On the use of self-organizing maps to accelerate vector quantization

Eric de Bodt[1], Marie Cottrell[2], Patrick Letremy[2], Michel Verleysen[3]

[1] Université catholique de Louvain, IAG-FIN, 1 pl. des Doyens,
B-1348 Louvain-la-Neuve, Belgium
and
Université Lille 2, ESA, Place Deliot, BP 381,
F-59020 Lille, France

[2] Université Paris I, SAMOS-MATISSE, 90 rue de Tolbiac,
F-75634 Paris Cedex 13, France

[3] Université catholique de Louvain, DICE, 3 pl. du Levant,
B-1348 Louvain-la-Neuve, Belgium

Abstract.
Self-organizing maps (SOM) are widely used for their topology preservation property: neighboring input vectors are quantified (or classified) either on the same location either on neighboring ones on a predefined grid. But SOM are also widely used for their more classical vector quantization property. We show in this paper that using SOM instead of the more classical Simple Competitive Learning (SCL) algorithm drastically increases the speed of convergence of the vector quantization process. We also suggest to use the result of SOM as initial conditions for the SCL algorithm, in order to benefit both from the increased convergence speed and the convergence towards optimal states.

1. Vector Quantization

Vector quantization (VQ) is a widely used tool in many domains of data analysis. It consists in replacing a continuous distribution by a finite set of quantifiers, while minimizing a predefined distortion criterion. Vector quantization may be used in clustering or classification tasks, where the aim is determine groups (clusters) of data sharing common properties. It can also be used in data compression, where the aim is to replace the initial data by a finite set of quantified ones; labeling the quantified set and using the labels rather than the data themselves makes compression possible. Vector quantization is basically an unsupervised process, while supervised variants exist (LVQ1, LVQ2, in Kohonen [5]); in this last case, the *distortion criterion* takes class labels into account.

Most of the methods used to perform VQ necessitate setting a priori the number of clusters or quantifiers. The choice of this number results from a trade-off between the precision (distortion) of the quantization and the necessity of an efficient description of the resulting clusters (quantity of information kept after quantization).

Once the number of quantifiers is predefined, a good criterion of the quality of the classification is the *distortion* which measures the deviation between the data and the corresponding quantifiers. Let us recall the main definitions and notations.

Let us consider a continuous data space Ω, of dimension d, endowed by a continuous probability density function (pdf) $f(x)$, where the cumulated density (or repartition function) is $F(x)=P(X<x)$ (where P is the probability law, and where the inequality is verified in each dimension).

A vector quantization Φ is an application from the continuous space Ω to a finite subset F (the *codebook*) formed by n *code-vectors* or *centroids* or *quantifiers* $q_1, q_2,\ldots,q_n$ of Ω. The positions of the code-vectors are supposed to be computed as a result of a *quantization algorithm* or *learning algorithm*.

The aim of a *vector quantization* (VQ) is to compress the information by replacing all elements x of a cluster C_i (subset of Ω) by a unique quantifier (or code-vector, or centroid) q_i. For a given number n of code-vectors, vector quantization tries to minimize the loss of information or *distortion*, measured by the mean quadratic error

$$x(f,\Phi) = x(f,q_1,q_2,\ldots,q_n) = \sum_{i=1}^{n} \int_{C_i} \|x - q_i\|^2 \, f(x)dx \,. \quad (1)$$

If a N-sample $x_1, x_2,\ldots,x_N$ is available (randomly chosen according to $f(x)$), this distortion is estimated by the intra-class sum of squares

$$\hat{x}(f,\Phi) = \hat{x}(f,q_1,q_2,\ldots,q_n) = \frac{1}{N} \sum_{i=1}^{n} \sum_{x_j \in C_i} \|x_j - q_i\|^2 \,. \quad (2)$$

All classical VQ algorithms (LBG, k-means,…) minimize this distortion function by choosing appropriate centroid locations. See for example Anderberg [1] or Lebart et al. [7] for proofs. There is no unique minimum of the distortion function, and the result strongly depends on the initialization.

2. Simple Competitive Learning and batch VQ algorithms

There exist many algorithms that deal with the VQ problem. Most of them are very slow in terms of convergence speed. The most popular one is the so-called Simple Competitive Learning algorithm (SCL) which can be defined as follows (see for example [4]):

let Ω be the data space (with dimension d), endowed with a density probability function $f(x)$. The data are randomly drawn according to the density $f(x)$ and are denoted by $x_1, x_2,\ldots,x_N,$. The number of desired clusters is a priori fixed to be n. The quantifiers $q_1, q_2, \ldots, q_n$ are randomly initialized. At each step t,

- a data x_{t+1} is randomly drawn according to the density $f(x)$;
- the winning quantifier $q_{i^*(t)}$ is determined by minimizing the classical Euclidean norm
$$\| x_{t+1} - q_{i^*(t)} \| = \min_j \| x_{t+1} - q_j \| \,;$$
- the quantifier $q_{i^*(t)}$ is updated by $q_{i^*(t+1)} = q_{i^*(t)} + \varepsilon(t)\,(x_{t+1} - q_{i^*(t)})$.

where $\varepsilon(t)$ is an adaptation parameter which satisfies the classical Robbins-Monro conditions ($\Sigma\, \varepsilon(t) = \infty$ and $\Sigma\, \varepsilon^2(t) < \infty$).

The SCL algorithm is in fact the *stochastic or on-line* version of the Forgy algorithm (also called *moving centers algorithm*, LBG, Lloyd's algorithm). See for example [3], [8], [9]. In that version of the algorithm, the quantifiers are randomly initialized. At each step t, the clusters $C_1, C_2, \ldots, C_n$ are determined by putting in class C_i the data which are closer to q_i than to any other quantifier q_j. Then the mean values in each cluster are simultaneously computed and taken as new quantifiers, before repeating the process. The Forgy algorithm works off-line as a batch algorithm; it will be referred to as BVQ (for Batch VQ) in the following. It also exists an intermediate version of the algorithm, frequently named the K-means method [10]. In that case, at each step, only one data is randomly chosen, and only the winning quantifier is updated as the mean value of its class.

It can be proven ([1], [7]) and it is well-known that BVQ minimizes the *distortion* (1) or, more exactly, the estimated one (2). Note that the stochastic SCL algorithm also minimizes this distortion, but only in mean value: at each step, there is a positive probability to increase the distortion, as for any stochastic algorithm.

Let us denote by $q_1^*, q_2^*, \ldots, q_n^*$ one set of quantifiers which (locally) minimizes the distortion. Generally the minimum is not unique and depends on the initial values. At a local minimum of the distortion, each q_i^* is the center of gravity of its class C_i, with respect to the density f, and the quantifiers are fixed points of the BVQ algorithm. In an exact form,

$$q^*_i = \frac{\int_{C_i} x\, f(x)\, dx}{\int_{C_i} f(x)\, dx}, \quad (3)$$

estimated by

$$\hat{q}^*_i = \frac{\sum_{x_j \in C_i} x_j}{\sum_{x_j \in C_i} 1} \,. \quad (4)$$

Let us note that equations are implicit ones, since the C_i are defined according to the positions of the q_i.

For example, in the one-dimensional case, the classes C_i ($1 \le i \le n$) are intervals defined by $C_i = [a_i, b_i]$, with $a_i = ½ (q^*_{i-1} + q^*_i)$ and $b_i = ½ (q^*_{i+1} + q^*_i)$, for $1 < i < n$, and $a_1 = \inf(\Omega)$, $b_n = \sup(\Omega)$.

The BVQ algorithm is nothing else than the iterative computation of the solutions of equations (3) or (4). The solution is not unique, and depends on the initial values.

The goal of this paper (after a preliminary work presented to ESANN'99, see [2]) is to evaluate the speed of convergence of VQ algorithms. In situations where the solution is unique and where it is possible to compute the exact values q^*_i, the performances will be evaluated through the rate at which the values q_i converge to q^*_i (see section 4).

3. Self-Organizing Maps (SOM)

Let us consider now the SOM algorithm (as defined by T. Kohonen in [5]) which can be seen as an extension of the Simple Competitive Learning Algorithm (SCL) in its classical stochastic form, and of the Forgy Algorithm (BVQ) in its batch form.

In this section, we consider the SOM algorithm with a fixed number of neighbors (although the number of neighbors uses to decrease with time in practical implementations).

Actually the SOM algorithm is different from the SCL algorithm only because a neighborhood structure is defined between the n quantifiers. The neighborhood structure of the SOM algorithm is mainly used for visualization and data interpretation properties.

For a given neighborhood structure, where $V(i)$ denotes the neighborhood of unit i, the SOM algorithm is defined as follows. The quantifiers $q_1, q_2, \ldots, q_n$ are randomly initialized. At each step t,

- a data x_{t+1} is randomly drawn according to the density $f(x)$;
- the winning quantifier $q_{i^*(t)}$ is determined by minimizing the classical Euclidean norm
$$\| x_{t+1} - q_{i^*(t)} \| = \min_j \| x_{t+1} - q_j \| ;$$
- the quantifier $q_{i^*(t)}$ and its neighbors $q_{j^*}(t)$ for j in $V(i)$ are updated by
$$q_{j^*(t+1)} = q_{j^*(t)} + \varepsilon(t) (x_{t+1} - q_{j^*(t)}).$$

where $\varepsilon(t)$ is an adaptation parameter which satisfies the classical Robbins-Monro conditions ($\Sigma\, \varepsilon(t) = \infty$ and $\Sigma\, \varepsilon^2(t) < \infty$).

We see that the SCL algorithm is a particular case of the SOM algorithm, when the neighborhood is reduced to zero. Sometimes SCL is called *0-neighbor Kohonen algorithm*. As to the batch SOM algorithm, it is also similar to the Forgy (BVQ) algorithm. The only difference is that at each step, for a given set of classes $C_1, C_2, \ldots, C_n$, the quantifier q_i is set to the mean value of the union of the class C_i and its neighbors. See [6] for example.

In the one-dimensional case, and for a one-dimensional structure of neighborhood, if the neighborhood $V(i)$ contains i-1, i, i+1 (two-neighbor case), the limit points q^*_i of the batch SOM algorithm verify equation (3) or equation (4), where C_i is replaced by $C_i^2 = C_{i-1} \cup C_i \cup C_{i+1} = [a_i, b_i]$, with $a_i = ½ (q^*_{i-2} + q^*_{i-1})$ and $b_i = ½ (q^*_{i+1} + q^*_{i+2})$, for $2 < i < n$-1, and $a_1 = a_2 = \inf(\Omega)$, and $b_{n-1} = b_n = \sup(\Omega)$.

Here again, the batch SOM algorithm is nothing else than the iterative computation of the solutions of equations (3) or (4), when C_i is replaced by C_i^2.

The batch SOM algorithm and the classical stochastic SOM algorithm do not decrease anymore the distortion (1), but the generalized distortion [11], that is the distortion extended to the neighbor classes (as long as the number of neighbors n remains fixed). This generalized distortion is given by

$$x_v(f, \Phi) = x_n(f, q_1, q_2, \ldots, q_n) = \sum_{i=1}^{n} \int_{\bigcup_{k \in V(i)} C_k} \left\| x - q_i \right\|^2 f(x) dx , \qquad (5)$$

where $V(i)$ is the set of indexes in the neighborhood of i, including i. This generalized distortion function can also be estimated through a finite set of samples $x_1, x_2, \ldots, x_N$, similarly to (2).

4. Experimental results: convergence to the exact solution of the VQ problem

The SOM algorithm is not equivalent to the SCL algorithm: it is deemed to minimize the generalized distortion (5), and not the classical distortion of VQ problems (1). Despite this fact, we will show that the SOM algorithms performs better than the classical SCL algorithm, i.e. converges faster towards the solution of (1), at least during the first iterations of the algorithm.

We study this phenomenon from two points of view. First, in some cases where it is possible to exactly compute the solutions of equations (3) or (4), we evaluate the error between the current values and the optimal values as a function of the number of iterations, for both the SCL and SOM algorithms; this is the topic of this section. Secondly, for more realistic data, we compare the decreasing slope of the true distortion (1) as a function of the number of iterations, also for both algorithms; this is the topic of section 6

In some one-dimensional cases ($d = 1$), if the set Ω is a real interval, and if the density f is known and well-behaved, it is possible to directly compute the solutions q^*_i , starting from a given set of increasing initial values, by an iterative procedure.

If the initial values are ordered, the current values $q_1, q_2, \ldots, q_n$ will remain ordered at each iteration of the SCL algorithm. As mentioned in the previous section, the classes C_i ($1 \le i \le n$) are therefore intervals defined by $C_i = [a_i, b_i]$, with $a_i = ½ (q_{i-1} + q_i)$ and $b_i = ½ (q_{i+1} + q_i)$, for $1 < i < n$, and $a_1 = \inf(\Omega)$, $b_n = \sup(\Omega)$. This constitutes the first set of equations (C_i as a function of q_i) used in this iterative procedure.

Equations (3) or (4) have no explicit solutions. However, it is possible to compute analytically the solutions q^*_i, as a function of the limits a_i and b_i of the intervals C_i, for some "easy" densities $f(x)$. This will constitute the second set of equations (equivalent to (3)) used in the iterative procedure. Table 1 presents these recurrent explicit equations for the densities $f(x) = 2x$, $3x^2$, e^{-x}.

Density f	Distribution Function	a_0	b_0	q_i
$2x$ on [0,1]	x^2	0	1	$q_i = \frac{2}{3}\frac{b_i^3 - a_i^3}{b_i^2 - a_i^2}$
$3x^2$ on [0,1]	x^3	0	1	$q_i = \frac{3}{4}\frac{b_i^4 - a_i^4}{b_i^3 - a_i^3}$
e^{-x} on [0,+∞[	$1 - e^{-x}$	0	+∞	$q_i = \frac{a_i e^{-a_i} + e^{-a_i} - b_i e^{-b_i} - e^{-b_i}}{e^{-a_i} - e^{-b_i}}$

Table 1 : Exact computation of the quantifiers as a function of the limits of the clusters, for some "easy" examples of densities.

This iterative procedure is similar to the BVQ algorithm; formulas in Table 1 are the analytical solution of equation (3), while the BVQ algorithm usually involves in practical experiments the use of approximation (4).

Knowing the optimal values of the quantifiers, it is be possible to study the speed of convergence of any Vector Quantization algorithm. In that purpose, we will study the Euclidean distance between the current values of ($q_i(t)$) resulting from some VQ algorithm and the solutions (q^*_i), as a function of the numbers of iterations. We define the mean quadratic error

$$D^2(t) = D^2(q(t),q^*) = (1/n)\ \Sigma_{1 \le i \le n}\ (q_i(t) - q^*_i)^2 \qquad (6)$$

which will be the *error measure* of the Vector Quantization algorithm into consideration.

In practical situations, one can observe that the error measure $D^2(t)$ decreases to 0 very slowly when using the SCL algorithm. Note that in all simulations we carefully start from the same initial increasing points, including for the exact computation of the (q^*_i), in order to avoid any effect due to the initial conditions.

In Figures 1, 2 and 3, we represent the variations of the error measure $D^2(t)$, for the SCL algorithm and for the SOM with 2, 4 and 8 neighbors. Figures 1, 2 and 3 respectively concern densities $f(x) = 2x$, $3x^2$ and e^{-x}. We can see for example that the SOM with neighbors decreases to the optimal values (q^*_i) much faster than the SCL algorithm, even if it finally converges to its own optimal points as detailed above in section 3. These optimal points minimize the generalized distortion extended to the neighbors (5), and are different from the (q^*_i).

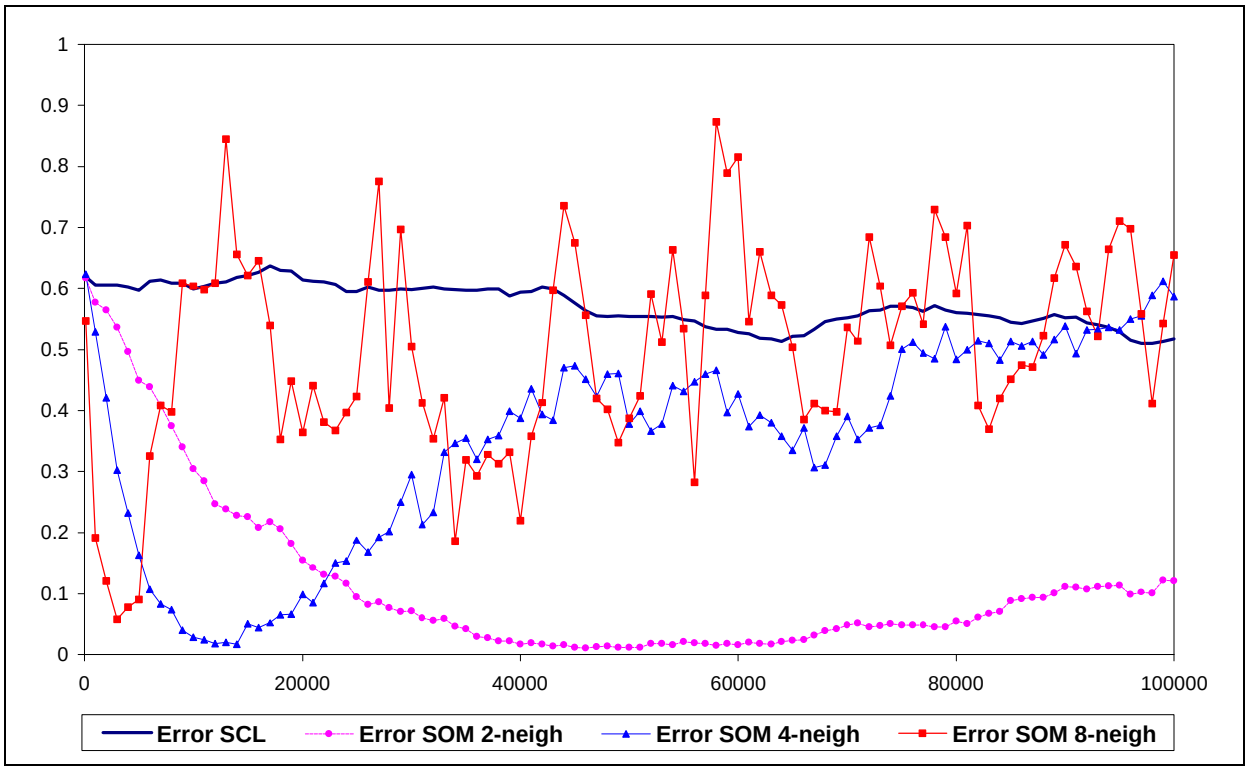

Figure 1 : Evolution of $D^2(t)$ as a function of the number of iterations, for the density $f(x) = 2x$.

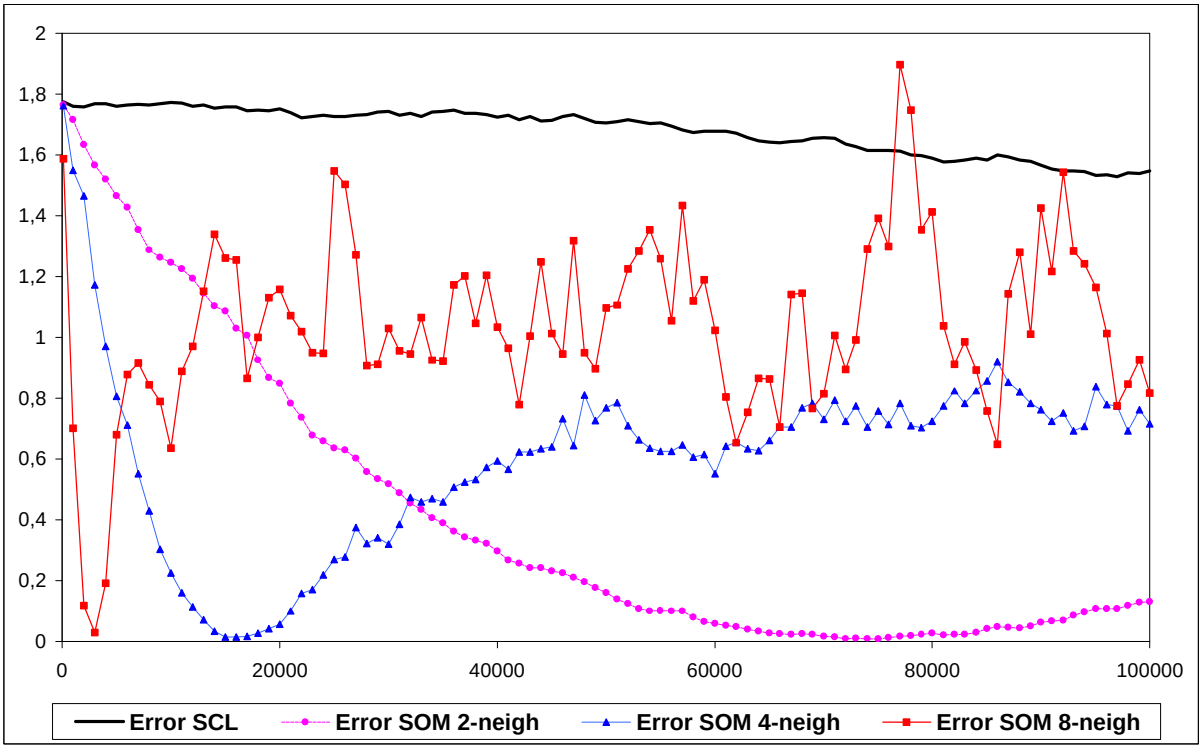

Figure 2 : Evolution of $D^2(t)$ as a function of the number of iterations, for the density $f(x) = 3x^2$

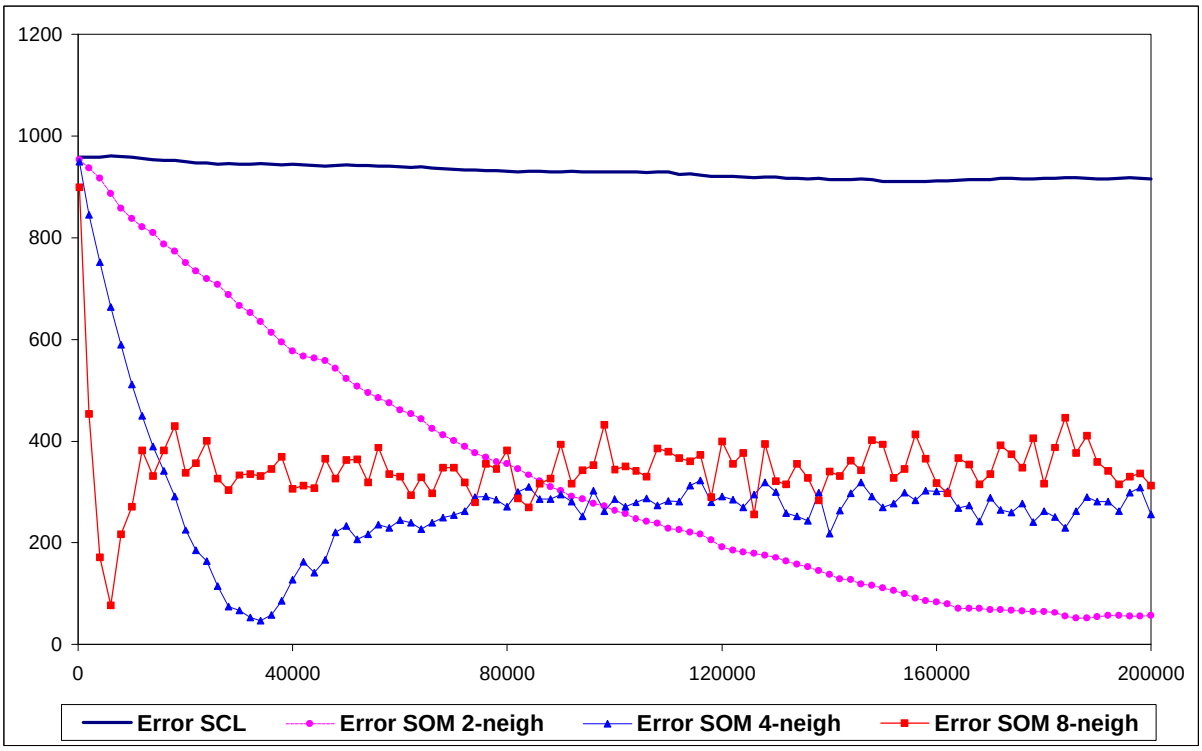

Figure 3 : Evolution of $D^2(t)$ as a function of the number of iterations, for the density $f(x) = e^{-x}$

We also measured the evolution of $D^2(t)$ as a function of the number of iterations, for a Gaussian density $N(0,1)$. In this case, the exact values q^*_i have been obtained through equation (4), by using very large samples to compute at each step the C_i. Figure 4 shows this evolution of $D^2(t)$ as a function of the number of iterations, respectively for the SCL algorithm and for the SOM with 2, 4, 8 and 16 neighbors.

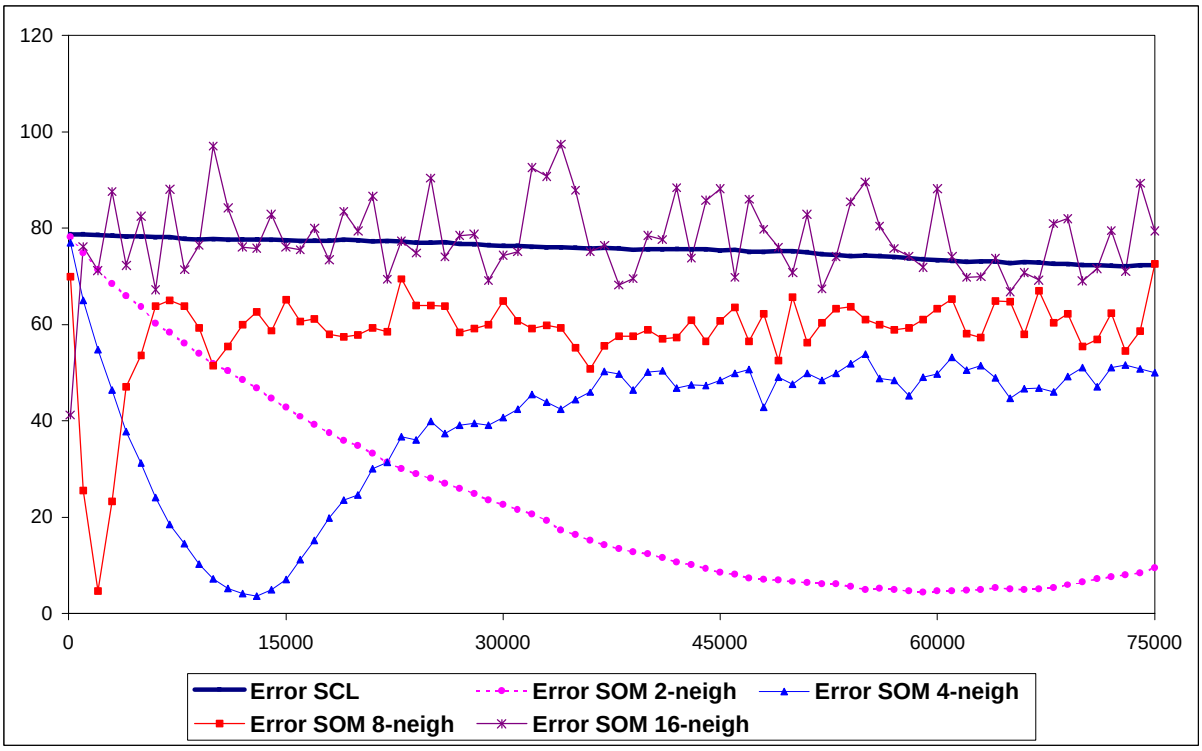

Figure 4 : Evolution of $D^2(t)$ as a function of the number of iterations, for the standard Gaussian $N(0,1)$ density

One could argue that the comparisons are made on different algorithms, where the processing time per iteration is different. The comparison in terms of number of iterations is thus not fair if the total processing time is searched for. Nevertheless, as an example, the difference of the processing time of one iteration when using the 2-neighbors SOM algorithm instead of the SCL algorithm is significantly less than 1%; the differences shown in Figures 1 to 4 thus remain significant.

This section has shown that the use of SOM can greatly increase the speed of convergence towards the exact solutions of the VQ problem. Nevertheless, it must not be forgotten that the SOM algorithm will not finally converge to these solutions after a great number of iterations, but rather will converge to the solution of (5). In the next section, we then to use a mixed algorithm, beginning by some iterations of the SOM algorithm and ending with a classical SCL procedure, in order to benefit both from the accelerated convergence and from the convergence towards optimal states.

5. Hybrid algorithm SOM/SCL

Based on the results of the previous section, we propose to use a hybrid VQ algorithm (denoted by KSCL for Kohonen SCL), which consists in an initial phase (a SOM algorithm with n neighbors), followed by the classical SCL. We compare the value of the error $D^2(t)$ after the same number of iterations for KSCL and SCL.

For example, let us fix a total number of iterations T, the initial ordered points $q_1(0)$, $q_2(0)$, …, $q_n(0)$, a constant ε and various probability functions : $f(x) = 2x$ on [0,1], $f(x) = 3x^2$ on [0,1], $f(x) = e^{-x}$ on [0, +∞ [, the standard Gaussian N(0,1). Let us also consider the 2-neighbors SOM algorithm ($n = 2$).

In Figures 5, 6, 7 and 8, we represent the evolution of the error measure for different KSCL algorithms, for the four probability densities that we took as examples. We consider four KSCL variants where the 2-neighbors SOM algorithm is used respectively during 0%, 30%, 60%, 90% of the total number of iterations T.

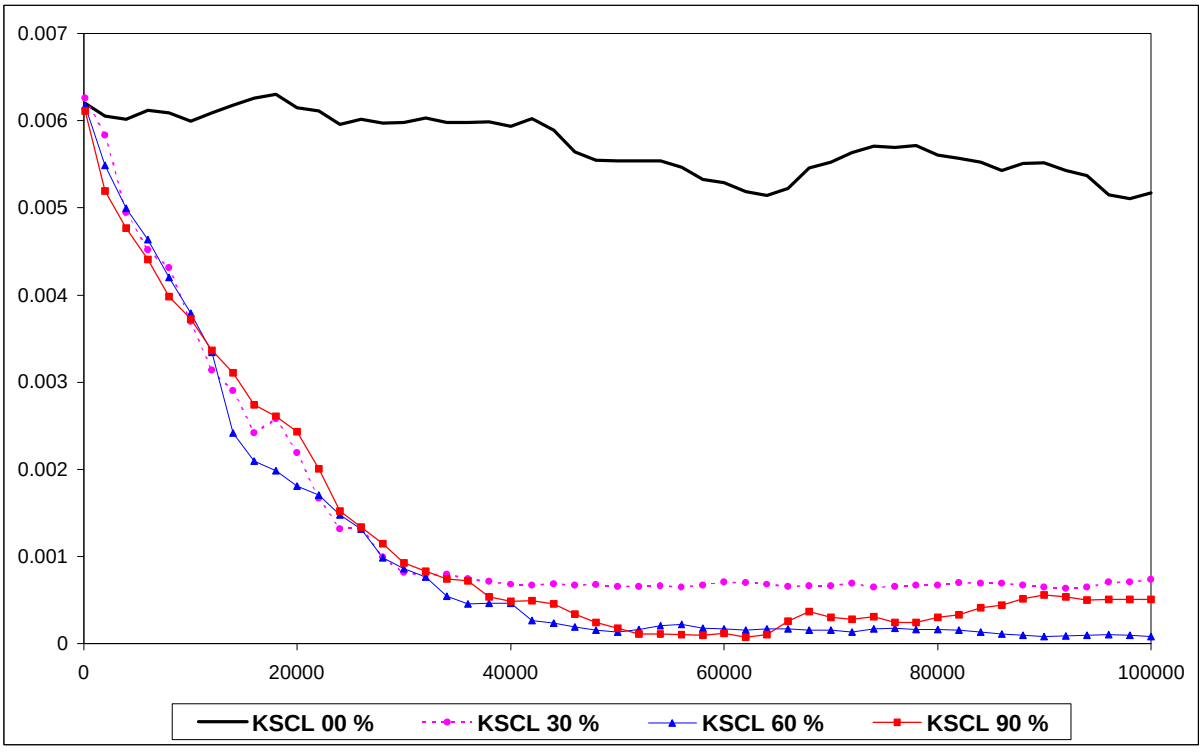

Figure 5 : Evolution of $D^2(t)$ as a function of the number of iterations, for 4 variants of the KSCL algorithm, on the $2x$ density.

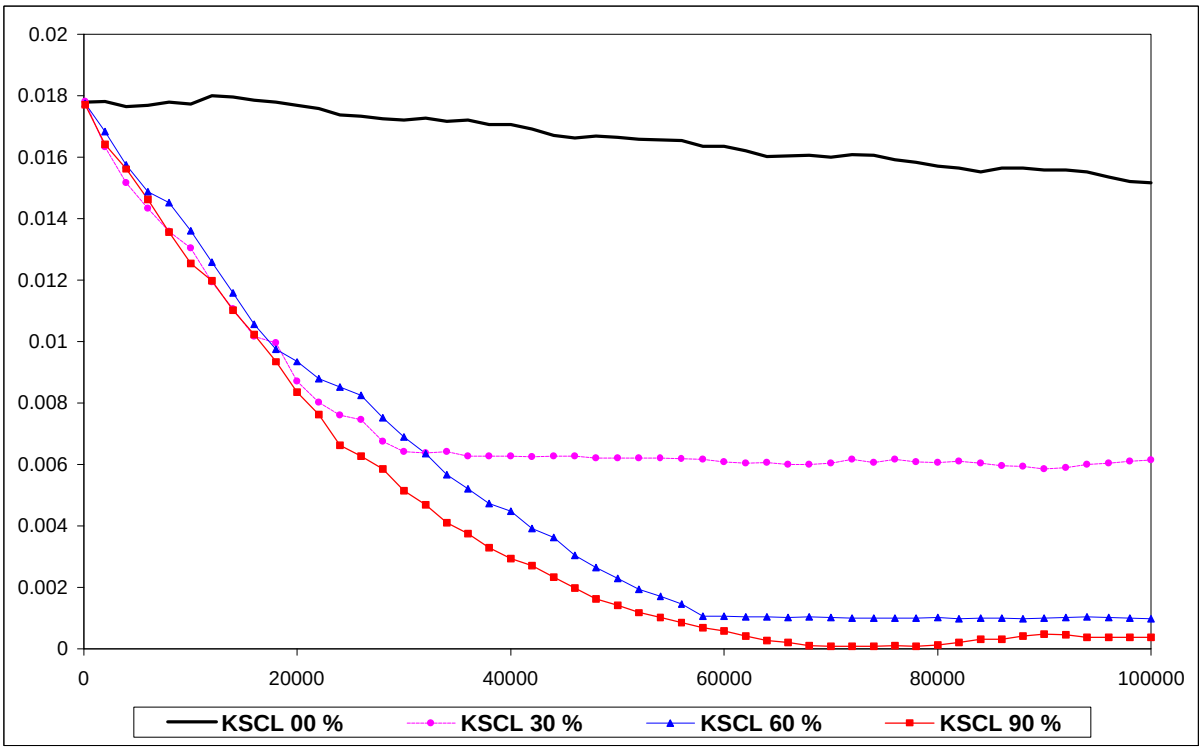

Figure 6 : Evolution of $D^2(t)$ as a function of the number of iterations, for 4 variants of the KSCL algorithm, on the $3x^2$ density.

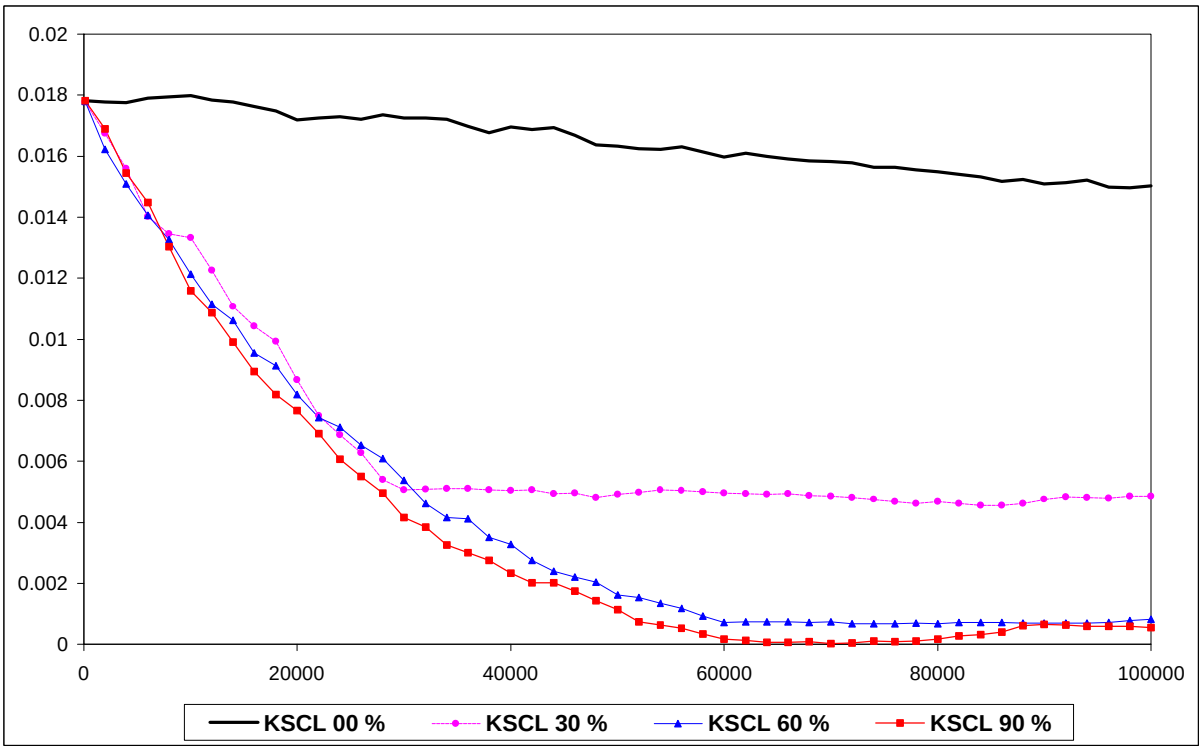

Figure 7 : Evolution of $D^2(t)$ as a function of the number of iterations, for 4 variants of the KSCL algorithm, on the e^{-x} density.

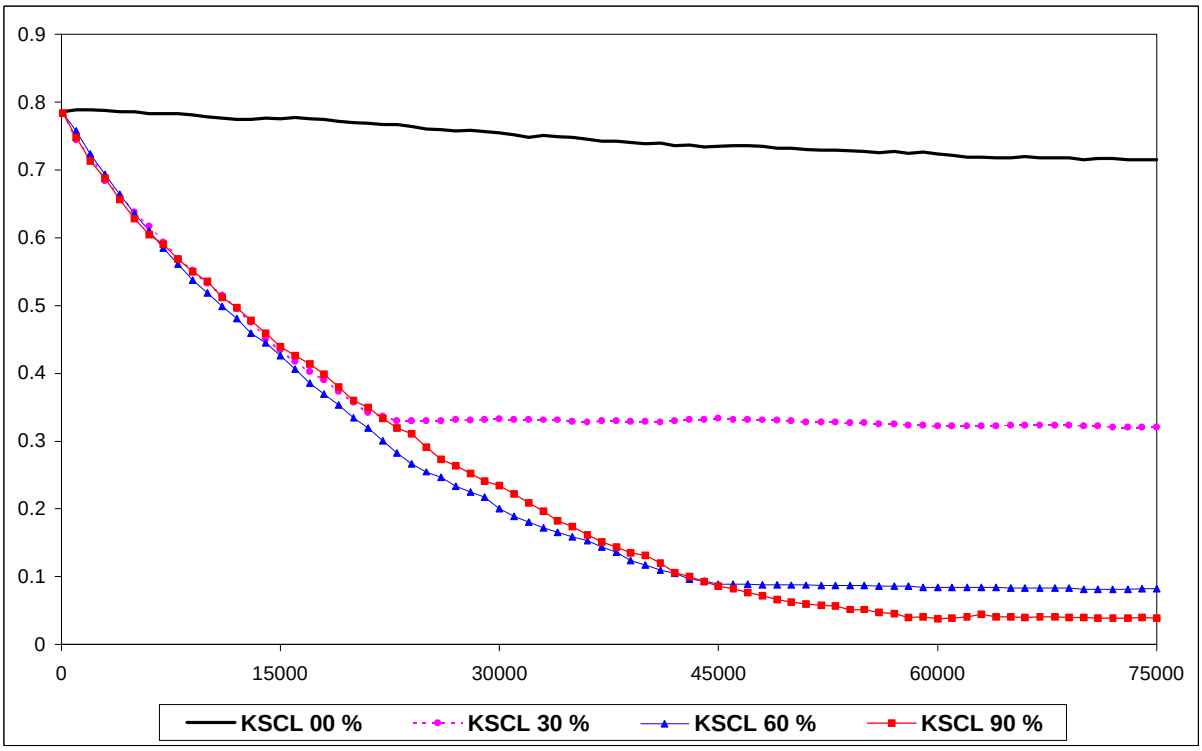

Figure 8 : Evolution of $D^2(t)$ as a function of the number of iterations, for 4 variants of the KSCL algorithm, on the standard Gaussian density.

We can observe in all simulations that the 2-neighbors algorithm greatly accelerates the decrease of the error measure in the beginning of the curves. In all cases, using too early the SCL algorithm slows down the decrease. Moreover, the performances remain better than those of the SCL algorithm, whatever is the choice of the KSCL variant. Nevertheless, it is also clear that determining the optimal iteration for substituting SOM by SCL strongly depends on the probability density. An optimal choice of this parameter would thus require extensive simulations, which is not the goal searched for here.

We may conclude this section by claiming that, in any case, *the SOM algorithm with a fixed number of neighbors can work as an efficient initialization of the SCL algorithm to accelerate the convergence and improve the performances.* We verified this statement on many other probability densities, real data and for several values of the number of neighbors *n.*

In practical implementations of the SOM algorithm, the number of neighbors is made decreasing. Our observations confirm that this widely used strategy is very efficient to improve the decreasing of the *error measure.*

It would be interesting to consider this so-called *error measure* in multidimensional settings. Nevertheless, this concept is not well suited to dimensions greater than 1. First, it becomes very hard to compute the optimal quantifiers q^*_i. Secondly, even an efficient (compared to others) VQ algorithm may converge slowly to the optimal quantifiers. The lack of ordering concept in dimension greater than 1 does not facilitate the problem either, and the correspondence between the current quantifiers at a given iteration and their optimal values looses its clear significance.

We will thus replace the concept of error measure by the concept of distortion as defined by (1) or (2). In the next section, we study how the distortion is decreasing along the quantization process, in both cases (SCL without neighbors, or SOM with a decreasing number of neighbors).

6. Experimental results: comparative evolution of the distortion

In this part, we study the vector quantization performances of SCL and, by computing the distortion defined in equation (2) in the case of real data.

As we mentioned previously, the SCL is supposed to minimize this distortion, while the SOM (with fixed or decreasing number of neighbors) is not. However we can observe that in any case, the SOM algorithm accelerates the decrease of the distortion, at least during a large part of the simulation.

We represent the distortion as a function of the number of iterations, for 5 different quantization algorithms: SCL and 4 variants of the SOM algorithm which differ by the number of neighbors. For a two-dimensional neighborhood structure, we consider successively 3 SOM algorithms with a fixed number of neighbors (SOM5, SOM9 and SOM25, the suffix being the number of neighbors) and then the classical SOM algorithm with a decreasing number of neighbors (from 25 to 1, the last part of the SOM iterations being equivalent to SCL).

We illustrate these simulations on two data set. The table SAVING contains 5 variables measuring economic ratios for 42 countries between 1960 and 1970; the table TOP500 contains 6 variables relative to 77 American companies in 1986. [1]

Figure 9 represents the distortion for the data SAVING, with 25 quantifiers, square grid (5 by 5) for SOM algorithms and 500 steps of iterations. Figure 10 represents the distortion for the data SAVING, with 100 quantifiers, square grid (10 by 10) for SOM algorithms and 1000 steps of iterations. Figure 11 represents the distortion for the data TOP500, with 100 quantifiers, square grid (10 by 10) for SOM algorithms and 1000 steps of iterations.

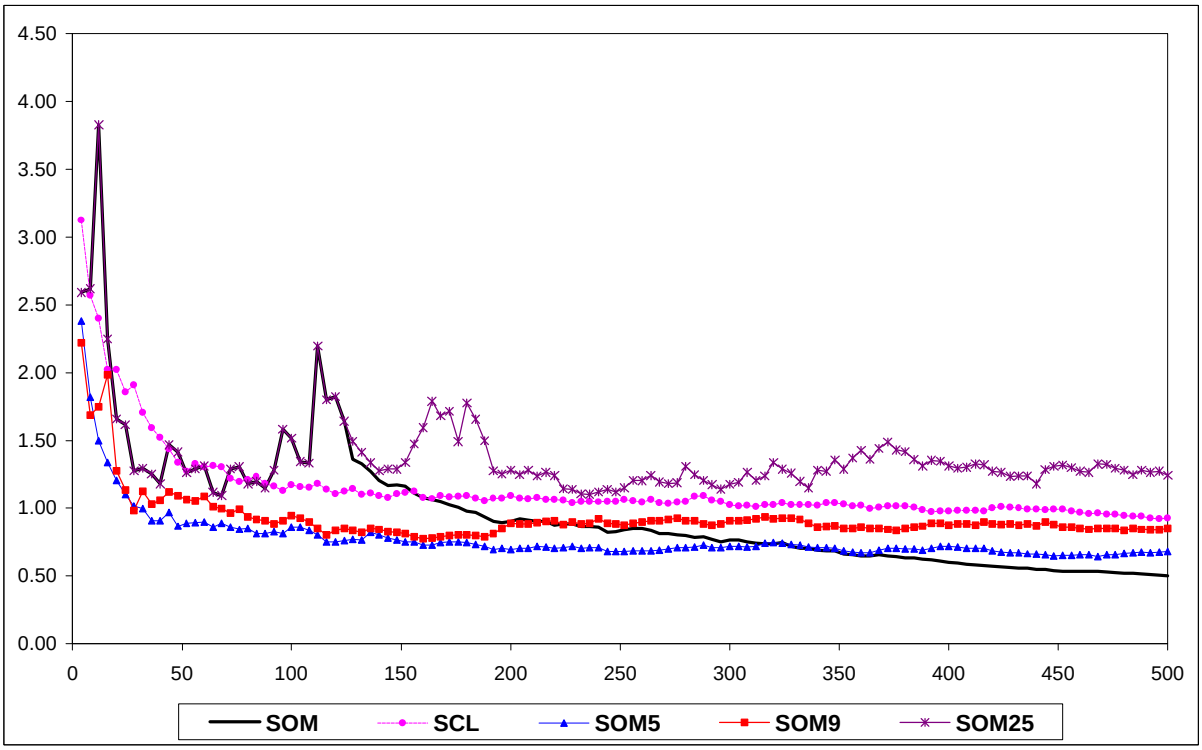

Figure 9 : Evolution of the distortion as a function of the number of iterations, on the Saving data set; 25 quantifiers are used (see text for details on the algorithms).

[1] The data are available on the WEB page …

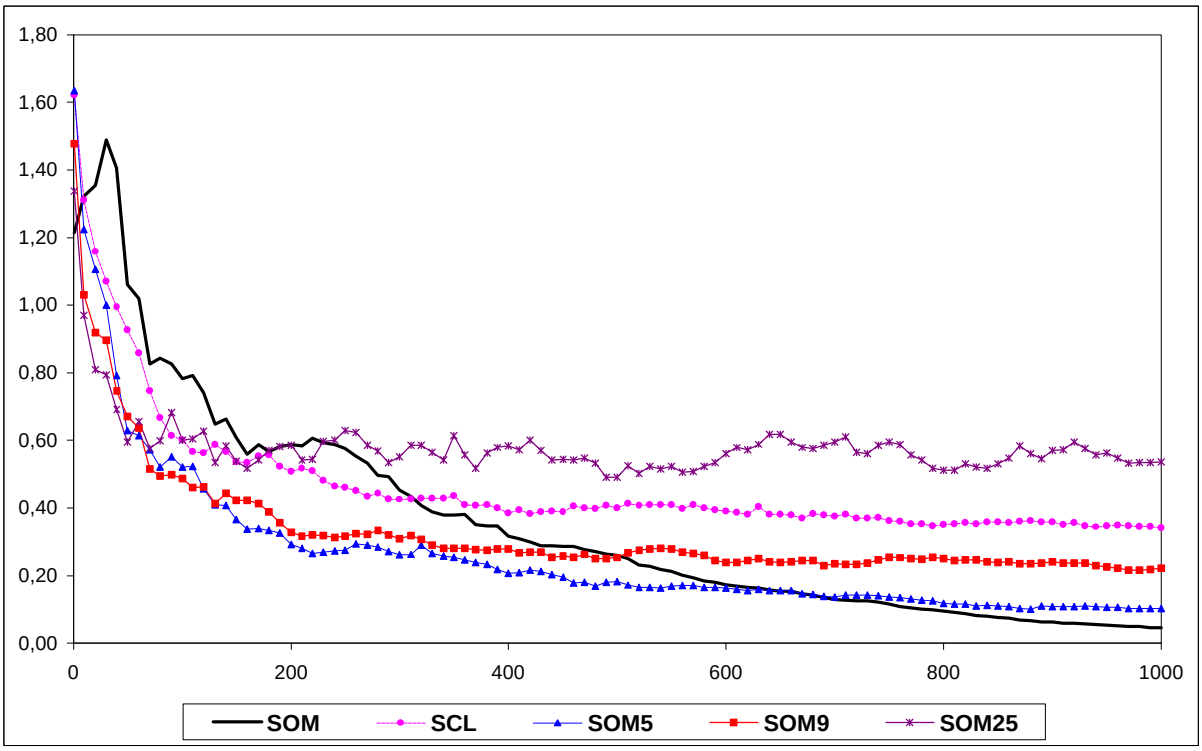

Figure 10 : Evolution of the distortion as a function of the number of iterations, on the Saving data set; 100 quantifiers are used (see text for details on the algorithms).

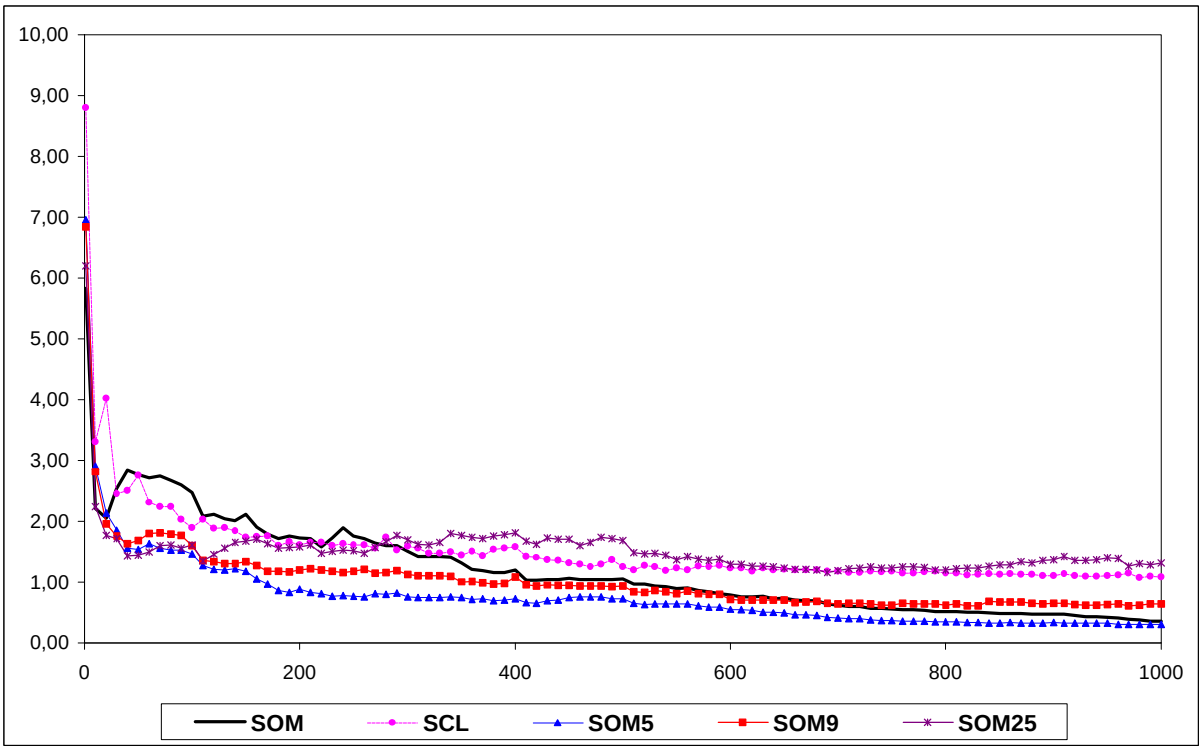

Figure 11 : Evolution of the distortion as a function of the number of iterations, on the Top500 data set; 100 quantifiers are used (see text for details on the algorithms).

In each simulation, we can see that the SOM algorithm performs as the best quantifier (it leads to a lower minimum of the distortion function). The SCL algorithm is very slow, and the SOM with non-decreasing number

of neighbors is powerful at the beginning of the iterations, but allows at some iteration the distortion to increase (or would allow, after a larger number of iterations). In fact the classical SOM algorithm which ends with no neighbor appears to be an excellent VQ algorithm. When the quality of the result is the ultimate goal (regardless of the computation time), one can use the SCL algorithm after the SOM one to refine the solution (or in other words, one can increase the number of iterations performed without neighbors in the SOM procedure).

7. Conclusion

The experiments illustrated in this paper, as well as many others performed on other data sets, prove that the quality of the SOM algorithm resides not only in its topology preservation property, but also its vector quantization one. The SOM algorithm may be recommended compared to other VQ techniques like SCL, in order to reach a better minimum of the distortion error with a fixed number of iterations, or to reach faster a similar value of the distortion.

The better convergence properties cannot be proven theoretically. Nevertheless, we can make the analogy with simulated annealing techniques: the use of a neighborhood in the SOM algorithm introduces apparent disorder, making it possible to escape from a local minimum of the objective function and to increase the slope of convergence. Ending the VQ procedure with the SCL algorithm may be compared to ending a simulated annealing technique with a "temperature" parameter equal to zero.

References

[1] Anderberg M.R., Cluster Analysis for Applications. *Academic Press*, New York, 1973.

[2] de Bodt E., Cottrell M., Verleysen M., *Using the Kohonen Algorithm for Quick Initialization of Simple Competitive Learning Algorithm*, ESANN'99, M.Verleysen Ed., D Facto, Bruxelles, 19-26, 1999.

[3] Forgy E.W., *Cluster analysis of multivariate data: efficiency versus interpretability of classifications.* Biometric Society Meetings, Riverside, California (Abstract in: Biometrics 21, 3, 768), 1965.

[4] Hertz J., Krogh A., Palmer R., Introduction to the Theory of Neural Computation, *Santa Fe Institute*, 1991.

[5] Kohonen T., Self-organizing maps, *Springer*, Berlin, 1995.

[6] Kohonen T., *Comparison of SOM Point Densities Based on Different Criteria*, Neural Computation, vol. 11, 2081-2095, 1999.

[7] Lebart L., Morineau A., Piron M., Statistique exploratoire multidimensionnelle, *Dunod*, Paris, 1995.

[8] Linde Y., Buzo A., Gray R.M., *An algorithm for vector quantizer design*, IEEE Transactions on Communications, vol. COM-28, no. 1, January 1980.

[9] Lloyd S.P., *Least squares quantization in PCM*, IEEE Transactions on Information Theory, vol. IT-28, no. 2, pp. 129-149, March 1982.

[10] MacQueen J. *Some methods for classification and analysis of multivariate observations*, Proc. of the Fifth Berkeley Symposium on Math., Stat. and Prob., vol. 1, pp. 281-296, 1967.

[11] Ritter H. and Shulten K., *On the Stationary State of Kohonen's Self-Organizing Sensory Mapping*, Biol. Cybern., 54, 99-106, 1986.