a parameter that has to be fixed at the beginning. This dimension should correspond to the number of degrees of freedom of the data distribution, i.e. the number of free parameters in the phenomenon underlying the data set. This number can be roughly estimated through a fractal dimension analysis of the data set.

Thus, we propose to base the choice of output dimension on fractal dimension analysis of the data set, in addition to any initial knowledge that can be collected about the data set and the underlying process. Then, this choice can be tuned in regard to a mapping quality representation explained hereafter.

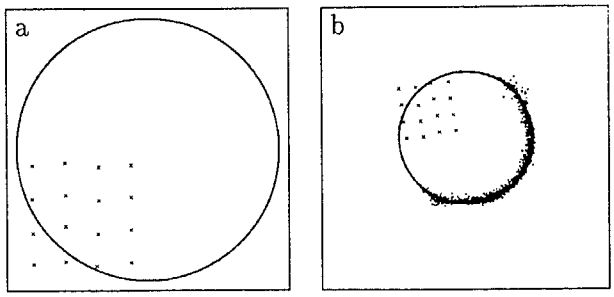

Figure 4: Projection of a circle with a VQP network having learned a small square. (a) Input space, with the weights quantizing the learned distribution (small square), and the test distribution of circular shape. (b) Output space, with output learned weights, and the interpolation/extrapolation of the circle.

In order to check the topology-preservation of Kohonen map, we have proposed in [6, 5] a representation which is called "$dy - dx$". It consists in the joint distribution of input and output distances between pairs of neurons: for each possible pair, one plots a point at $[dy, dx]$, where dy is the physical distance between the neurons (the distance on the grid) and dx the distance between their weight vectors. In a well topology-preserving mapping, dy should be proportional to dx, at least for small dy's. This representation is directly usable in the frame of CCA: dx is simply equivalent to X_{ij} and dy corresponds to Y_{ij}.

In this representation, a locally correct mapping is shown by a straight line (of slope 1) near the origin. On the other hand, strong unfolding is revealed by curvature and spreading of the dy, dx characteristic. This is illustrated in figure 2.

4 Continuous projection and backward projection

The relation $\boldsymbol{x} \rightarrow \boldsymbol{y}$ is quantized be N prototypes ($\boldsymbol{x}_i \rightarrow \boldsymbol{y}_i$). To obtain the continuous projection of any point $\boldsymbol{x}_0$ in the distribution, the same function (1) is minimized, but only for point $\boldsymbol{y}_0$ that should be the projection of $\boldsymbol{x}_0$. Thus, instead of moving each vector with respect to each other, only one point $\boldsymbol{y}_0$ is adapted while keeping all others 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. This procedure gives very accurate interpolation, but also good extrapolation, which is a particularly new feature. This task is generally not or badly done by most networks whose target is to continuously link an output with an input space, and which generally perform interpolation only. For example, with RBF networks ("Radial Basis Functions", [13]), or some SOM implementations ([4, 3]) or with "Counter-Propagation network" ([10]), this interpolation is made by compu-

tation of a center of gravity weighted by kernel functions. Since these kernels are defined 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. In addition, the interpolation accuracy strongly depends on the kernel radii: too small radii give a projection where points are clustered around the $\boldsymbol{y}_i$'s, while too large kernels will cluster the whole distribution around the center of gravity of the $\boldsymbol{y}_i$'s. Thus, adaptation of these radii is a complex task which depends from several constraints and slows down the learning process.

In contrast, our method performs extrapolation as well as interpolation, as it is shown in figure 4. 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 projection needs several adaptation steps, and the parameters schedule during this "sub-convergence" is empiric up to now.

In order to obtain *backward projection*, input and output are simply swapped and the same scheme is used.

5 Discussion

The VQP tool can be seen as a complete homeomorphism between a data set structure and a map of this structure that has been found in a self-learning way. It reveals the structure of the underlying submanifold of a data set through a relevant map whose dimension is selectable (and has to be set accordingly to fractal dimension analysis of the data set and any initial knowledge about the underlying process).

Finding structure in high dimensional distributions and mapping them towards lower dimensional space are very generic tasks. Up to now, we have successfully applied this Curvilinear Component Analysis to a large area of applications, including:

- **Data fusion.** Audition and vision fusion for phonemes mapping, localization tasks (digitizer, GPS, ...), time series learning, super-resolution algorithms, fault detection in circuits and machines, geopolitic analysis, textures mapping.
- **Process monitoring.** State graph extraction, nuclear plant monitoring.
- **Construction of metrics.** Adaptive packet routing in ATM telecommunication network, graph matching, knowledge representation and concept mapping.

In fact, these few examples are only tracks for real applications, they show the large number of potential applications of this data analysis method.

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] F. Blayo and P. Demartines. Data analysis: How to compare Kohonen neural networks to other techniques ? In A. Prieto, editor, *International Workshop on Artificial Neural Networks*, volume 540 of *Lecture Notes in Computer Science*, pages 469–476. Springer-Verlag, 1991.

[3] Y. Cheneval, P. Demartines, and L. Tettoni. Function approximation using an architecture based on Kohonen self-organizing maps. In *Journées Neurosciences et Sciences de l'Ingénieur*, Oléron, May 1992.

[4] V. Cherkassky and H. Lari-Najafi. Constrained topological mapping for nonparametric regression analysis. *Neural Networks*, 4:27–40, 1991.

[5] 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.

[6] P. Demartines. Organization measures and representations of Kohonen maps. In J. Hérault, editor, *First IFIP Working Group 10.6 Workshop*, 1992.

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

[8] P. Demartines and J. Hérault. Representation of nonlinear data structures through fast VQP neural network. In *Neuronîmes*, pages 411–424, October 1993.

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

[10] R. Hecht-Nielsen. Counterpropagation networks. *Applied Optics*, 26(23):4979–4984, 1987.

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

[12] T. Kohonen. The self-organizing maps. *Proc. of the IEEE*, 78(9):1464–1480, 1990.

[13] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[14] H. Ritter. Parametrized self-organizing maps. In *IEEE International Conference on Artificial Neural Networks, Espoo, Finland*, Amsterdam, The Netherlands, September 1993.

[15] H. Ritter, T. Martinetz, and K. Schulten. *Neural computation and self-organizing maps: an introduction.* Addison-Wesley Publishing Company, 1992.

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

[17] C. von der Malsburg. Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, 14:85–100, 1973.