STOCHASTIC ON-LINE ALGORITHM VERSUS BATCH ALGORITHM FOR QUANTIZATION AND SELF ORGANIZING MAPS

Jean-Claude Fort
Institut Elie Cartan et SAMOS-MATISSE
Université Nancy 1, F-54506 Vandoeuvre-Lès-Nancy, France
E-mail: fortjc@iecn.u-nancy.fr

and
Marie Cottrell, Patrick Letremy
SAMOS-MATISSE UMR CNRS 8595
Université Paris 1, F-75634 Paris Cedex 13, France
E-mail : cottrell,pley@univ-paris1.fr

Abstract. The Kohonen algorithm (SOM) was originally designed as a stochastic algorithm which works in an on-line way and which was designed to model some adaptative features of the human brain. In fact it is nowadays extensively used for data mining, data visualization, and exploratory data analysis. Some users are tempted to use the batch version of the Kohonen algorithm since it is a deterministic algorithm which can be convenient if one needs to get reproducible results and which can go faster in some cases. In this paper, we try to elucidate the mathematical nature of this batch variant and give some elements of comparison of both algorithms. Then we compare both versions on a real data set.

INTRODUCTION

Since about twenty years, the Self-Organizing Maps (SOM) of Teuvo Kohonen have spread through numerous domains where it found effective applications by itself or coupled with other data analysis devices (source separation algorithm or classical filtering for signal processing, multilayer perceptrons for speech recognition, etc.). See for example [15], [16], [14], [8], [18], [4] etc. for definitions and numerous applications.

Nevertheless, SOM appears to be a very powerful extension (see [7], or [6]) of the classical Simple Competitive Learning algorithm (SCL) used to realize the quantization of probability measures, (see for example in [13]

the definition of SCL). So we will treat simultaneously the two algorithms, mentioning when it will be necessary some special features.

The Kohonen algorithm and the SCL algorithm are both on-line algorithms, which means they compute the current values of the code vectors (or weight vectors) at each arrival of a data. They mimic the adaptation of a biological system to a variable environment, susceptible to suffer some changes. These potential modifications are instantaneously taken into account through the variations of the distribution and of the statistics along the observed data series.

Another point of view to design an algorithm in order to quantify or build organized maps is to use all the available information. If we have already observed N data, then we use at one go these N values. Of course we cannot find all at once the good code vectors (or weight vectors). So by iterating this process, we hope to increase the quality of the result at each new iteration. This is the case for the algorithms known as Kohonen Batch algorithm ([17]) or Forgy algorithm ([10]) when there is no neighborhood relations.

ON-LINE ALGORITHMS

The ingredients necessary to the definition of these algorithms are:

- An initial value of the d-dimensional code vectors $X_0(i), i \in I$, where I is the set of units;
- A sequence $(\omega_t)_{t\geq 1}$ of observed data which are also d-dimensional vectors;
- A symmetric neighborhood function $\sigma(i,j)$ which measures the link between units i and j;
- A gain (or adaptation parameter) $(\varepsilon_t)_{t\geq 1}$, constant or decreasing.

Then the algorithm works in 2 steps:

1. Choose a winner (in the winner-take-all version) or compute an activation function (we mention this case, but we will concentrate only on the previous winner-take-all version)
   - winner : $i_0(t+1) = arg\min_i \ dist(\omega_{t+1}, X_t(i))$, where $dist$ is generally the Euclidean distance, but could be any other one;
   - activation function, for instance :

$$\phi(i, X_t) = \frac{\exp(\frac{1}{T})dist(\omega_{t+1}, X_t(i))}{\sum_{j\in I}\exp(\frac{1}{T})dist(\omega_{t+1}, X_t(j))}.$$

     In this expression, if we let the positive parameter T going to infinity, we retrieve the winner-take-all case.

2. Modify the code vectors (or weight vectors) according to a reinforcement rule: the closer to the winner, the stronger is the change.

   - $X_{t+1}(i) = X_t(i) + \varepsilon_{t+1}\sigma(i_0(t+1), i)(\omega_{t+1}(i) - X_t(i))$ in the winner-take-all case or
   - $X_{t+1}(i) = X_t(i) + \varepsilon_{t+1}\sum_j \sigma(j,i)\phi(j, X_t)(\omega_{t+1}(i) - X_t(i))$ in the activation function case.

The case of the SCL algorithm (i.e. only quantization) is obtained by taking $\sigma(i,i) = 1$ and $\sigma(i,j) = 0, i \neq j$. It can be viewed as a 0-neighbor Kohonen algorithm.

Actually few results are known about the mathematical properties of these algorithms (see [2], [3], [11], [1], [20], [21]) except in the one dimensional case. The good framework of study, as for almost all the neural networks learning algorithms, is the theory of stochastic approximation ([12]). In this framework, the first and most informative quantity is the associated Ordinary Differential Equation (ODE). Before writing it down, let us introduce some notations.

From now and for simplicity, we choose the Euclidean distance to compute the winner unit. For any set of code vectors $x = (x(i)), i \in I$, we put

$$C_i(x) = \{\omega / \|x(i) - \omega\| = \min_j \|x(j) - \omega\|\},$$

that is the set of data for which the unit i is the winner. The set of the $(C_i(x)), i \in I$ is called the Voronoï tessellation defined by x.

When it is convenient, we write $x(i,t)$ instead of $x_t(i)$ and $x(.,t)$ for $x_t = (x_t(i), i \in I)$.

Then the ODE (in the winner-take-all case) reads :

$$\forall i \in I, \frac{dx(i,u)}{du} = -\sum_{j \in I} \sigma(i,j) \int_{C_j(x(.,u))} (x(i,u) - \omega)\mu(d\omega),$$

where μ stands for the probability distribution of the data set and where the second member is the expectation of $\sigma(i_0(t), i)(\omega_{t+1}(i) - X_t(i))$ with respect to the distribution μ.

In practice, distribution μ looks like a discrete one, because we only use a finite number of "examples" or at least a denumerable set. But generally the good way to model the actual set of possible data is to consider a continuous probability distribution (for a frequency, some Fourier coefficients or coordinate in an Hilbertian base, a level of gray, ...).

When μ is discrete (in fact has a finite support), one can see that the ODE locally derives from an energy function (it means that the second member can be seen as the opposite gradient of this energy function, which is then decreasing along the trajectories of the ODE), that we call *extended distortion*, in reference to the classical distortion in the SCL case. See [19] for elements of proof.

In the SCL case, the (classical) distortion is

$$D(x) = \sum_{i \in I} \int_{C_i(x)} \|x(i) - \omega\|^2 \mu(d\omega) \quad (1)$$

$$= \int \min_i \|x(i) - \omega\|^2 \mu(d\omega). \quad (2)$$

In the general case of SOM, the extended distortion is

$$D(x) = \sum_{i \in I} \sum_j \sigma(i,j) \int_{C_j(x)} \|x(i) - \omega\|^2 \mu(d\omega).$$

When μ is diffuse (or has a diffuse part), then D is no more an energy function for the ODE, because there are spurious (or parasite) terms due to the neighborhood function (see [9] for proof). Then when μ is diffuse, the only case where D is still an energy function for the ODE is the SCL case.

Nevertheless, it has to be noticed that

- when D is an actual energy function, then it is not everywhere differentiable and we only get local information which is not what is expected from such a function. In particular, the existence of this energy function is not sufficient to rigorously prove the convergence !
- when D is everywhere differentiable, then it is no more an actual energy function even if it gives a very good information about the convergence of the algorithm.

In any case, if they would be convergent, both algorithms (SOM and SCL) would converge toward an equilibrium of the ODE (even if this fact is not mathematically proved). These equilibria are defined by

$$\forall i \in I, \sum_j \sigma(i,j) \int_{C_j(x^*)} (x^*(i) - \omega) \mu(d\omega) = 0.$$

This also reads

$$x^*(i) = \frac{\sum_j \sigma(i,j) \int_{C_j(x^*)} \omega \ \mu(d\omega)}{\sum_j \sigma(i,j) \mu(C_j(x^*))} \quad (3)$$

In the SCL case, it means that all the $x^*(i)$ are the centers of gravity of their Voronoï tiles. In the statistical literature, it is called self-consistent point. More generally, in the SOM case, $x^*(i)$ is the center of gravity (for the weights which are given by the neighborhood function) of all the centers of the tiles.

From this remark, the definitions of the batch algorithms can be derived.

THE BATCH ALGORITHMS

Using (3), we immediately derive the definitions of the batch algorithms. Our aim is to find a solution of (3) through a deterministic iterative procedure. In a first approach, let us define a sequence of code vectors by:

- x^0 is an initial value
- and

$$x^{k+1}(i) = \frac{\sum_j \sigma(i,j) \int_{C_j(x^k)} \omega \ \mu(d\omega)}{\sum_j \sigma(i,j) \mu(C_j(x^k))}.$$

Now if the sequence converges (or at least for any sub-sequence that converges, which always exists if $(x^k)_{k \geq 1}$ is bounded), it converges toward a solution of (3).

This defines exactly the Kohonen Batch algorithm when μ weights only a finite number N of data. It reads :

$$x_N^{k+1}(i) = \frac{\sum_j \sigma(i,j) \sum_{l=1}^N \omega_l \ \mathbf{1}_{C_j(x^k)}(\omega_l)}{\sum_j \sigma(i,j) \sum_{l=1}^N \mathbf{1}_{C_j(x^k)}(\omega_l)}.$$

Recall that $\mathbf{1}_{C_j(x^k)}(\omega_l)$ is 1 if ω_l belongs to $C_j(x^k)$ and 0 elsewhere.

Moreover, if we assume that when N goes to infinity, $\mu_N = \frac{1}{N} \sum_{l=1}^N \delta_{\omega_l}$ (δ_ω stands for the Dirac measure at ω) weakly converges toward μ, we then have (under mild conditions)

$$\lim_{N \to \infty} \lim_{k \to \infty} x_N^{k+1}(i) = x^*(i),$$

where x^* is a solution of (3).

The extended distortion is piecewise convex since the changes of convexity take place on the median hyperplanes which are the frontiers of the Voronoï tessellation. And at this stage, we may notice that the Kohonen Batch algorithm is nothing else but a "quasi-Newtonian" algorithm which minimizes the extended distortion D associated with μ_N.

In fact, when there is no data on the borders of the Voronoï tessellation, it is possible to verify that

$$x_N^{k+1} = x_N^k - \left(diag \nabla^2 D(x_N^k) \right)^{-1} \nabla D(x_N^k)$$

where $diag M$ is the diagonal matrix made with the diagonal of M, ∇D is the gradient of D and $\nabla^2 D$ is the Hessian of D, which exists. It is a "quasi-Newtonian" algorithm because it uses only the diagonal part of the Hessian matrix and not the full matrix.

This proves that in every convex set where D is differentiable, (x_N^k) converges toward a x^* minimizing D. Unfortunately, there are many such disjoint sets and in each of them there is a local minimum of D, even if we do not take into account the possible symmetries or geometric transformations letting μ invariant.

A COMPARISON ON SIMULATED DATA

Actually, it very often happens that the batch algorithm finds not fully satisfactory solution of (3). It strongly depends on the choice of the initial value x^0. On the contrary, the on-line algorithm, especially with constant ε, can escape from reasonably deep traps and find a better solution of equation (3) even starting from the attraction basin of a poor quality solution.

In Fig.1 and Fig.2, are depicted two solutions respectively found by a Kohonen Batch and an on-line SOM algorithms in the following conditions :

The units are organized on a (7×7) grid endowed with a 8 nearest neighbors topology. The data set is made of a 500-sample of independent, uniformly distributed on $[0, 1]$, random variables. The initial code vectors are random too.

The Batch algorithm is processed for 100 times, which leads to the limit (numerically at least).

The stochastic on-line algorithm is iterated 50 000 times by drawing at random one data at each step.

The gain ε_t is decreasing and given by $\varepsilon_t = 0.125/(1 + 6.10^{-4}t)$.

The two solutions that we found are very stable for each algorithm, but due to the random property, the on-line algorithm discovers the best one.

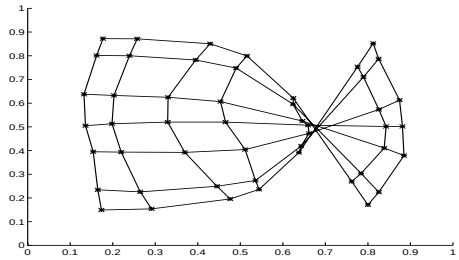

Figure 1: Kohonen Batch for a 7 by 7 grid

We can observe that the Kohonen Batch get trapped in a local minimum and do not correctly realize neither the self-organization nor the quantization. We did a lot of similar simulations and always got the same kind of results.

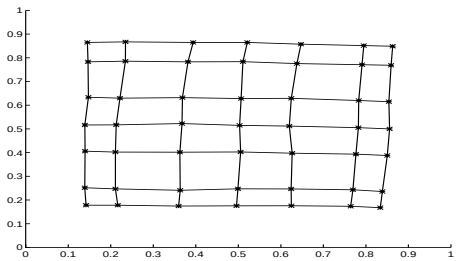

Figure 2: On-line SOM for a 7 by 7 grid

APPLICATION TO DATA ANALYSIS

If the data consist in a set of N observations, each of one being described by a d-dimensional vector, the Kohonen algorithms (both on-line and batch versions) provide a very powerful tool to classify, represent and visualize the data in a Kohonen map (most of time a two- or one-dimensional map, one speaks of *grid* or *string*). See [4], [14], [18] for numerous examples.

After learning, that is after convergence of the algorithm (whatever it is), each unit i is represented in the $\mathcal{R}^d$ space by its code vector $X(i)$. Each code vector resembles more to its neighbors than to the other code vectors, due to the topology conservation property. This feature provides an organized map of the code vectors, which gives prominence to the progressive variations across the map for each variable which describes the data.

Classification and Representation

After convergence, each observation is classified by the nearest neighbor method (in $\mathcal{R}^d$) : observation l belongs to class C_i if and only if the code vector $X(i)$ is closer to observation i than to any other. Similar observations are classified into the same class or into neighbor classes, that is not true when using any other classification method. Then the observations can be listed or drawn inside their classes and this fact provides a two- or one-dimensional representation on the grid (or along the string). In this kind of map, one can appreciate how the observations differ from one class to the next ones, and can study the homogeneity of the classes.

Two-level classification

The classification that we get in the first step is generally based on a large number of classes, and this fact makes the interpretation and the establishment of a typology difficult. So it is very convenient to reduce the number of classes by using a hierarchical clustering of the code vectors. So the most similar classes are grouped into a reduced number of larger ones. These macro-classes create connected areas in the initial maps, and the neighborhood relations are kept. This grouping together facilitates the interpretation of the contents and the description of a typology of the observations.

Quality of the organization
We can control the quality of the organization by considering the results of the two-level classification. The fact that the hierarchical clustering of the code vectors only groups connected areas means that the map is well organized, since it means that to be close in the data space coincides with to be close on the map.

Quality of the quantization
It is given by the distortion (2).

Quality of the classification
As many classifications methods can be used, we need to define some quality criteria to compare classifications with the same number of classes. We compute as usual the one-dimensional Fisher statistics for each variable which describes the observations and the multi-dimensional Wilks and Hotelling statistics for global criteria. Let us recall the definitions :

Let us denote by J the number of classes (it will be less than I if we proceed to a reduction of the classes number using a hierarchical clustering). If the observations are re-numbered according to the class they belong, we denote by $\omega_{jl} = (\omega_{jl}^1, \omega_{jl}^2, \ldots, \omega_{jl}^d), 1 \leq j \leq I, 1 \leq l \leq N_j$, the l-th observation in class j.

For each component $p, 1 \leq p \leq d$, the intra-classes sum of squares is

$$SCW(p) = \sum_{j=1}^{J} \sum_{l=1}^{N_j} (\omega_{jl} - \omega_{j.})^2$$

while the inter-classes sum of squares is

$$SCB(p) = \sum_{j=1}^{J} N_j (\omega_{j.} - \omega_{..})^2,$$

where the points signify the means computed on the absent indexes.

The Fisher statistic associated to variable p is defined by

$$F(p) = \frac{SCB(p)/J - 1}{SCW(p)/N - J}.$$

This statistics has a Fisher$(J-1, N-J)$ distribution when the classes have no sense, that is when the J classes can be confounded. So the larger the value of $F(p)$ is, the better the variable p is discriminating.

These statistics are very useful, but give indications only for each variable separately. In order to globally evaluate the quality of the classification, one has to define in the same way the intra-classes variance matrix W, and the inter-classes variance matrix B, and to compute for example the Wilks statistics

$$Wilks = \frac{\det(W)}{\det(W+B)}$$

or the Hotelling statistics

$$Hot = trace(W^{-1}B).$$

These statistics are tabulated, but very often one uses an approximation by a chi-squared distribution. The smaller (resp. larger) the Wilks (resp. Hotelling) statistic is, the better is the classification. Note that it is not possible to find a criterion which would be better than any other, since there does not exist an uniformly most powerful test of the hypothesis "the classes can be confounded" against "the classes are different", except for a two-classes problem.

Let us remark that a "good" classification is that one which has a strong homogeneity inside the classes and a strong separation between the classes. So we will use these indicators of quality to compare several classifications.

The data

We use in this part a real database. It contains seven ratios measured in 1996 on the macroeconomic situation of 96 countries: annual population growth, mortality rate, analphabetism rate, population proportion in high school, GDP per head, GDP growth rate and inflation rate. This kind of dataset (but not always for the same year) has been already used in [14], [18] or [8] and is available through http://panoramix.univ-paris1.fr/SAMOS.

So in order to compare both Kohonen algorithms (on-line and Batch), we classify the 96 countries using a 6×6 grid, and a simplified neighborhood function where $\sigma(i,j) = 1$ if i and j are neighbors and $= 0$ if not. The size of the neighborhood decreases with time. The initial values of the code vectors are randomly chosen.

First we use the classical SOM algorithm, with 500 iterations, a decreasing function ε. For lack of space, we do not represent the resulting map. It provides a nice representation of all the countries displayed over the 36 classes. Rich countries are at the right and at the top, the very poor countries are at the left, there is continuity from the rich countries to the poor ones. There are two countries in the corner at the top and at the left and they are those which have a very large inflation rate, etc. Figure (3) shows the macro-classes and the code vectors which can be considered as typical countries which summarize the countries of their classes. We choose to keep 7 macro-classes, and it would be easy to describe them and derive a simple typology

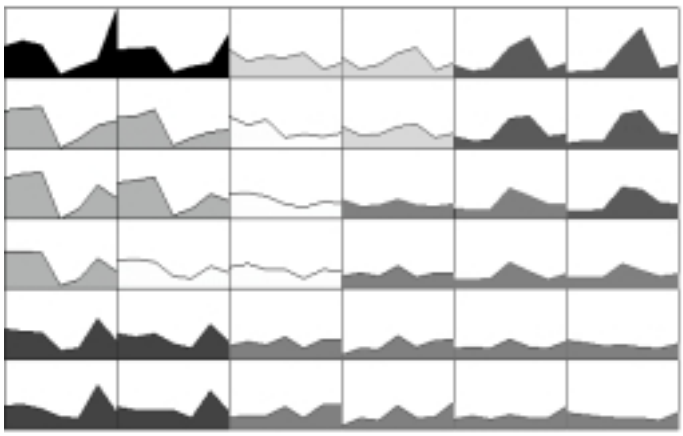

Figure 3: SOM on-line algorithm, 36 code-vectors, 7 macro-classes, the 7-dimensional code vectors are drawn as curves which join the 7 points

of the countries. We observe that the code vectors are very well organized, vary with continuity from one class to the next ones and that the clustering groups together only connected classes. That is a visual proof of the quality of the organization.

Secondly, we use the Batch Kohonen algorithm, with 5 iterations (which is equivalent to 500 for the on-line algorithm), and a decreasing neighborhood function. First we start from the same initialization as in the first case. The resulting map is not organized. All the 96 countries are grouped into 4 classes (among the 36). One class contains the rich countries (Germany, France, USA, etc...). In the second we find some intermediate countries (Greece, Argentina, Czechoslovakia, etc...). The poor countries are divided into two classes, the poor (Cameroon, Bolivia, Lebanon, etc...) and the very poor ones (Yemen, Haiti, Angola, etc...). The classification is meaningful, but there is no organization on the map, close code vectors are not similar, the macro classes gather not connected classes, etc. See figure (4). In this case, it is not necessary to compute the macro-classes, since there are only 4 non- empty classes.

At last, we use the Batch Kohonen algorithm, with the same parameters, but starting from another initial values for the code vectors. In that case, we get a reasonably organized map, the countries are displayed over all the map, but the two countries characterized by the high inflation rate are in opposite classes (at the right top and at the left bottom). When we use a hierarchical clustering with 7 macro-classes, these opposite classes belong to the same macro-class and that means that the organization is not complete. See figure (5).

In the next table, we compare the classification criteria and the distortion for 4 classifications: On-line SOM with 4 macro classes (SOM4), Batch algorithm I with 4 non-empty classes (Batch4), On-line SOM with 7 macro-

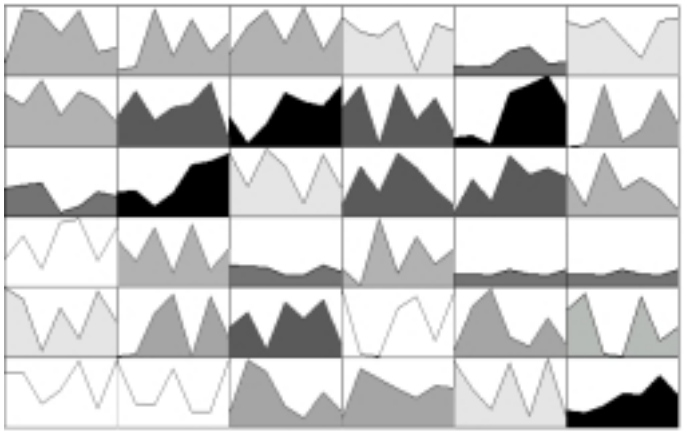

Figure 4: Batch algorithm I, 36 code-vectors, 7 macro-classes, only 4 non-empty classes

classes (SOM7), Batch algorithm II with 7 macro-classes (Batch7).

	SOM4	Batch4	SOM 7	Batch7
$F(1)$	51.80	52.84	29.16	47.32
$F(2)$	126.92	146.24	63.14	80.70
$F(3)$	135.98	164.91	76.66	67.52
$F(4)$	71.79	60.96	37.40	57.22
$F(5)$	102.58	160.56	76.02	78.24
$F(6)$	18.80	21.28	22.65	18.13
$F(7)$	4.37	2.24	81.74	91.55
Wilks	0.023	0.017	0.001	0.0007
Hotelling	12.23	13.79	21.49	24.98
Distortion	0.99	2.71	0.99	0.4

So the Batch algorithm II gives the best results from both points of view of classification and quantization. But we saw that it is not perfectly organized. We note also that the Batch algorithm is extremely sensitive to the initial values while the SOM algorithm is very robust. We did a lot of different runs and always got similar results. See for example [5].

CONCLUSION

We have tried to clearly define and compare the two most spread methods to compute SOM maps: one (on-line SOM) belongs to the family of stochastic algorithms while the second one is linked to quasi-Newtonian methods. De-

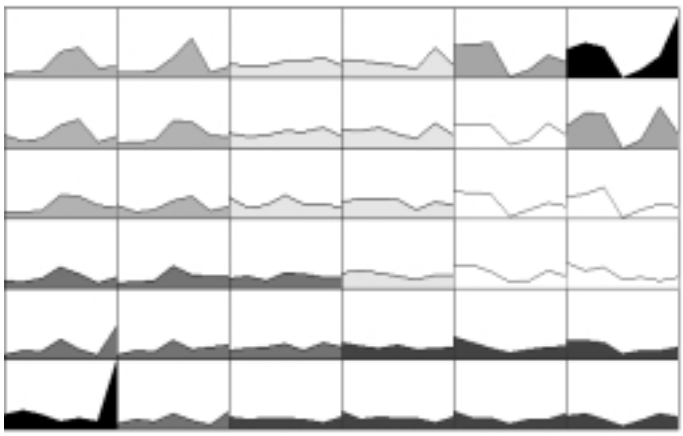

Figure 5: Batch algorithm II, 36 code-vectors, 7 macro-classes

spite its simplicity, we put some insight into the fact that randomness could lead to better performances. This is not truly surprising, but we aimed at giving precise definitions and clear numerical results. In fact, these remarks lead to conclude that the initialization choice is much more crucial for the Kohonen Batch algorithm. The Kohonen Batch algorithm could seem to be easier to use because there is no gain function ε_t to adjust and because it can converge quicker than the on-line one. But this can be misleading because for the batch algorithm, it is much more necessary to carefully choose the initial values and eventually repeat a lot of computations in order to find better solutions.

REFERENCES

[1] M. Benaïm, J. Fort and G. Pagès, "Convergence of the one-dimensional Kohonen algorithm," **Advances in Applied Probability**, vol. 30, 1998.

[2] M. Cottrell, J. Fort and G. Pagès, "Two or three things that we know about the Kohonen algorithm," in **Proc. of ESANN'94**, Bruges: D Facto, 1994, pp. 235–244.

[3] M. Cottrell, J. Fort and G. Pagès, "Theoretical aspects of the SOM algorithm," **Neurocomputing**, vol. 21, pp. 119–138, 1998.

[4] M. Cottrell and P. Rousset, "The Kohonen Algorithm: A Powerful Tool for Analysing and Representing Multidimensional Quantitative and Qualitative Data," in **Proc. IWANN'97, Biological and Artificial Computation: From Neuroscience to Technology**, Springer, 1997, pp. 861–871.

[5] E. de Bodt and M. Cottrell, "Bootstrapping Self-Organizing Maps to Assess the Statistical Significance of Local Proximity," in **Proc. of ESANN'2000**, Bruges: D Facto, 2000, pp. 245–254.

[6] E. de Bodt, M. Cottrell and M. Verleysen, "Using the Kohonen Algorithm for Quick Initialization of Simple Competitive Learning Algorithm," in **Proc. of ESANN'99**, Bruges: D Facto, 1999, pp. 19–26.

[7] E. de Bodt, M. Verleysen and M. Cottrell, "Kohonen Maps versus Vector Quantization for Data Analysis," in **Proc. of ESANN'97**, Bruges: D Facto, 1997, pp. 211–218.

[8] G. Deboeck and T. Kohonen, **Visual explorations in Finance with Self-Organizing Maps**, Springer, 1998.

[9] E. Erwin, K. Obermayer and K. Shulten, "Self-organizing maps : ordering, convergence properties and energy functions," **Biological Cybernetics**, vol. 67, pp. 47–55, 1992.

[10] E. Forgy, "Cluster analysis of multivariate data: efficiency versus interpretability of classifications," **Biometrics**, vol. 21, no. 3, pp. 768, 1965.

[11] J. Fort and G. Pagès, "On the a.s. convergence of the Kohonen algorithm with a general neighborhood function," **Annals of Applied Probability**, vol. 5, no. 4, pp. 1177–1216, 1995.

[12] J. Fort and G. Pagès, "Convergence of Stochastic Algorithms: from the Kushner & Clark theorem to the Lyapunov functional," **Advances in Applied Probability**, vol. 28, no. 4, pp. 1072–1094, 1996.

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

[14] S. Kaski, "Data Exploration Using Self-Organizing Maps," **Acta Polytechnica Scandinavia**, vol. 82, 1997.

[15] T. Kohonen, **Self-Organization and Associative Memory**, Berlin: Springer, 1984-1989.

[16] T. Kohonen, **Self-Organizing Maps**, Berlin: Springer, 1995.

[17] T. Kohonen, "Comparison of SOM Point Densities Based on Different Criteria," **Neural Computation**, vol. 11, pp. 2081–2095, 1999.

[18] E. Oja and S. Kaski, **Kohonen Maps**, Elsevier, 1999.

[19] H. Ritter, T. Martinetz and K. Shulten, **Neural Computation and Self-Organizing Maps, an Introduction**, Reading: Addison-Wesley, 1992.

[20] A. Sadeghi, "Asymptotic Behavior of Self-Organizing Maps with Non-Uniform Stimuli Distribution," **Annals of Applied Probability**, vol. 8, no. 1, pp. 281–289, 1997.

[21] A. Sadeghi, "Self-organization property of Kohonen'smap with general type of stimuli distribution," **Neural Networks**, vol. 11, pp. 1637–1643, 1998.