ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 13-20

A Robust Nonlinear Projection Method

John Aldo Lee[*], Amaury Lendasse, Nicolas Donckers[†], Michel Verleysen[‡]

Université catholique de Louvain
Place du Levant, 3, B-1348 Louvain-la-Neuve, Belgium
{lee,donckers,verleysen}@dice.ucl.ac.be, lendasse@auto.ucl.ac.be

Abstract. This paper describes a new nonlinear projection method. The aim is to design a user-friendly method, tentatively as easy to use as the linear PCA (Principal Component Analysis). The method is based on CCA (Curvilinear Component Analysis). This paper presents two improvements with respect to the original CCA: a better behavior in the projection of highly nonlinear databases (like spirals) and a complete automation in the choice of the parameters value.

1 Introduction

Often, one has to face the problem of dealing with huge numerical databases. These databases consist in numerous samples (or vectors) x_i^n defined in a n-dimensional space. One way to make huge databases more manageable is to project the databases in a low-dimensional space (say p-dimensional, $p < n$). A well known method to project such a database is the Principal Component Analysis (PCA).

PCA detects the linear dependencies between the coordinates (or features) of vectors x_i^n in the database $X^n \subset \mathbb{R}^n$. The main drawback of PCA holds in the fact that only linear dependencies are found. Nonlinear projection methods exist but their performances strongly depend on complex adjustment of some parameters. In comparison, the only parameter required by PCA is the loss of variance accepted when the database is projected. With this single parameter, PCA automatically determines the dimension p of the space where to project and gives the best LMS projection of the database. It would be nice to have such a user-friendly projection method, but with nonlinear capabilities. To reach this goal, the CCA algorithm [1, 2, 3], described below, is a promising approach.

[*] This work was realized with the support of the 'Ministère de la Région wallonne', under the 'Programme de Formation et d'Impulsion à la Recherche Scientifique et Technologique'

[†] N.D. is working towards the Ph.D. degree under FRIA fellowship.

[‡] M.V. works as a research associate of the Belgian FNRS.

2 The original CCA

The basic idea behind CCA lies far from the PCA. In a few words, CCA tries to reproduce the database 'topology' from the initial space $\mathbb{R}^n$ to the projection space $\mathbb{R}^p$. In CCA spirit, the word 'topology' means 'the distances between all pairs of vectors in the database'. Actually, CCA simply tries to find vectors x_i^p in the projection space $\mathbb{R}^p$ such that they reproduce the distances measured in the initial space $\mathbb{R}^n$.

Formally, CCA works by minimizing an objective (or error) function, which is simply a kind of 'topology difference' between vectors x_i^n and x_i^p :

$$E_{CCA} = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=0}^{N} (d_{i,j}^n - d_{i,j}^p)^2 F(d_{i,j}^p), \qquad (1)$$

where N is the number of samples in the database X^n. The function $d_{i,j}$ measures the Euclidian distance between vectors x_i and x_j, either in $\mathbb{R}^n$ $(d_{i,j}^n)$ or in $\mathbb{R}^p$ $(d_{i,j}^p)$; F is a decreasing function of $d_{i,j}^p$.

Which role does the function F play? Actually, when the dependencies in the database are not linear, a perfect reproduction of all distances is not possible. In this case, F acts as a weighting function giving more importance to the reproduction of small distances. In other words, CCA tries above all to preserve the 'local topology' of the database, reproducing 'global topology' only when it is possible.

At first glance, CCA looks like Sammon's [4] mapping or nonlinear MDS [5]. However, some differences in the objective function makes CCA more powerful in real cases. More details and a comparison between the three methods can be found in [1].

3 How to implement CCA?

CCA is implemented by a modified stochastic gradient descent applied to the objective function E_{CCA}:

$$\forall i \neq j, \qquad \Delta x_i^p = \alpha(t) \frac{d_{i,j}^n - d_{i,j}^p}{d_{i,j}^p} \, u\left(\lambda(t) \max_{i,j}(d_{i,j}^p) - d_{i,j}^p\right) (x_j^p - x_i^p), \qquad (2)$$

where $\alpha(t)$ (learning factor) and $\lambda(t)$ (neighborhood factor) are time decreasing parameters, with values between 0 and 1. Also note that $u(.)$ is the step function, standing as a special instance of F, parameterized by $\lambda(t)$ (for the reasons of this choice, see [1]).

At this point, a practical problem arises: each iteration of the gradient descent has a computational cost which is proportional to N^2! Therefore, the convergence for large databases is time-consuming.

This problem is solved by using vector quantization (VQ) before applying CCA, in order to obtain a smaller set of vectors, called 'centroids'. Incidentally, VQ will also smooth the hyper-surface if it is noisy.

At this point, applying criterion (1) only to centroids leads to a mapping between the positions of the centroids in the two spaces, but not really a projection of the whole database X^n. Therefore, an additional procedure has to be designed to perform an interpolation between centroids. This interpolation uses the same adaptation rule (eq. 2) as for the main algorithm, but considering now that only the vector to project will be adapted, while the position of all other centroids is frozen(more details in [1]).

4 Choosing the parameters for CCA

CCA requires three parameters: first the projection space dimension (p), and secondly the two time decreasing parameters ($\alpha(t)$ and $\lambda(t)$) used by the adaptation rule.

Dimension p can be computed by fractal dimension estimation [6] or by LPCA (see below). The learning factor $\alpha(t)$ requires no particular attention (for example, exponential decrease between 0.95 to 0.01). On the other hand, the neighborhood factor $\lambda(t)$ is critical: if $\lambda(t)$ decreases too slowly, the nonlinear dependencies are not well unfolded, whereas a fast decrease compromises the convergence. Although CCA outperforms many other nonlinear projection algorithms, this dependency on critical parameters (and on the density of samples) raises difficulties to unfold 'hard nonlinear' structures like a spiral. In such a case, CCA converges slowly: large distances $d^n_{i,j}$ in the initial structure are poorly correlated with the corresponding distances $d^p_{i,j}$ in the perfectly unfolded and projected structure.

5 How to improve the basic idea of CCA?

Looking at the example of the spiral once again, CCA globally remains a good idea, but perhaps the use of another distance than the Euclidian one could improve the convergence. The best distance function should produce the same result for both the initial spiral and its projection. To reach this goal, one needs a kind of 'curvilinear distance' $\delta^n_{i,j}$, like in Fig. 1c. Such a distance[1] is computed *inside* the spiral and not *through* the spiral, like the Euclidian distance.

An approximation of the curvilinear distance can be computed in two steps (see Fig. 2):

Step 1: Linking the centroids. After the vector quantization, the centroids can be linked (or connected) so that they become a graph. Two centroids get linked when they are the nearest ones from a database vector. This idea of linking centroids is not new[2]. In CCA, the first utility of links is visual: for example, crossing links often means projection faults.

[1] in accordance with the idea of sparse distance matrix suggested in [3].

[2] See for example the work of Bernd Fritzke [7]

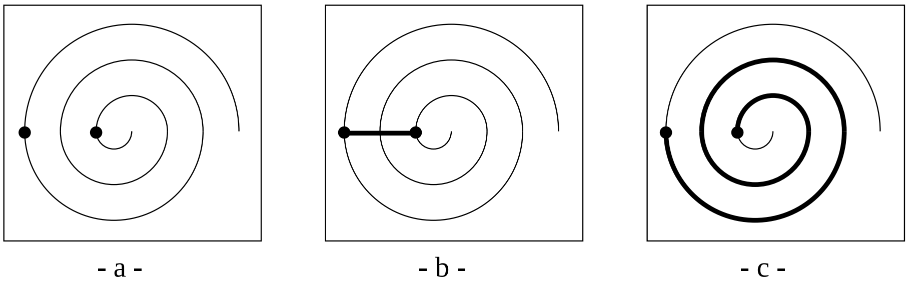

Figure 1: The 'curvilinear distance': (-a-) two points in a spiral, (-b-) the Euclidian distance between the two same points and (-c-) the curvilinear distance.

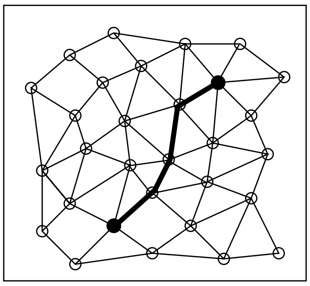

Figure 2: Approximation of the 'curvilinear distance' by means of the shortest path *via* the links between centroids (here the distance between both blackened centroids).

Step 2: Computing a distance *via* the links. Links also have a second utility: they help to compute the abovementioned curvilinear distance. A good approximation of $\delta^n_{i,j}$ is given by the sum of the Euclidian lengths of all links in the shortest path from centroid i to centroid j, provided there are no 'shortcut' links.

5.1 The Curvilinear Distances Analysis

We propose an enhanced version of Demartines' CCA, called CDA (Curvilinear Distances Analysis). The objective function remains identical, but the Euclidian distance $d^n_{i,j}$ is replaced by the curvilinear distance $\delta^n_{i,j}$:

$$E_{CDA} = \sum_{i=1}^{N} \sum_{j=0}^{N} (\delta^n_{i,j} - d^p_{i,j})^2 F(d^p_{i,j}). \qquad (3)$$

This objective function gives a new adaptation rule:

$$\forall i \neq j, \qquad \Delta x_i^p = \alpha(t) \frac{D_{i,j}^n - d_{i,j}^p}{d_{i,j}^p} \, u\left(\lambda(t) \max_{i,j}(d_{i,j}^p) - d_{i,j}^p\right)(x_j^p - x_i^p) \qquad (4)$$

where $D_{i,j}^n$ is a generalized distance between centroids i and j in the n-dimensional space:

$$D_{i,j}^n = (1 - \omega(t)) d_{i,j}^n + \omega(t) \delta_{i,j}^n. \qquad (5)$$

The generalized distance $D_{i,j}^n$ helps to build a unique algorithm that combines the Euclidian distance and the curvilinear one; the result is an algorithm with a third parameter $\omega(t)$, varying between 0 and 1, and allowing to dynamically switch between classical CCA and CDA.

6 Automatic choice of the parameters

The second improvement brought to CCA is the complete automation for the choice of parameters. Remember that the goal is to get a method as simple as PCA, i.e. a method with a single parameter l (the acceptable loss in the database variance).

Parameters for vector quantization. Usually, VQ requires two parameters: the number of centroids and a learning factor $\alpha(t)$. Instead, we use a dynamic VQ in which centroids are created when all available ones lie further from a sample than a fixed threshold r. Of course, smaller is the threshold r, better is VQ quality, so r can be made proportional to the tolerable loss l:

$$r = l \max_{i,j} d_{i,j}^n. \qquad (6)$$

Parameters for CDA. CDA requires four parameters: first the projection space dimension (p), and then the three parameters used by the adaptation rule ($\alpha(t)$, $\lambda(t)$, $\omega(t)$).

The optimal dimension p of the space where to project is easily determined by a method called LPCA (local PCA [8]). LPCA works by performing a vector quantization[3] and then a PCA on each Voronoi region, assuming that the database is linear locally (i.e. at the scale of the Voronoi regions). Given the tolerable loss l, the required dimension p_{V_i} of the projection space is computed for each Voronoi region V_i, according to PCA standard procedure; the global p is simply the average of all p_{V_i}. Obviously, no mathematical proof guarantees that p will lead to an effective loss smaller than l; however, we assume (and verify experimentally) that, in the scope of a Voronoi region, CDA works at least as well as PCA.

[3] Nothing prevents to use the same vector quantization for CDA and for LPCA!

F or the learning factor$\alpha(t)$ and the neighborhood factor $\lambda(t)$, one uses the classical exponential decrease (as in numerous adaptative algorithms), within the following arbitrary chosen bounds:

$$\begin{aligned} 1.0 &\geq \alpha(t) \geq & 0.02, \qquad (7)\\ 1.0 &\geq \lambda(t) \geq & \frac{\min_{i,j} \delta^n_{i,j}}{\max_{i,j} \delta^n_{i,j}}. \qquad (8) \end{aligned}$$

In Demartines' CCA, the choice of the bounds for $\lambda(t)$ is quite difficult. The CDA method is less sensitiv eto the choice of $\lambda(t)$, due to the enhancement brought by the curvilinear distance: the curvilinear distance fastens and simplifies the conv ergence.

Finally, how to determine the last parameter $\omega(t)$? In tuitiv ely, if the dependencies in the database are linear, $\omega(t)$ should be set near zero since the classical CCA works perfectly in this case. In the same way, if the dependencies are strongly nonlinear, a value for $\omega(t)$ close to one should take profitof the curvilinear distances. With this reasoning in mind, the value of $\omega(t)$ is empirically computed as a function of the quotients $d^n_{i,j}/\delta^n_{i,j}$, measuring the linearity of the database:

$$\mathbf{D}(t) = \left\{ (i,j) \mid \delta^n_{i,j} \leq \lambda(t) \max_{i,j} \delta^n_{i,j} \right\}, \qquad (9)$$

$$\omega(t) = \min \left\{ 1, \frac{\pi}{\pi - 2\sqrt{2}} \left(1 - \frac{1}{|\mathbf{D}(t)|} \sum_{(i,j)\in\mathbf{D}(t)} \frac{d^n_{i,j}}{\delta^n_{i,j}} \right) \right\}, \qquad (10)$$

where $\mathbf{D}(t)$ is simply a subset of all pairs (i,j).

7 Some illustrative examples

This section shows some artificial databases projected with the CDA algorithm. Their purpose is only illustrative: muc h more complex structures can be successfully handled by CDA, but their visual aspect is less meaningful.

Although the implementation allows to tune the parameters, all examples below are projected by the automatic method. All figures below include some v ectors of the database (shown as points), all cen troids (circles) and links (lines). Centroids and links are not shown for the projections of the knot and the sphere.

The horseshoe (Fig. 3) is a tw o-dimensional rectangle embedded in a three-dimensional space, slightly curved to obtain three quarters of a cylinder. The horseshoe is a classical benchmark for nonlinear projection methods.

The trefoil knot (Fig. 4) is a mono-dimensional object embedded in a three-dimensional space. The CDA unties it in a mono-dimensional space rapidly and automatically.

The projection of the sphere (Fig. 5) is more complex. Indeed, a good projection in a plane requires that the algorithm cuts and stretches the sphere (if not, the projection would not be bijective).

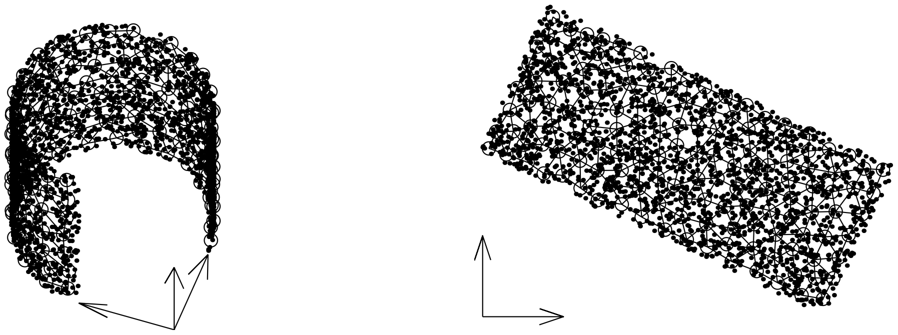

Figure 3: Nonlinear projection of the horseshoe (from dimension 3 to 2).

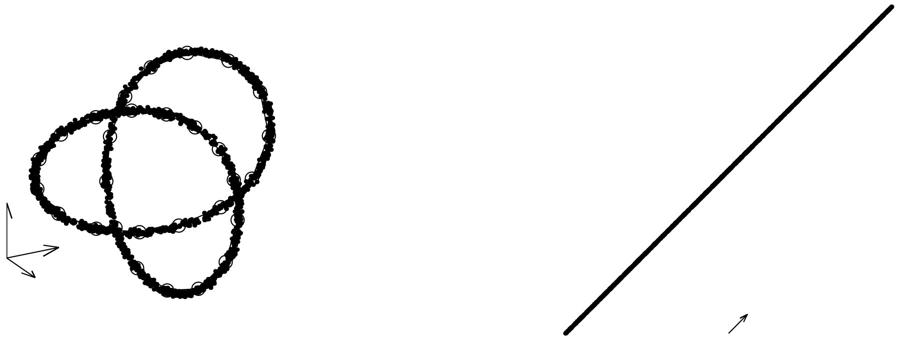

Figure 4: Nonlinear projection of a trefoil knot (from dimension 3 to 1).

8 Conclusion

The CDA method nicely answers the problem of nonlinear projection. The whole process is as easy as PCA, but with the advantage of nonlinear capabilities. Thanks to the approximation of curvilinear distances, the CDA outperforms the classical CCA on strongly nonlinear databases. The CDA algorithm also shows more flexibility than the Kohonen self-organized maps (SOM, see for example [9]), which have a fixed shape.

Further work will focus on improvements on the interpolation algorithm and on the robustness against noise.

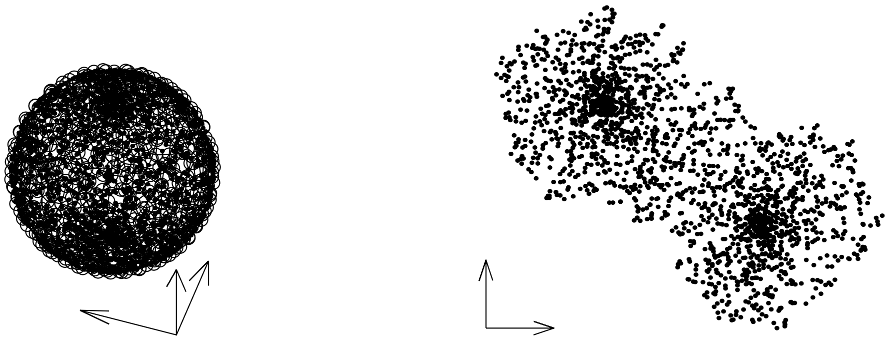

Figure 5: Nonlinear projection of the sphere (from dimension 3 to 2).

9 Acknowledgement

The authors would like to thank A. Guérin and J. Hérault for their appreciated comments.

References

[1] P. Demartines, J. Hérault, *Curvilinear Component Analysis: A self-organizing neural network for nonlinear mapping of data sets.* IEEE Transaction on Neural Networks 8: (1) 148-154, January 1997.

[2] J. Hérault *et al.*, *Curvilinear Component Analysis for High-Dimensional Data Representation: I. Theoretical Aspects and Practical Use in the Presence of Noise*, in J. Mira and J.V. Sánchez-Andrés (Eds.), Proc. IWANN'99, Alicante, Spain, Springer, 625-634, 1999.

[3] A. Guérin-Dugué *et al.*, *Curvilinear Component Analysis for High-Dimensional Data Representation: II. Examples of Additional Mapping Constraints in Specific Applications*, in J. Mira and J.V. Sánchez-Andrés (Eds.), Proc. IWANN'99, Alicante, Spain, Springer, 635-644, 1999.

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

[5] R.N. Sheppard, *Multidimensional scaling, tree-fitting and clustering.* Science, 210-390-398, 1980.

[6] P. Grassberger, I. Procaccia, *Measuring the Strangeness of Strange Attractors.* Physica D56, 189-208, 1983.

[7] B. Fritzke, *Growing Self-orginizing Networks - Why?*, in M. Verleysen (Ed.), ESANN'96, Brussels, Belgium, D-Facto publishers, 61-72, 1996.

[8] N. Kambhatla, T.K. Leen, *Dimension reduction by local principal component analysis* Neural Computation, 9: (7) 1493-1516, October 1997.

[9] T. Kohonen, *Self-Organizing Maps*, Second Extended Edition. Springer Series in Information Sciences, Vol. 30, Springer, Berlin, Heidelberg, New York, 1995, 1997.