
PERGAMON

Neural Networks 15 (2002) 993–1003

Neural Networks

www.elsevier.com/locate/neunet

2002 Special Issue

Self-organizing maps with recursive neighborhood adaptation

John A. Lee*, Michel Verleysen[1]

Department of Electricity, Université catholique de Louvain, Place du Levant, 3, B-1348 Louvain-la-Neuve, Belgium

Abstract

Self-organizing maps (SOMs) are widely used in several fields of application, from neurobiology to multivariate data analysis. In that context, this paper presents variants of the classic SOM algorithm. With respect to the traditional SOM, the modifications regard the core of the algorithm, (the learning rule), but do not alter the two main tasks it performs, i.e. vector quantization combined with topology preservation. After an intuitive justification based on geometrical considerations, three new rules are defined in addition to the original one. They develop interesting properties such as recursive neighborhood adaptation and non-radial neighborhood adaptation. In order to assess the relative performances and speeds of convergence, the four rules are used to train several maps and the results are compared according to several error measures (quantization error and topology preservation criterions). © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Self-organizing maps; Vector quantization; Recursive neighborhood adaptation; Non-radial neighborhood adaptation; Topology preservation; Topographic mapping

1. Introduction

The seminal idea leading to the principle of the self-organizing maps (SOMs) can already be found in the work of von der Malsburg (von der Malsburg, 1973). The goal of his paper was to model the stochastic patterns of eye dominance and orientation preference in the visual cortex. After this biological introduction to the topic, the analysis of the mathematical properties of the topographic maps only started in the eighties when Teuvo Kohonen (Kohonen, 1982, 1989) simplified the biological model of von der Malsburg. Kohonen also clearly stated the famous learning rule which allows the topographic maps to organize themselves. This learning process has widely popularized the topographic maps under the names of SOMs or 'self-organizing feature maps' (SOFMs).

Actually, the explanation of this success can be found in the fact that SOMs are easy to understand: the task they perform is very intuitive (but, paradoxically, it is very difficult to express it with mathematical formulas). More precisely, SOMs perform simultaneously the combination of two subtasks: vector quantization and topographic representation. This 'magic mix' has been used not only as a vector quantization method, but also in other fields where self-organization plays a key role. For example, SOMs can be used for non-linear Blind Source Separation (Pajunen, Hyvärinen, & Karhunen, 1996), or non-linear projection of data (Kraaijveld, Mao, & Jain, 1995; Mao & Jain, 1995). Considering SOMs as a special case of vector quantization method, the difference with more classical methods holds in some kind of predefined information given to the neurons before the learning phase. Actually, this predefined information may be seen as a mutual organization of the neurons, which affects the learning process. The result is the self-organization property which tries to reproduce the topographic organization in the quantized space.

In view of that combination of vector quantization and topology preservation, this paper studies the modifications that can be made to the classic SOM algorithm. The proposed changes only affect the 'heart' of the algorithm, i.e. the learning rule. The goal consists in defining new learning rules which develop the same mix of interesting properties as the traditional SOM.

After this introduction, Section 2 will review the SOM algorithm in details, as it was stated in (Kohonen, 1989). Section 3 presents some changes that can be made to the traditional SOM learning rule, without altering its fundamental properties (vector quantization and topology preservation). After a short and intuitive presentation, Section 3.1 defines a new learning rule, followed by the presentation of one of its properties in Section 3.2. Section 4 extends the new rule and generalizes it to a set of four rules. Opening the experimental part of the work, Section 5 presents the material used to evaluate practically the four rules, i.e. the map parameters, the learning sets and the error criterions.

* Corresponding author. Tel.: +32-2-47-8133; fax: +32-2-47-2598.
E-mail address: lee@dice.ucl.ac.be (J.A. Lee).

[1] Michel Verleysen is senior Research Associate of the Belgian National Fund for Scientific Research (FNRS).

PII: S0893-6080(02)00073-4

Section 6 discusses the results of the experiments from two points of view, i.e. the quantization quality and the topology preservation. Finally, Section 7 draws some conclusions, showing the advantages of the proposed rules with respect to the traditional one.

2. Review of the SOM algorithm

Basically, there are two main classes of vector quantization algorithms: the 'winner takes all' (WTA) methods and the 'winner takes most' ones. In the WTA class, only one neuron is adapted when a learning pattern stimulates the network. This sole neuron is often called the 'best matching unit' (BMU) and defined as the closest one from the pattern, with regards to a well specified distance measure. In more biological terms, the WTA class allows only one neuron to fire and adapt, while several other neurons in addition to the BMU may fire in WTM algorithms like the self-organized map. Indeed, the strength of adaptation does not only depend on the distance to the pattern, but also on some static information stored in the neurons. Actually, before learning, the neurons are given a fixed position in a Euclidian space. As these positions are often two-dimensional (2D) and regularly spaced, the SOM looks like a grid in what is called the 'grid space'. These locations also define topographic or topological relations between the neurons, in a space well distinguished from the one where the patterns lies. Such relations indicates which neurons have to be adapted simultaneously with the BMU. This means that unlike other WTM algorithm, the SOM adapts neurons with a strength that depends not only on the distance from the BMU in the pattern space, but also on to the same distance computed in the grid space. Translating all these ideas in formulas, the map can be defined by:

(1) a matrix $\vec{W}$, of which row $\vec{w}_r$ gives the weights or coordinates of neuron r in the pattern space;
(2) a function $d(q, r)$, measuring the grid distance between neurons q and r in the grid space.

Notice that the distance $d(q,r)$ may be implicitly determined either by a true position in a grid or by another more general mathematical structure like a weighted graph with neurons as nodes. Anyway, this definition leaves enough freedom regarding the shape of the map or the structure of the neighborhood. For example, the neurons may be placed on a one-dimensional (1D) string (straight line) or on a 2D grid, the neighborhood may be rectangular or hexagonal, etc. Having $\vec{W}$ and $d(q, r)$, the map may begin its learning process. Assuming that:

- vectorial pattern $\vec{x}_i$ stimulates the map at learning time t,
- index $* = \arg\min_r \|\vec{w}_r^t - \vec{x}_i\|$ points to the best-matching unit,

then all neurons are adapted according to following rule:

$$\vec{w}_r^{t+1} = \vec{w}_r^t + \Delta\vec{w}_r^t = \vec{w}_r^t + \gamma_r^t(\vec{x}_i - \vec{w}_r^t). \quad (1)$$

In the last equation, γ_r^t is the learning rate for neuron r, which factorizes in the following product:

$$\gamma_r^t = \alpha^t \nu_r^t \quad (2)$$

where α^t and ν_r^t are values between 0 and 1, decreasing with time t. The first factor α^t is the value of the global learning rate, common for all neurons, while ν_r^t is known as the neighborhood factor or neighborhood kernel. In order to observe the self-organization property, the neighborhood factor must be a decreasing function of the grid distance $d(*, r)$ between the BMU and the neuron being adapted. For example, the neighborhood function may be the 'Bubble' function (Kohonen, 1995):

$$\nu_r^t = 1, \quad \text{when } d(*, r) < \lambda^t \quad (3)$$

$$\nu_r^t = 0, \quad \text{when } d(*, r) > \lambda^t \quad (4)$$

or a Gaussian kernel (Kohonen, 1995; Ritter, Martinetz, & Schulten, 1992):

$$\nu_r^t = \exp(-0.5(d(*, r)/\lambda^t)^2). \quad (5)$$

In both equations, λ^t acts like a neighborhood radius, which has to decrease as time goes by, in order to ensure the convergence of the map. As mentioned above, some freedom has already been explored regarding either the neighborhood structure or the neighborhood function. Are there such possibilities for the learning rule itself? Actually, all modifications are acceptable provided two conditions are fulfilled:

- The BMU is adapted radially towards the learning pattern, in order to guarantee a good quantization;
- The changes made to the learning rule keep using the distance function $d(q, r)$ (i.e. the predefined topographic information stored in the map), in order to observe self-organization in the quantized space.

Modified learning rules are defined in Section 3.

3. Recursive neighborhood adaptation

As mentioned in Section 1, the SOM performs the combination of two tasks: vector quantization and topographic mapping. In the original algorithm, these two subtasks are deeply interlaced in the learning rule. This is due to the fact that the learning rule adapts the neighbors of the BMU in the same way as the BMU, i.e. in a vector quantization way, radially towards to the stimulating vector. But fundamentally, there is no reason to adapt neighbors with VQ in mind, because neighbors do not play any interesting role in the coding and decoding processes of vector quantization, where only the

BMU is useful. Actually, the radial adaptation of the neighbors achieves indirectly the task of topographic mapping. Obviously, other ways exist to perform more directly the topology preservation. For example, the adaptation of the neighbors does not require to take into account the stimulating vector; only the BMU and the topographic relations stored in the neurons are important. The same conclusions may be drawn more intuitively, as explained below.

3.1. The fisherman's rule

From a geometrical point of view, the traditional SOM rule adapts neurons radially around the stimulating vector $\vec{x}_i$. Other possibilities clearly exist. Indeed, the SOM may be seen intuitively as a fishing net floating in the sea. If a fish compares to a learning pattern, then the net will react more or less like a SOM and the node touched by the fish corresponds to the BMU. But at this point, neighbors of the BMU will not move radially towards the fish mouth. Instead, neighboring nodes will be pulled by each other: the BMU pulls the direct neighbors, these neighbors pull farther neurons and so on. Actually, neurons are adapted in a recursive manner. All these ideas may be written more formally into the following learning rule

$$\Delta\vec{w}_*^t = \alpha^t(\vec{x}_i - \vec{w}_*^t) \tag{6}$$

$$\Delta\vec{w}_r^t = \gamma_r^t\left(\vec{w}_q^{t+1} - \vec{w}_r^t\right), \qquad \forall r \neq * \tag{7}$$

where the last equation is recursive. Therefore, index q is determined recursively and neurons may not be adapted in random order: after the update of the BMU, only the direct neighbors may be adapted. When this is done, neighbors of second order are moved and so on. More generally, if the map is defined by means of a weighted graph, then the grid or map distance $d(*, r)$ to the BMU can be computed in advance by a shortest path algorithm (Dijkstra, 1959) and the condition for adaptation becomes: if all neighbors on the shortest path between neuron r and the BMU have already been adapted, then neuron r may also be moved.

Unlike in the original learning rule (Eq. (1)), the two subtasks of the SOM, i.e. vector quantization and topographic mapping, have now their own learning rule. Eq. (6) adapts the BMU in the same way as Competitive Learning, while Eq. (7) moves neighbors without explicit use of the learning pattern $\vec{x}_i$.

Although this recursive learning rule seems uselessly complex to implement, it shows an interesting behavior.

3.2. Property of the fisherman's rule

By comparison to the original rule, the fisherman's rule presents an interesting property due to its recursive formulation. To demonstrate it simply, suppose that the map is reduced to a string in a 1D pattern space and that its current state is such that:

- $\vec{w}_r^t = r$, with $\vec{w}_*^t = * = 0$;
- $\vec{w}_r^t$ is linked with $\vec{w}_{r-1}^t$;
- the stimulating pattern is $\vec{x}_i = 0$.

If λ^t is set to $+\infty$, then the neighborhood has no limit, λ_r^t degenerates to α^t and one can rewrite the traditional rule into

$$\vec{w}_r^{t+1} = \vec{w}_r^t + \alpha^t(\vec{x}_i - \vec{w}_r^t) = (1 - \alpha^t)r, \tag{8}$$

while the fisherman's rule leads to

$$\vec{w}_*^{t+1} = \vec{w}_*^t + \alpha^t(\vec{x}_i - \vec{w}_*^t) = (1 - \alpha^t)r = 0, \tag{9}$$

$$\vec{w}_r^{t+1} = \vec{w}_r^t + \alpha^t(\vec{w}_{r-1}^{t+1} - \vec{w}_r^t) = (1 - \alpha^t)r + \alpha^t\vec{w}_{r-1}^{r+1}, \quad \forall r \neq *, \tag{10}$$

The recurrence in Eq. (10) is solved by

$$\vec{w}_r^{t+1} = (1 - \alpha^t)\sum_{i=0}^{r} i(\alpha^t)^{r-i} = r - \alpha^t\frac{1 - (\alpha^t)^r}{1 - \alpha^t}. \tag{11}$$

The results of the calculations above are illustrated in Fig. 1 which is a plot of the weight ratio $\vec{w}_r^{t+1}/\vec{w}_r^t$. Obviously, the fisherman's rule induces an attenuation of the adaptation, as the grid distance $d(*, r)$ to the winning neurons grows. This behavior recalls the effects of the Gaussian neighborhood, i.e. when a Gaussian kernel replaces the simpler Bubble function. This property of the fisherman's rule is quite desirable, as it is known that SOMs with Gaussian neighborhood often give better results (Kohonen, 1995; Ritter et al., 1992) than with the Bubble function.

4. Hybrid rules

Trying to compare the traditional learning rule and the fisherman's one is almost impossible because their nature is totally different. In order to make them comparable, one has to list their differences one by one and try each combination. Actually, there are two differences between these two learning rules: the traditional rule is non-recursive and purely radial, while the fisherman's one is recursive but not radial. This leads to four combinations shown in Eqs. (12)–(15), Table 1 and Figs. 2 and 3. The four rules can be written more or less in the same way:

$$\Delta\vec{w}_r^t = \gamma_r^t\|\vec{x}_i - \vec{w}_r^t\|\frac{\vec{x}_i - \vec{w}_r^t}{\|\vec{x}_i - \vec{w}_r^t\|}, \tag{12}$$

$$\Delta\vec{w}_r^t = \gamma_r^t\|\vec{w}_q^{t+1} - \vec{w}_r^t\|\frac{\vec{x}_i - \vec{w}_r^t}{\|\vec{x}_i - \vec{w}_r^t\|}, \tag{13}$$

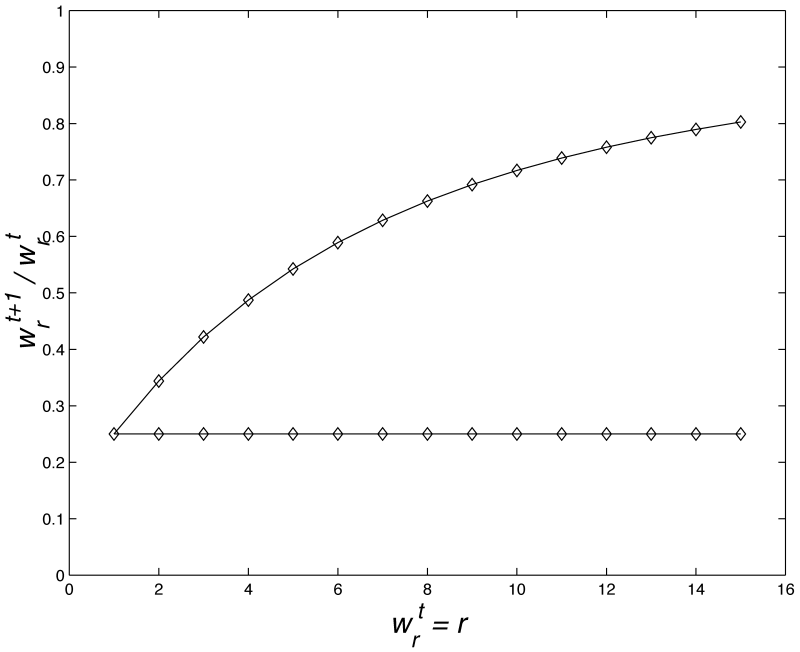

Fig. 1. Weight ratio after and before adaptation, with $\alpha^t = 0.75$, for the traditional SOM rule (constant horizontal line) and for the fisherman's rule (increasing curve).

$$\Delta\vec{w}_r^t = \gamma_r^t \|\vec{x}_i - \vec{w}_r^t\| \frac{\vec{w}_q^{t+1} - \vec{w}_r^t}{\|\vec{w}_q^{t+1} - \vec{w}_r^t\|}, \tag{14}$$

$$\Delta\vec{w}_r^t = \gamma_r^t \|\vec{w}_q^{t+1} - \vec{w}_r^t\| \frac{\vec{w}_q^{t+1} - \vec{w}_r^t}{\|\vec{w}_q^{t+1} - \vec{w}_r^t\|}, \tag{15}$$

In each of these four equations, the three factors from left to right are respectively the learning rate, the norm of the adaptation and its direction. The dynamical behavior of these four rules are assessed by experiments described in Section 5.

5. Experiments

In order to evaluate the learning performance of the four rules described in Section 4, several experiments on different map configurations have been tried. Sections 5.1–5.4 present the maps, the learning sets, the error criterions used to evaluate the learning quality and, finally, the learning parameters.

5.1. Maps

For the experiments, the maps have always a hexagonal neighborhood shape, i.e. the neurons are disposed like the cells of a honeycomb. As a result, the links between the neurons all have the same unitary length and form equilateral triangles. These links are used to compute the fixed grid distances $d(r, q)$ between the neurons, with a shortest path algorithm. Therefore, such distances give not exactly the same values as if they were computed with the Euclidian distance in the grid. In the worst case, the difference between both measures does not exceed 15%.[2] As the hexagonal neighborhood shape seems more natural than the rectangular one in an Euclidian space, no experiments were made with the latter.

Regarding the neighborhood function, the two traditional options are tested: the bubble function (Eqs. (3) and (4)) and the Gaussian kernel (Eq. (5)).

Finally, just before the learning phase, two methods are also tried for the initialization of the maps: random initialization and linear initialization. The first one consists in choosing randomly as much vectors in the learning set as there are neurons in map. The result is a map well placed inside the learning cloud but heavily crumpled. The second method is more complex but also more judicious. All learning patterns are used to compute the principal components of the learning cloud (for more details about the Principal Component Analysis, see Jolliffe (1986)). Next, the map is initialized in the plane spanned by the two first components, i.e. the ones associated with the largest eigenvalues of the covariance matrix of the learning set. Contrary to the first method, the neurons are not always right inside the cloud but in return the map is well unfolded.

5.2. Learning sets

Two artificial learning sets were chosen for the

[2] The worst case happens when the shortest path follows the same number of 'diagonal' links (at 30 or 60°) and vertical/horizontal ones, leading to a Euclidean distance which is equal to the length of the shortest path multiplied by cos(30) = 0.866.

Table 1
Rules

	Non-recursive	Recursive
Radial	Rule 12 (Fig. 2, left) Traditional SOM	Rule 13 (Fig. 3, right) Hybrid
Non-radial	Rule 14 (Fig. 3, left) Hybrid	Rule 15 (Fig. 2, right) fisherman's rule

experiments. The first one is a spiral (Fig. 4) embedded in a 2D space. The second one (Fig. 5) is called a 'half-cube', because it contains three faces of a cube, which are placed so that they form a corner. Both learning sets comprise 2400 patterns with some Gaussian noise (absolute variance = 0.01). These simple learning sets are chosen in order to test the behavior of the two classical SOM architecture (1D string and 2D grid).

The maps trained on the learning sets contain 48 neurons (2% of the number of patterns). The width and length of the maps are fitted to the shape of the learning cloud: 48 by 1 for the spiral and 12 by 4 for the cube corner (4 × 4 neurons per face).

5.3. Error criterions

When using a SOM, the quality of the learning phase may be seen from two points of view: the quality of the vector quantization or the quality of the topographic mapping.

The quantization performance is measured as usually by the quantization error. Formally, if the coding and decoding operations are defined as

$$\mathrm{cod}(\vec{x}_i) = \arg \min_r \|\vec{x}_i - \vec{w}_r\| \tag{16}$$

$$\mathrm{dec}(r) = \vec{w}_r \tag{17}$$

then the vector quantization error is the quadratic error between the initial learning patterns and the corresponding vectors after coding and decoding. It can be written as

$$E_{VQ} = \sum_i^L \|\vec{x}_r - \mathrm{dec}(\mathrm{cod}(\vec{x}_i))\|^2, \tag{18}$$

where index i traverses the whole learning set. The error may be normalized by the number of patterns L or by the variance of the learning set.

Another criterion to assess the quantization quality consists in counting the percentage of 'lost units', i.e. neurons that are never BMU for any pattern of the learning set. Lost units are often neurons placed outside the learning cloud; they are wasted resources for the quantization.

Regarding the topographic mapping, the topology preservation may be measured by numerous criterions (Bauer, Herrmann, & Villmann, 1999; Goodhill & Sejnowski, 1996; Bauer & Pawelzik, 1992; Villman, Der, Hermann, & Martinetz, 1997). Considering the large number of experiments made to evaluate the maps, the criterions giving non scalar results are discarded, i.e. scatter plot (Demartines & Hérault, 1997) or function plot (Bauer & Pawelzik, 1992). Three scalar criterions are retained.

The first one is the Topographic Error, mentioned in (Kiviluoto, 1996). This criterion is data dependent, in the sense that it uses the learning patterns. The formal definition is as follows:

$$E_{T1} = 1 - \frac{1}{L}\sum_{i=1}^{L} l(\vec{x}_i), \tag{19}$$

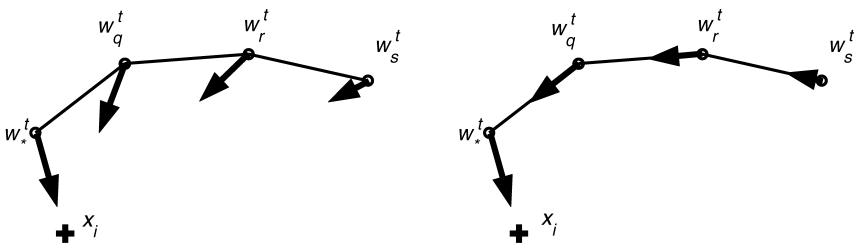

Fig. 2. Traditional SOM rule (left) and fisherman's rule (right).

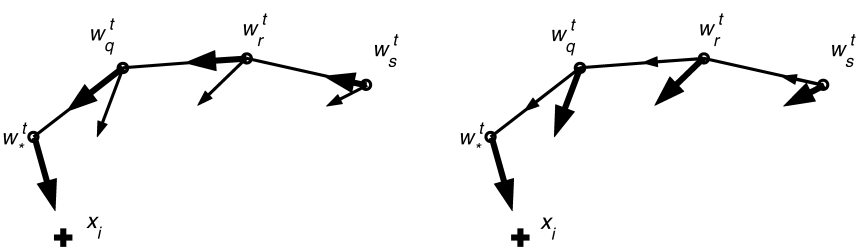

Fig. 3. Hybrid rules: non radial non recursive (left) and recursive radial (right).

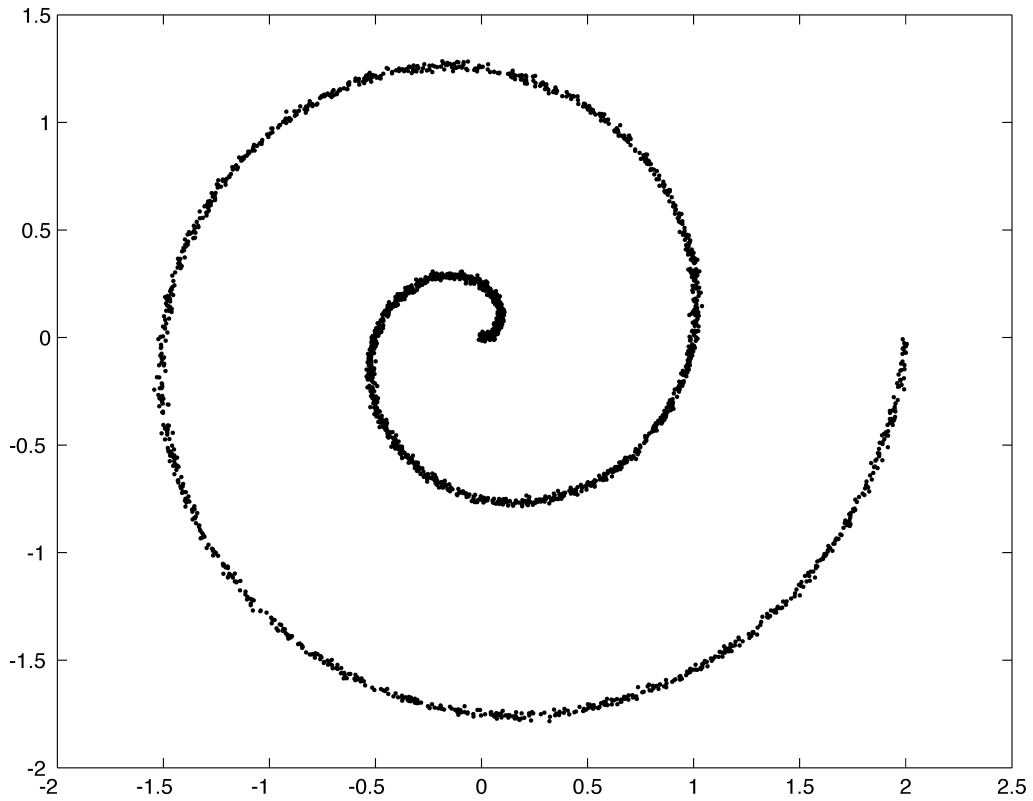

Fig. 4. Learning set: the spiral.

where $l(\vec{x}_i)$ equals 1 when the two nearest neurons from $\vec{x}_i$ are linked, i.e. are direct neighbors in the map; otherwise, $l(\vec{x}_i)$ equals 0. Intuitively, E_{T1} decreases from 1 to 0 when the two nearest neurons from a pattern are not neighbors in the map. This situation can occur for example when the map is folded on itself and if the two nearest neighbors are located on the two different layers.

The second criterion is inspired from (Jones, Van Sluyters, & Murphy, 1991). Its computation does not require to know the learning patterns. In this case, the

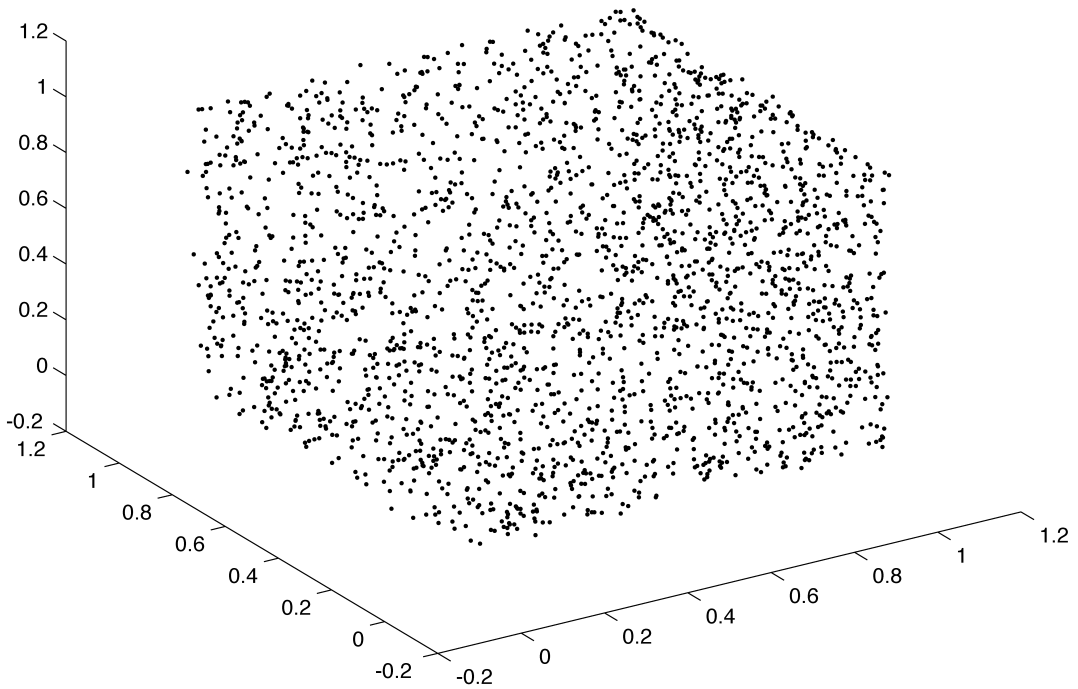

Fig. 5. Learning set: the half-cube (three faces of a cube).

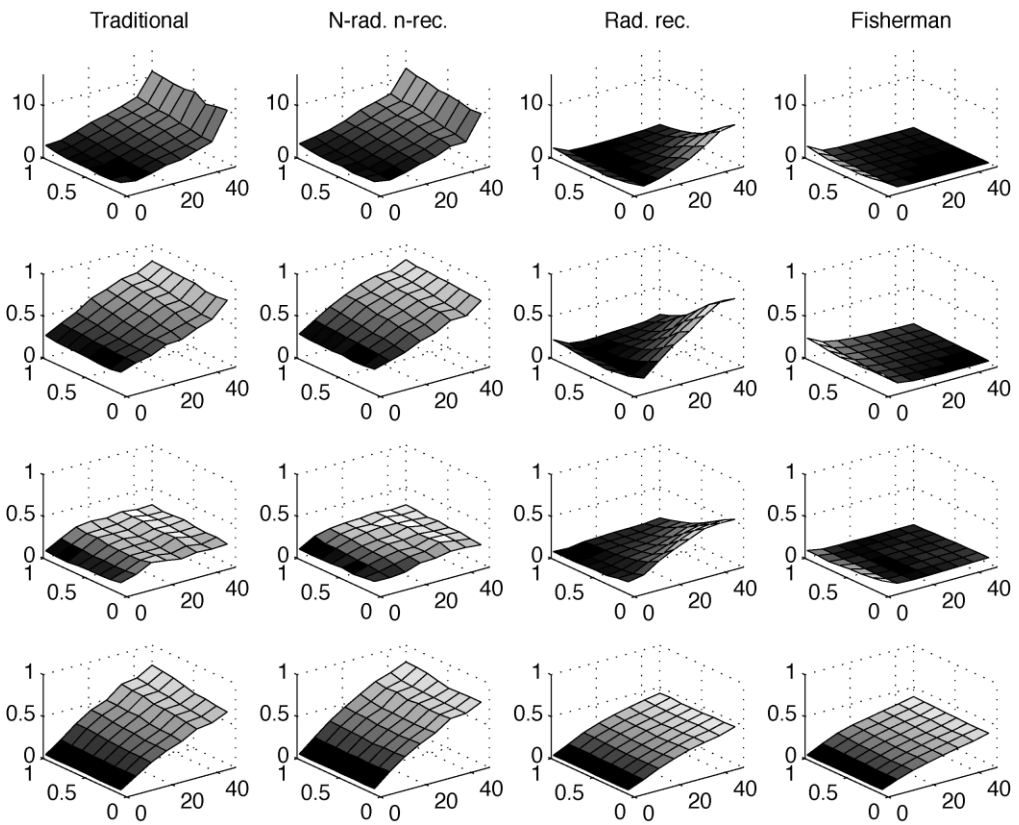

Fig. 6. Graphical comparison between the four rules: mean error surfaces are drawn for each learning rule; from left to right: the traditional rule, the non-radial non-recursive rule, the radial recursive rule and the fisherman's rule; from top to bottom: the topology preservation criterions E_{T3}, E_{T2}, E_{T1} and the vector quantization error E_{VQ}. The results are averaged for 10 000 string maps trained on the spiral learning set with randomly chosen parameters ($0 < \alpha^{t=1} < 1$ on the Y axis, $0 < \lambda^{t=1} < 48$ on the X axis), after two epochs, with random initialization.

criterion just compares if the direct neighbors of a neuron r in the grid space are also the nearest ones to r in the sense of the Euclidian distance measured in the feature space. To write this formally, suppose that neuron r has $n = |N(r)|$ direct neighbors and that function $g(r) < n$ gives the number of direct neighbors of neuron r which are also ranked in the n nearest neurons in the feature space:

$$E_{T2} = \frac{1}{P} \sum_{r=1}^{P} \frac{g(r)}{|N(r)|}. \tag{20}$$

Finally, the third criterion (Lee, Donckers, & Verleysen, 2001) resembles to the second one. The idea consists in building around each neuron r a neighborhood in the feature space defined as a sphere with radius

$$R(r) = \max_{s \in N(r)} \|\vec{w}_r - \vec{w}_s\|. \tag{21}$$

Then, for each neuron, the error criterion counts the number of neurons which are inside this Euclidian neighborhood while they should be located outside, because they are not direct neighbors of neuron r in the grid. This gives:

$$E_{T3} = \frac{1}{P} \sum_{r}^{P} |\{s | s \neq r, s \notin N(r), \|\vec{w}_r - \vec{w}_s\| < R(r)\}|. \tag{22}$$

5.4. Learning parameters

In order to test the sensitivity of the four rules to the learning parameters, several thousands of experiments were made with different values for the learning rate α^t, the neighborhood width λ^t and the number of epochs, i.e. the number of stimulations of the map by the whole learning set. Practically, for each

- learning set (spiral or half-cube),
- neighborhood function (Bubble or Gaussian),
- initialization method (random or PCA),
- number of epochs (2, 5 or 10),
- rule (classic, fisherman's, hybrid 1 and 2),

the initial parameters values are chosen randomly 10000 times between 0 and 1 for $\lambda^{t=1}$, and between 0 and 48 for $\lambda^{t=1}$. The final values are the initial ones divided by the

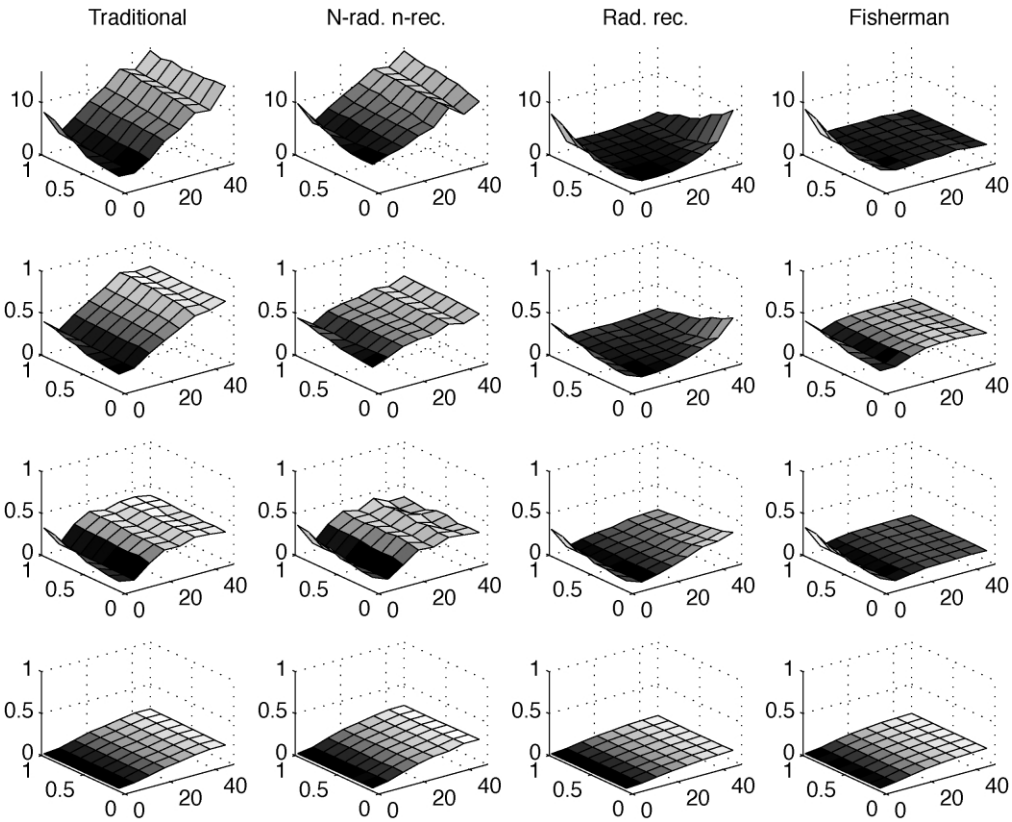

Fig. 7. Graphical comparison between the four rules: mean error surfaces are drawn for each learning rule; from left to right: the traditional rule, the non-radial non-recursive rule, the radial recursive rule and the fisherman's rule; from top to bottom: the topology preservation criterions E_{T3}, E_{T2}, E_{T1} and the vector quantization error E_{VQ}. The results are averaged for 10 000 grid maps trained on the half cube learning set with randomly chosen parameters ($0 < \alpha^{t=1} < 1$ on the Y axis, $0 < \lambda^{t=1} < 48$ on the X axis), after five epochs, with random initialization.

number of epochs and the intermediate values are interpolated by an exponential function. All these choices are maybe quite arbitrary, but even if they are not exhaustive, they cover a wide range of possibilities. Shortly said, the conducted experiments illustrate the robustness of the different rules against a naive choice of the parameters, more than 'best-case performances'.

6. Results and discussion

Due to the large number of experimental results and their difficulty to be interpreted, this section includes only a few tables and figures. For example, Figs. 6 and 7 show the averaged results of the 10 000 maps, as a function of the learning parameters. The maps are trained with the spiral (Fig. 6) and with the half-cube (Fig. 7); after a random initialization, each map ran during 2 (Fig. 6) or 5 epochs (Fig. 7), with randomly chosen learning parameters. The whole set of numerical results are available on request to the authors.

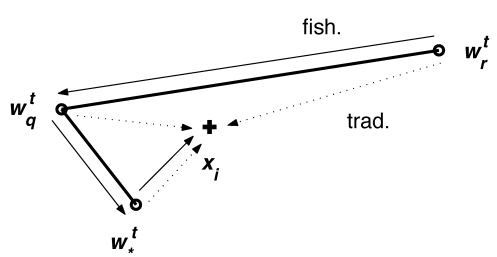

Fig. 8. Comparison between the traditional rule and the fisherman's one: the fisherman's rule unfolds better the map by moving the second order neighbor $\vec{w}_r^t$ (solid arrow) farther than the traditional rule (dotted arrow).

6.1. Random vs linear initialization

During the first epochs of the learning phase, the map has to unfold itself inside the learning cloud. Linear initialization fasten this important task. Unfortunately, PCA is not exactly a neural method, so what happens when the simpler random initialization is used instead? In this case, visible differences appear between the recursive and the non-recursive learning rules. More precisely, when the map is still heavily crumpled, just after the initialization, the following situation can often occur (Fig. 8): second order neighbors of the BMU are not behind the BMU $\vec{w}_*^t$, but they lie far beyond the pattern $\vec{x}_i$. Now, with the equations of the learning rules in mind, it is easy to see that the fisherman's rule (solid arrows) unfolds the map stronger than the traditional one (dotted arrows). Intuitively, the reason is simple: the traditional rule realizes topology preservation as

Table 2
Mean (μ) and standard deviation (σ) for 10 000 experiments; after random initialization, the string maps are trained on the spiral learning set, during 2 epochs, with the Bubble function and with randomly chosen parameters ($0 < \alpha^{t=1} < 1, 0 < \lambda^{t=1} < 48$)

Rule	μE_{T3}	μE_{T2}	μE_{T1}	μE_{VQ}	σE_{T1}	σE_{T2}	σE_{T3}	σE_{VQ}
Traditional	4.325	0.496	0.221	0.397	3.109	0.190	0.120	0.231
Non-rad. non-rec.	4.765	0.544	0.220	0.449	3.049	0.188	0.148	0.250
Rad. rec.	1.559	0.264	0.197	0.257	1.826	0.200	0.143	0.137
Fisherman	0.319	0.074	0.060	0.227	0.619	0.063	0.032	0.114

Table 3
Mean (μ) and standard deviation (σ) for 10 000 experiments; after random initialization, the grid maps are trained on the half cube learning set, during 2 epochs, with the Bubble function and with randomly chosen parameters ($0 < \alpha^{t=1} < 1, 0 < \lambda^{t=1} < 48$)

Rule	μE_{T3}	μE_{T2}	μE_{T1}	μE_{VQ}	σE_{T3}	σE_{T2}	σE_{T1}	σE_{VQ}
Traditional	27.948	0.806	0.182	0.269	15.559	0.242	0.204	0.136
Non-rad. non-rec.	27.453	0.768	0.177	0.284	15.130	0.252	0.213	0.128
Rad. Rec.	5.693	0.347	0.196	0.139	4.871	0.154	0.111	0.052
Fisherman	4.037	0.348	0.136	0.161	1.901	0.076	0.081	0.057

Table 4
Mean (μ) and standard deviation (σ) for 10 000 experiments; after random initialization, the string maps are trained on the spiral learning set, during 5 epochs, with the Bubble function and with randomly chosen parameters ($0 < \alpha^{t=1} < 1, 0 < \lambda^{t=1} < 48$)

Rule	μE_{T3}	μE_{T2}	μE_{T1}	μE_{VQ}	σE_{T3}	σE_{T2}	σE_{T1}	σE_{VQ}
Traditional	1.424	0.268	0.154	0.147	0.710	0.093	0.104	0.109
Non-rad. non-rec.	1.723	0.315	0.128	0.191	0.737	0.099	0.084	0.130
Rad. Rec.	0.674	0.167	0.158	0.108	0.592	0.096	0.102	0.073
Fisherman	0.415	0.085	0.064	0.101	0.635	0.061	0.034	0.061

Table 5
Mean (μ) and standard deviation (σ) for 10 000 experiments; after random initialization, the grid maps are trained on the half cube learning set, during 5 epochs, with the Bubble function and with randomly chosen parameters ($0 < \alpha^{t=1} < 1, 0 < \lambda^{t=1} < 48$)

Rule	μE_{T3}	μE_{T2}	μE_{T1}	μE_{VQ}	σE_{T3}	σE_{T2}	σE_{T1}	σE_{VQ}
Traditional	8.126	0.496	0.287	0.117	4.274	0.197	0.145	0.069
Non-rad. non-rec.	9.028	0.452	0.288	0.154	3.411	0.108	0.179	0.087
Rad. Rec.	2.393	0.198	0.156	0.076	2.756	0.112	0.103	0.040
Fisherman	3.384	0.307	0.120	0.097	1.932	0.076	0.085	0.048

an indirect consequence of a particular vector quantization method while the fisherman's rule tries to reach topology preservation more explicitly than the traditional rule (Section 3). Practically, the difference is clear, since the traditional rule moves the neighbors towards the learning pattern (Fig. 8), while the fisherman's rule pulls the second order neighbors towards the direct neighbors.

For the unfolding of the spiral with a string map, the best results for all criterions are performed by the fisherman's rule (Fig. 6, Table 2). The fisherman's rule and the radial recursive one work well for the grid map used with half-cube (Fig. 7, Table 5).

6.2. Gaussian vs Bubble neighborhood function

When the number of epochs grows, the recursive rules keep their performance advantages if the neighborhood is computed with the Bubble function (see Tables 2–5 for the Bubble function). As already stated, the fisherman's rule works better for the spiral and the radial recursive for the half cube.

Once the neighborhood function becomes a Gaussian kernel, the tendency is reversed: the traditional rule works better (see Tables 6 and 7). A possible explanation is that the combination of the Gaussian kernel and the recursive adaptation causes a very fast decrease of the neighborhood adaptations as the grid distance to the BMU grows. As a consequence, the recursive rules favor the vector quantization (they give a lower error than the traditional learning rule), but they are not as good with respect to topology preservation. The advantage of the traditional rule is clearly visible for the spiral (Table 6), while the best choice is not so

Table 6
Mean (μ) and standard deviation (σ) for 10 000 experiments; after random initialization, the string maps are trained on the spiral learning set, during 2 epochs, with the Gaussian function and with randomly chosen parameters ($0 < \alpha^{t=1} < 1, 0 < \lambda^{t=1} < 48$)

Rule	μE_{T3}	μE_{T2}	μE_{T1}	μE_{VQ}	σE_{T3}	σE_{T2}	σE_{T1}	σE_{VQ}
Traditional	0.039	0.009	0.008	0.574	0.174	0.033	0.017	0.291
Non-rad. non-rec.	0.273	0.081	0.022	0.586	0.254	0.062	0.026	0.297
Rad. Rec.	2.271	0.331	0.225	0.313	2.837	0.273	0.165	0.151
Fisherman	0.151	0.050	0.056	0.248	0.240	0.044	0.025	0.119

evident for the half-cube (Table 7), for which the fisherman's rule performs very well. In brief, the recursive neighborhood adaptation and the Gaussian kernel play the same role: they smooth the neighborhood adaptations around the BMU. But the combination of both techniques decreases their efficiency.

From this point of view, it would be interesting to compare the traditional rule with Gaussian kernel and the fisherman's rule with the Bubble function. Unfortunately, this raises some practical difficulties. Indeed, the slope of the Gaussian kernel is dynamically adjustable by tuning its width λ^t. On the contrary, the shape of the recursive adaptation is statically fixed on the basis of the distances $d(*, r)$ and the only dynamical feature consists in skipping some of the weakest adaptation with the Bubble function.

Consequently, a promising idea for future work would be the comparison between the traditional SOM learning rule with the following one:

$$\Delta\vec{w}_*^t = \alpha^t\left(\vec{x}_i - \vec{w}_*^t\right) \tag{23}$$

$$\Delta\vec{w}_r^t = \lambda^t\left(\vec{w}_q^{t+1} - \vec{w}_r^t\right), \qquad \forall r \neq *. \tag{24}$$

With respect to the traditional learning rule, the neighborhood factor γ_r^t is simplified by deleting the global learning rate α^t and by replacing the neighborhood function by λ^t. This last rule makes the vector quantization and the topographic mapping even more independent than the fisherman's rule.

6.3. Robustness

Considering the different quality criterions as a function of the learning parameters α^t and λ^t, the fisherman's rule and the radial recursive one are robust in the sense that 'extreme' values of the learning parameters do not degrade dramatically the learning efficiency. Actually, the recursive rules give error surfaces which are relatively flat and present wide minima (see Figs. 6 and 7). This is an advantage when the algorithm has to be parameterized by an inexperienced user. Not only the averaged results are satisfying, but the standard deviations are also lower (see Tables 2–5 and 7).

The non-recursive rules outperform the recursive ones when the goal consists in minimizing simultaneously both quantization and topology errors. But this objective often needs a careful choice of the learning parameters.

6.4. Speed

Until here, the relative speeds of the four different learning rules have not been mentioned. Indeed, speed is not a real problem compared to quality issues like quantization error and topology preservation. Nevertheless, it is worth to notice that the fisherman's rule with the Bubble function behaves like the traditional rule with a Gaussian kernel. From a computational point of view, this avoids to compute for each adaptation an exponential function depending on the dynamical parameter λ^t. Actually, the fisherman's rule replaces this demanding task by the computation of the grid distances $d(q, r)$, which can be achieved statically before learning.

7. Conclusion

This paper shows on simple examples that some changes can be made to the traditional SOM learning rule. Indeed, well chosen modifications do not alter the subtle mix of vector quantization and topographic mapping performed by the traditional SOM algorithm. On the contrary, the proposed learning rules try to supervise each subtask as independently as possible, giving the possibility to put the emphasis on one or the other subtask of the SOMs.

Table 7
Mean (μ) and standard deviation (σ) for 10 000 experiments; after random initialization, the grid maps are trained on the half cube learning set, during 2 epochs, with the Gaussian function and with randomly chosen parameters ($0 < \alpha^{t=1} < 1, 0 < \lambda^{t=1} < 48$)

Rule	μE_{T3}	μE_{T2}	μE_{T1}	μE_{VQ}	σE_{T3}	σE_{T2}	σE_{T1}	σE_{VQ}
Traditional	2.793	0.364	0.196	0.285	1.248	0.132	0.125	0.102
Non-rad. non-rec.	4.512	0.465	0.153	0.300	1.426	0.071	0.187	0.098
Rad. Rec.	4.078	0.296	0.170	0.139	4.026	0.155	0.098	0.043
Fisherman	3.149	0.330	0.124	0.161	1.067	0.075	0.059	0.048

From a theoretical point of view, the proposed rules introduce the concept of recursive neighborhood adaptation, inspired by physical ideas. Concerning speed, the recursive neighborhood adaptation allows the algorithm to mimic the properties of a Gaussian neighborhood function, without the need to compute Gaussian kernels for each neuron at each pattern presentation.

Considering the performances, the new learning rules provide a robust behavior against the choice of the learning parameters. The recursive neighborhood adaptation works particularly well for the initial unfolding of the map.

Acknowledgments

This work was realized with support of the 'Ministère de la Région wallonne', under the 'Programme de Formation et d'Impulsion à la Recherche Scientifique et Technologique'. The authors wish to thank the reviewers for their judicious and helpful comments.

References

Bauer, H.-U., & Pawelzik, K. R. (1992). Quantifying the neighborhood preservation of self-organizing maps. *IEEE Transactions on Neural Networks*, *3*, 570–579.

Bauer, H.-U., Herrmann, M., & Villmann, T. (1999). Neural maps and topographic vector quantization. *Neural Networks*, *12*, 659–676.

Demartines, P., & Hérault, J. (1997). Curvilinear component analysis: A self-organizing neural network for non linear mapping of data sets. *IEEE Transactions on Neural Networks*, *8*, 148–154.

Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numerical Mathematics*, *1*, 269–271.

Goodhill, G. J., & Sejnowski, T. J. (1996). Quantifying neighbourhood preservation in topographic mappings. *Proceedings of the Third Joint Symposium on Neural Computation, University of California, San Diego and California Institute of Technology*, Pasadena, CA: California Institute of Technology, pp. 61–82.

Jolliffe, I. T. (1986). *Principal component analysis*. New York: Springer.

Jones, D. G., Van Sluyters, R. C., & Murphy, K. M. (1991). A computationnal model for the overall pattern of ocular dominance. *Journal of Neurosciences*, *11*, 3794–3808.

Kiviluoto, K. (1996). In P. IEEE Neural Networks Council (Ed.), Topology preservation in self-organizing maps (Vol. 1) (pp. 294–299). *Proceedings of International Conference on Neural Networks, ICNN'96, New Jersey, USA*.

Kohonen, T. (1982). Self-organization of topologically correct feature maps. *Biological Cybernetics*, *43*, 59–69.

Kohonen, T. (1989). *Self-organization and associative memory* (3rd ed). Berlin: Springer.

Kohonen, T. (1995). *Self-organizing maps* (2nd ed). Heidelberg: Springer.

Kraaijveld, M. A., Mao, J., & Jain, A. K. (1995). A nonlinear projection method based on Kohonen's topology preserving maps. *IEEE Transactions on Neural Networks*, *6*(3), 548–559.

Lee, J. A., Donckers, N., & Verleysen, M. (2001). Recursive learning rules for soms. In N. Allinson, Y. Hujun, L. Allinson, & J. Slack (Eds.), *Advances in self-organizing maps* (pp. 67–72). Berlin: Springer.

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

Mao, J., & Jain, A. K. (1995). Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, *6*(2), 296–317.

Pajunen, P., Hyvärinen, A., & Karhunen, J. (1996). Non-linear blind source separation by self-organizing maps. *Proceedings of International Conference on Neural Information Processing, Hong Kong*, pp. 1207–1210.

Ritter, H., Martinetz, T., & Schulten, K. (1992). *Neural computation and self-organizing maps*. Reading, MA: Addison-Wesley.

Villman, T., Der, R., Hermann, M., & Martinetz, T. M. (1997). Topology preservation in self-organizing maps: exact definition and measurement. *IEEE Transactions on Neural Networks*, *8*, 256–266.