Hindawi Publishing Corporation
Advances in Artificial Neural Systems
Volume 2010, Article ID 142540, 10 pages
doi:10.1155/2010/142540

Research Article

A Sequential Algorithm for Training the SOM Prototypes Based on Higher-Order Recursive Equations

Mauro Tucci and Marco Raugi

Department of Electrical Systems and Automation, University of Pisa, via Diotisalvi 2, 56126 Pisa, Italy

Correspondence should be addressed to Mauro Tucci, mauro.tucci@dsea.unipi.it

Received 29 July 2010; Accepted 27 November 2010

Academic Editor: Songcan Chen

A novel training algorithm is proposed for the formation of Self-Organizing Maps (SOM). In the proposed model, the weights are updated incrementally by using a higher-order difference equation, which implements a low-pass digital filter. It is possible to improve selected features of the self-organization process with respect to the basic SOM by suitably designing the filter. Moreover, from this model, new visualization tools can be derived for cluster visualization and for monitoring the quality of the map.

1. Introduction

The self-organizing map, with its variants, is one of the most popular neural network algorithms in the unsupervised learning category [1–3]. The topologically preserving mapping from a high dimensional space to a low dimensional grid formed by the SOM finds a very wide range of applications [4–6]. The SOM consists in an ordered grid of a finite number of cells $i = 1 \cdots D$ located in a fixed, low-dimension output array, where a distance metric $d(c, i)$ is defined. Each unit is associated to a model vector (or weight) $\mathbf{m}^i \in \Re^n$ that lives in the same high dimensional space of the input patterns $\mathbf{r} \in \Delta \subset \Re^n$, where Δ is the dataset to be analyzed. The distribution of the model vectors, after the training, resembles a vector quantization of the input data with the additional important feature that model vectors tend to assume, in the input space, the same topological arrangement of the correspondent cells in the predefined output grid so that the topology of the data is reflected into the lattice. In general, the learning is based on a competitive paradigm, where for each input presented to the map a best matching unit is selected by maximizing the similarity between the input and model vectors. Successively, the winning unit and its topological neighbours are adaptively updated in order to increase the matching with the input. The incremental learning algorithm of the SOM has been established heuristically following this paradigm, where the matching of the models with the input is increased by moving *all* the vectors in the direction of the present input sample

$$\mathbf{m}^i(t+1) = \mathbf{m}^i(t) + h_{ci}(t)\Big(\mathbf{r}(t) - \mathbf{m}^i(t)\Big), \quad i = 1 \cdots D. \tag{1}$$

The update step for the cell i is proportional to a time-dependent smoothing kernel function $h_{ci}(t)$, and c is the winner node

$$c = \underset{i}{\operatorname{argmin}}\left\{\left\|\mathbf{r}(t) - \mathbf{m}^i(t)\right\|\right\}. \tag{2}$$

The model vector represents the center of the receptive field of the cell. A widely applied kernel is written in terms of the Gaussian function centered over the winner node

$$h_{ci}(t) = \mu(t) \cdot \exp\left(-\frac{d(c,i)^2}{2\sigma^2(t)}\right). \tag{3}$$

The scalar function $h_{ci}(t)$ has a central role in the learning process defined by (1), and a large part of the literature on SOM is focused on the theoretical and practical aspects of the different possible implementations of this kernel function.

The SOM learning rule (1) can be interpreted as an approximate gradient descent of a cost function related to the quantization error [7], but it does not possess an exact cost function for continuous input distributions. In [3] it is shown that in order to better minimize a distortion measure the one step rule (1) may be modified formulating an improved but still approximate minimizing step. Equation (1) can be related to the asymptotic point density of the SOM or the so called magnification factor, which relates the density of the SOM weights to the density of the input data [8]. In [9], equation (1) is slightly modified in a convex or concave nonlinear expression achieving a certain control on the magnification factor. In [10] there is a similar nonlinear modification of the learning rule, but the focus is on time series reconstruction. In [11] a rival model penalization is introduced in the SOM, where the rivals are identified as those cells that are not direct neighbours of the winner, except their model vectors are closer to the input than those of the topological neighbours of the winner. For these rival cells, the kernel function assumes negative values so that through (1) their models are moved away from the input. In [12], a time invariant learning rate is proposed, and in [13] a long-term depression of the learning rate is proposed, but with a repulsive kernel function that leads to a novelty detector. Other works are related to an automatic or adaptive generation of the parameters $\mu(t)$ and $\sigma(t)$ [14–16]. In the simple approach of [14] $\mu(t)$ and $\sigma(t)$ are obtained as functions of the worst distance, registered during the training, between the winner and the input data.

The common feature of all this SOM variants is that the structure of (1) is maintained almost unchanged while different types of $h_{ci}(t)$ functions are adopted. Then, they have different rules to determine the length of the update step, while the direction of the update step is always imposed by the input. Consequently, at each step the entire map moves toward the presented sample (in [11] some rival models are moved in the opposite direction), without any memory of the previous update directions. This behaviour has theoretical support only in the final convergence phase of the learning [5, 8] when it determines a good local statistical accuracy, and it controls the magnification factor. However, during the first phase (when the width of the kernel is shrinking and the global topology of the input data is learned), a smoother change of the update step direction than the one forced by (1) would be a desired feature.

To obtain this feature, in this paper a novel SOM model is proposed where the learning is accomplished by means of a higher-order linear difference equation that implements a predefined tracking filter. In the proposed model, each cell contains a set of memory vectors in order to take trace of the past positions of the model vectors and of the inputs. The update direction is defined by the dynamic of a properly defined filter that guides the movement of the model vectors as a combination of the past vectors. In that way, the proposed model gives a general framework to define different training strategies, where the basic SOM is a particular case. Moreover the proposed method has some new useful additional visualization and data analysis capabilities. In particular each cell can give more information such as local density and learning trajectories when higher order filters are used.

2. Model Description

2.1. Learning Filter. From (1), the new weight depends explicitly on the present input and on the present weight position (it is in fact a convex combination of these two vectors), and there is not a direct dependence on past input and weight values.

Our idea is to add more degrees of freedom to the incremental learning equation of the SOM, in order to improve the relaxation process that takes place during the self-organization of the model vectors.

This is obtained by adding to the neuron model a "*memory*" of the past values of the inputs and model vectors of the cell. The new weight vector is calculated as a linear combination of the vectors in the memory. To properly define this linear combination, it is necessary to employ a stable, discrete-time filter, which in the following will be referred as *learning filter (LF)*. Using the Z-transform formalism, the learning filter is defined by means of its transfer function $G(z)$

$$G(z) = \frac{b_1 z^{N-1} + \cdots + b_N}{z^N - a_1 z^{N-1} - a_2 z^{N-2} \cdots - a_N}. \tag{4}$$

The filter $G(z)$ is implemented by means of a linear difference equation described by the LF coefficients $\mathbf{a} = [a_1, \ldots, a_N]^T$, $\mathbf{b} = [b_1, b_2, \ldots, b_N]^T$ as:

$$m(t) = \sum_{k=1}^{N} b_k r(t-k) + a_k m(t-k), \tag{5}$$

where $r(t-k)$ represents the input sequence and $m(t-k)$ represents the output sequence. A rigorous requirement of the LF is that it has to be a low-pass filter at least of type 1, that is, with unitary static gain: $G(z=1) = 1$. This also means that for a constant input sequence $r(t) = r$ the error sequence $r - m(t)$ will tend to zero as time goes to infinity. Hence, the limit $\lim_{t\to\infty} m(t) = r$ holds and the LF can track constant inputs with zero error. Guidelines for the design of the LF will be described in subsequent sections.

2.2. Proposed SOM Algorithm

2.2.1. Neuron Model. In order to introduce the dynamic of the LF in the SOM, we associate two matrices (composed by N column) to each element of index i of the map vectors, $\mathbf{R}^i = [\mathbf{r}_1^i, \ldots, \mathbf{r}_N^i]$ and $\mathbf{M}^i = [\mathbf{m}_1^i, \ldots, \mathbf{m}_N^i]$, where $\mathbf{m}_k^i \in \Re^n$, $\mathbf{r}_k^i \in \Re^n$. The columns of $\mathbf{M}^i$ represent the sequence of the last N values of the weight vector of the cell i, while the columns of $\mathbf{R}^i$ take trace of the last N values of the input $\mathbf{r}(t) \in \Re^n$ *to the cell* i. Note that $\mathbf{r}(t)$ in (1) represents the sequence of samples randomly selected from the input distribution during the learning, which are presented to the whole map. Then, in the basic SOM all the cells receive the same input at each time step. Conversely, in the proposed model each cell contains a personalized input sequence $\mathbf{R}^i$, related to the winning frequency of the cell.

The $n \times N$ memory matrices $\mathbf{R}^i$ and $\mathbf{M}^i$ represent the memory added to the neuron model in order to take trace of the past events. Each cell of the proposed SOM calculates the value of the model vector at time t as a linear combination of the memory vectors at time t, through the coefficients of the LF

$$\begin{aligned}\mathbf{m}^i(t) &= \sum_{k=1}^{N} b_k \mathbf{r}_k^i(t) + a_k \mathbf{m}_k^i(t) \\ &= \left[\mathbf{R}^i(t) \vdots \mathbf{M}^i(t)\right] \begin{pmatrix} \mathbf{b} \\ \mathbf{a} \end{pmatrix} = \mathbf{Q}^i(t) \cdot \mathbf{g},\end{aligned} \tag{6}$$

where the $n \times 2N$ matrix $\mathbf{Q}^i(t) = [\mathbf{R}^i(t) \vdots \mathbf{M}^i(t)]$ represents the whole memory of cell i, while the $2N$ column vector $\mathbf{g} = \begin{pmatrix} \mathbf{b} \\ \mathbf{a} \end{pmatrix}$ contains the LF coefficients. Hence, the model vector $\mathbf{m}^i(t)$ is directly calculated—from a given memory matrix $\mathbf{Q}^i(t)$ and the LF coefficients **g.**

2.2.2. Network Training Procedure for the Proposed Model. Given the vectorial input data to analyze $\mathbf{r} \in \Delta \subset \Re^n$, we first create the output grid array of the SOM, and then choose the neighbourhood arrangement, defining the distance function $d(c, i)$ between two cells of indexes c and i. Then, we design the linear difference equation of order N that drives the learning process, and suitably define the coefficients $\mathbf{a} = [a_1, \ldots, a_N]^T$, $\mathbf{b} = [b_1, b_2, \ldots, b_N]^T$ of the numerator and denominator of the low-pass LF $G(z)$ in (4).

Initial values for the memory vectors $\mathbf{Q}^i(0)$ can be selected at random, and the so-called linear initialization, that uses the principal axes of the input distribution, can give some benefit as in a classical SOM.

When a new input sample $\mathbf{r}(t)$ is presented to the map during the training the winner unit c is selected as the node that has the nearest model vector $\mathbf{m}^i(t)$ to the input sample $\mathbf{r}(t)$

$$c = \underset{i}{\operatorname{argmin}} \left\{ \left|\left| \mathbf{r}(t) - \mathbf{m}^i(t) \right|\right| \right\}. \tag{7}$$

The receptive field of the cell i is given by the Voronoi region of the model vector

$$\mathbf{m}^i(t) = \mathbf{Q}^i(t) \cdot \mathbf{g}. \tag{8}$$

Then, the memory of the cells needs to be updated in order to take trace of the new sample. If each neuron is considered as a standalone filter, without neighborhood collaboration, the update of the memory matrices is a straightforward one-step time shift of the memory vectors

$$\begin{aligned}\mathbf{R}_+^i(t) &= \left[\mathbf{r}(t), \mathbf{r}_1^i(t), \ldots, \mathbf{r}_{N-1}^i(t)\right], \\ \mathbf{M}_+^i(t) &= \left[\mathbf{m}(t), \mathbf{m}_1^i(t), \ldots, \mathbf{m}_N^i(t)\right], \\ \mathbf{Q}_+^i(t) &= \left[\mathbf{R}_+^i(t), \mathbf{M}_+^i(t)\right].\end{aligned} \tag{9}$$

To include the neighborhood collaboration in our model, we consider the following update expressions:

$$\begin{aligned}\mathbf{R}^i(t+1) &= \mathbf{R}^i(t) + h_{ci}(t)\left(\mathbf{R}_+^i - \mathbf{R}^i(t)\right), \\ \mathbf{M}^i(t+1) &= \mathbf{M}^i(t) + h_{ci}(t)\left(\mathbf{M}_+^i - \mathbf{M}^i(t)\right),\end{aligned} \tag{10}$$

and the global memory matrix

$$\mathbf{Q}^i(t+1) = \mathbf{Q}^i(t) + h_{ci}(t)\left(\mathbf{Q}_+^i(t) - \mathbf{Q}^i(t)\right), \tag{11}$$

where the Gaussian neighborhood function is calculated as

$$h_{ci}(t) = \mu(t) \cdot \exp\left(-\frac{d(c,i)^2}{2\sigma^2(t)}\right). \tag{12}$$

In that way, only the memory of the winner and its topological neighbors have to be updated, whereas the memory of the units that are far from the winner remain almost unchanged.

2.2.3. Update Direction and Convergence. The neighbour collaboration is accomplished by means of an adaptive update of the "memory" of the neurons. Neurons that do not recognise the input pattern will not update their memory, while the memory of the neuron that recognise the pattern is updated together with the memory of its topological neighbours. It is known that in order to have the self-organization of the model vectors the update has to increase the similarity with the input pattern [3]. In our scheme, the model vectors are a linear combination of the memory vectors, as the update direction of the model vectors is determined by (6) and (11). Substituting (11) in (6), we have the following implicit expression of the updated prototypes:

$$\mathbf{m}^i(t+1) = \mathbf{m}^i(t) + h_{ci}(t)\left(\mathbf{m}_+^i(t) - \mathbf{m}^i(t)\right). \tag{13}$$

Then, the model vector is updated in the direction of $\mathbf{m}_+^i(t)$, which is the next value of the LF output in the case of the absence of neighbourhood cooperation

$$\begin{aligned}\mathbf{m}_+^i(t) &= \mathbf{Q}_+^i(t) \cdot \mathbf{g} \\ &= b_1 \mathbf{r}(t) + a_1 \mathbf{m}^i(t) + \sum_{k=2}^{N-1} b_k \mathbf{r}_k^i(t) + a_k \mathbf{m}_k^i(t).\end{aligned} \tag{14}$$

It is central to point out the similarity of the implicit update expression (13) of our algorithm with the classic SOM update rule (1). In the classic SOM update rule (1), the target vector is represented by the randomly selected input pattern $\mathbf{r}(t)$, which is the same for all the cells, whereas in the update expression (13) the same role is played by the vector $\mathbf{m}_+^i(t)$, which represents a target pattern different for each cell, defined as a linear combination of the memory vectors and the actual input sample and model vector. In particular this combination is given by an LF that is designed in order reduce the error $\mathbf{r}(t) - \mathbf{m}_+^i(t)$ to zero in the case of a continuously repeated input sample $\mathbf{r}(t) = \mathbf{r}$. In this sense, the similarity of the model vector with the input sample

is increased when the model vector moves in the direction defined by (13). A necessary requirement of every kind of LF to have this property is that they need to be at least of type 1 [17]. This requirement can be translated in a constraint on the coefficients of the LF $G(z)\mathbf{a} = [a_1, \ldots, a_N]$, $\mathbf{b} = [b_1, \ldots, b_N]$. If an LF is of type 1, then

$$\sum_{k=1}^{N} b_k + \sum_{k=1}^{N} a_k = 1, \tag{15}$$

which is equivalent to require a unitary static gain $G(1) = 1$. This means that the linear combination in (6) is an affine combination of the vectors $\mathbf{r}_k^i, \mathbf{m}_k^i, k = 1 \cdots N$ and, in the case that $a_i, b_i \geq 0$, it becomes a convex combination. This yields some possible geometrical interpretations of (6). In the classic SOM, the space for the weight update is the segment between the weight and the present input sample. In the proposed scheme, the new weight can move in the higher dimensional space given by the affine combination, defined by the LF coefficients of the $2N$ vectors $[\mathbf{r}_1^i, \ldots, \mathbf{r}_N^i]$ and $[\mathbf{m}_1^i, \ldots, \mathbf{m}_N^i]$.

Another important remark is related to the fact that the target vector sequence $\mathbf{m}_+^i(t)$ can be seen as the output of a filter $G(z)$, where the input is related to the stochastic sequence $\mathbf{r}(t)$. Therefore, using an LF with static gain $G(1) = 1$ corresponds to require a filter which maintains the first-order moment of the stochastic process $\mathbf{r}(t)$, which is also the mean value of the input distribution. In fact, it is known that the mean value of the output sequence $\mathbf{m}_+^i(t)$ of a linear filter $G(z)$ is related to the mean value of the input stochastic sequence $\mathbf{r}(t)$ by the following relationship:

$$E\left\{\mathbf{m}_+^i(t)\right\} = G(1) \cdot E\{\mathbf{r}(t)\}, \tag{16}$$

where the symbol $E\{\cdot\}$ represents the expectation. As a result the static properties of the input distribution are well represented by the sequence of target vectors $\mathbf{m}_+^i(t)$, and moving the prototypes as in (13) reduces the distance to the input distribution. Consequently, the *local* stability and convergence of the proposed model are assured by the stability and static gain features of the learning filter, while the global behaviour of the map follows the same heuristic principle of the basic SOM. In particular, to have the *global* convergence of the model vectors it is necessary that $h_{ci}(t) \to 0$, when $t \to \infty$, and it is necessary to respect the well-known convergence conditions of the Robbins-Monro stochastic approximation method [2].

In the proposed model, the function $h_{ci}(t)$ can be defined similarly to other self organizing paradigms as vector quantization, neural gas [18], or SOM, obtaining analogous self organization behaviours.

As a final result of the proposed model a new self organizing algorithm is defined by (6), (7), and (11). Very stable and robust self organization activities of the model vectors are observed when executing (6), (7), and (11) if the LF $G(z)$ is carefully designed.

2.3. Guidelines to Design the Learning Filter (LF). In this section we describe general principles for a useful design of the LF. We first describe the LF that gives the basic SOM. To reproduce the classical SOM in our model, we have to consider, for example, the kernel function

$$h_{ci}(t) = \mu(t) \exp\left(-\frac{d(c,i)^2}{2\sigma^2(t)}\right), \tag{17}$$

where we consider the following annealing schemes [3]:

$$\begin{aligned} \sigma(t) &= \frac{\sigma_{\max} - (\sigma_{\max} - \sigma_{\min})t}{T}, \quad t < T_1, \\ \sigma(t) &= \sigma_{\min}, \quad t \geq T_1, \\ \mu(t) &= \frac{\mu_{\max} - (\mu_{\max} - \mu_{\min})t}{T}, \quad t < T_2, \\ \mu(t) &= \frac{\mu_{\min} \cdot T}{t}, \quad t \geq T_2, \end{aligned} \tag{18}$$

and T_1, T_2 are, respectively, the decreasing times of the kernel width and of the learning rate.

The exact SOM equations are obtained from (6), (7), and (11) considering the following first-order learning filter LF1:

$$G(z) = \frac{\alpha}{z + \alpha - 1}, \tag{19}$$

where $\alpha \in (0, 1)$ is a global "learning constant." By substituting the coefficients of the LF1 in (19) into (6) we have that (6), (7), and (11) become formally equivalent to the basic SOM. Hence, the basic SOM algorithm is included in our framework and it can be obtained by using the first-order LF1. The learning constant α, which determines the bandwidth of the learning filter, in this contest acts as a global attenuation constant of the learning factor $\mu(t)$.

A second-order learning filter LF2 has the form

$$G(z) = \frac{b_1 z + b_0}{z^2 - a_1 z - a_0}. \tag{20}$$

Filters coefficients can be chosen so that the learning filter (20) is a low-pass filter with the same normalized cut-off frequency $\omega_0 \in (0, 1)$ (-3 db band) of filter (19). In this case, the same global attenuation of the learning factor can be expected and the advantage of filter (20) on the classical SOM (19) is the enhanced filtering action obtainable by the higher-order update equation.

Due to their well-known "optimal" characteristics of gain and bandwidth reported in circuit analysis, we use discrete low-pass Butterworth filters [17] as LF in many numerical experiments obtaining good results. With this choice, the only free parameters of the LF are the order N of the filter and the cut-off frequency ω_0 [17]. Figure 1 shows two different 20×20 maps trained with two different values of ω_0, by using a second-order low-pass Butterworth filter. The input distribution is clustered in four regions with uniform distribution in $\Re^2$ as in Figure 1. Trained model vectors are the nodes of the depicted grid. With a high value of $\omega_0 = 0.8$, the final distribution of model vectors in Figure 1(a) shows a better fitting (or regression) of the input distribution. For a lower value of $\omega_0 = 0.1$, the map has a more interpolative

behaviour shown in Figure 1(b). This is due to the fact that with smaller values of ω_0 the learning rate factor decreases more rapidly, while the decrease of the neighbourhood width is unchanged. Then, it is possible to select suitable values of ω_0 depending on the particular data to be analyzed with the SOM. For example, usually a more interpolative map is desired for cluster analysis. In a basic SOM, this can be achieved by generating different maps changing the initial values of the learning factor $\mu(t)$ or by taking a greater final value of the neighbourhood width $\sigma(t)$. In the proposed framework, we utilize the learning velocity constant ω_0, which defines the band of the learning filter, to obtain maps with different interpolative behaviours, instead of operating on the annealing scheme of the learning rate factor $\mu(t)$.

3. Visualization Tools

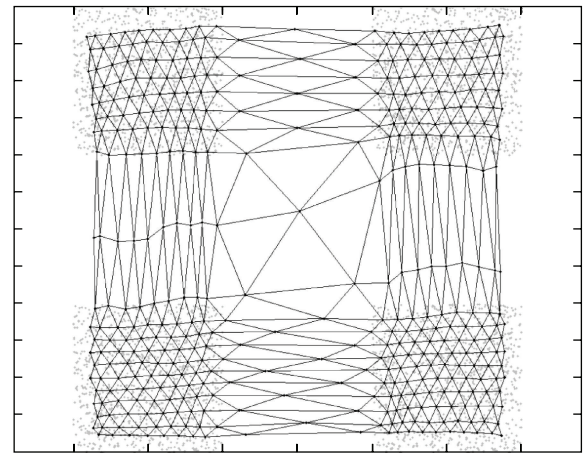
The proposed method add a new feature to the SOM algorithm, by taking into account the old values of the model vector and of the input during training and using this information when updating the model vectors. When the training is completed for each cell i, we have an $n\times 2N$ matrix $\mathbf{Q}^i = [\mathbf{r}_1^i, \ldots, \mathbf{r}_N^i, m_1^i, \ldots, m_N^i]$. The centres of the receptive fields of the cells are obtained from (6) as $\mathbf{m}^i = \mathbf{Q}^i \cdot \mathbf{g}$, but more useful information can be extracted from the memory matrices $\mathbf{Q}^i$. Vectors $\mathbf{m}_1^i, \ldots, \mathbf{m}_N^i$ can be interpreted as the last N values of the model vector during training, so that they give information on the final learning trajectory and on the velocity of the model vector of the cell i. The vectors $\mathbf{r}_1^i, \ldots, \mathbf{r}_N^i$ are related to the last N values of the input to cell i during training, so that, in general they give a rough indication of the receptive area of the cell.

By the numerous performed simulations, it comes out that, from these vectors $\mathbf{m}_1^i, \ldots, \mathbf{m}_N^i$, useful information can be extracted for visualization purposes. If the LF is of order $N \geq 2$, the following quantity can be computed for each cell after the training of the network:

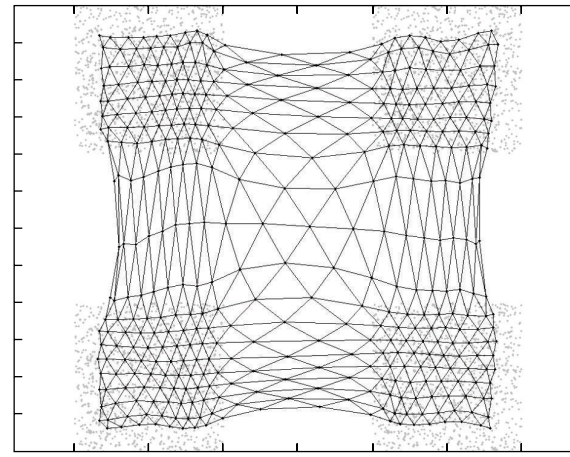
$$\mathbf{e}^i = \frac{1}{N-1}\sum_{k=2}^{N} \mathbf{m}_{k-1}^i - \mathbf{m}_k^i. \tag{21}$$

The norm of this vector gives the average lengths of the distances between subsequent vectors $\mathbf{m}_k^i$. It comes out from simulations that $\|\mathbf{e}^i\|$ is related to the point density of the input in the Voronoi region of the cell i. In particular, this norm is smaller for higher-density regions, and vice versa. When $\|\mathbf{e}^i\|$ is visualized in the output grid, defining the colour of the unit, a visualization map similar to the U-matrix [19] is obtained and we call the proposed visualization the E-matrix. The U-matrix is obtained by visualizing the distances between the model vectors of adjacent units while the E-matrix is obtained by considering information given by each cell itself, without considering the topological neighbourhood. Examples of the usefulness of this visualization tool are given in the next section.

A second visual aid on the map is obtained by considering the direction of the normalized vectors $\mathbf{e}^i/\|\mathbf{e}^i\|$, that indicate the final direction of the trajectory of the model vector during training. This direction is defined in the input space



(a) $\omega_0 = 0.8$



(b) $\omega_0 = 0.1$

Figure 1: Distribution of the model vectors of two different maps trained with two different values of the cut-off frequency ω_0, using a second order Butterworth learning filter.

and can be translated in the output map space by finding, within the neighbours of the cell i, the one whose model vector $\mathbf{e}^i$ is "pointing to". This is achieved by first finding, for the cell i, all vectors $\mathbf{d}_q^i = \mathbf{m}^q - \mathbf{m}^i$, $q \in \Lambda^i$, where Λ^i is the set of the direct neighbours of cell i, and then computing

$$u^i = \underset{q}{\operatorname{argmax}}\left(\left\langle \frac{\mathbf{e}^i}{\|\mathbf{e}^i\|}, \frac{\mathbf{d}_q^i}{\left\|\mathbf{d}_q^i\right\|} \right\rangle \right), \quad q \in \Lambda^i, \tag{22}$$

where $\langle \cdot, \cdot \rangle$ denote the dot product. Then, in the output map grid, an arrow can be traced that connects the cell i to the cell u^i, and we call the proposed visualization the A-matrix. This reveals to be a useful tool for visualizing the zones of attraction, or contrarily the empty zones of the map.

4. Numerical Results

We consider now various examples of input distributions in order to evaluate the performance of the proposed algorithm

and the effect of changing the filter order N and cut-off frequency ω_0.

To show the characteristics of the proposed model, we will consider some features of the convergence phase and compare them with data obtained with the basic SOM. To evidence the effects of the newly introduced filter, we will adopt the same kernel function $h_{ci}(t)$ by choosing the same annealing schemes for $\sigma(t)$ and $\mu(t)$.

The total number of presentations used along our experiments is 100000. The presented results show typical trends of several tests performed by the authors. To assess the quality of the map, two indexes are used: quantization error and distortion measure.

Quantization error is calculated as

$$\mathrm{QE}(\Delta) = \frac{1}{\|\Delta\|} \sum_{\mathbf{r}\in\Delta} \|\mathbf{r} - \mathbf{m}^c\|, \tag{23}$$

where $\|\Delta\|$ denote the cardinality of the input dataset. Quantization error indicates how close the model vectors are fitting the data, but does not take into account the ordering state of the map. The following empirical distortion measure $\mathrm{DM}(\sigma)$ is used:

$$\mathrm{DM}(\Delta,\sigma) = \frac{1}{\|\Delta\|} \sum_{\mathbf{r}\in\Delta} \sum_{i=1}^{D} w_{ci}(\sigma) \left\|\mathbf{r} - \mathbf{m}^i\right\|, \tag{24}$$

where $w_{ci}(\sigma) = \exp(-d(c,i)^2/2\sigma^2)$. Unlike the quantization error, the distortion measure $\mathrm{DM}(\Delta,\sigma)$ considers the SOM topology and we have $\mathrm{DE}(\Delta,\sigma) \to \mathrm{QE}(\Delta)$ when $\sigma \to 0$. The minimization of QE is the goal of vector quantization, while the SOM can be regarded as a computational intelligence algorithm that aims to minimize the distortion measure $\mathrm{DM}(\Delta,\sigma)$ for some $\sigma > 0$. It is known that the SOM only minimizes $\mathrm{DM}(\Delta,\sigma)$ approximately, whereas the effective minimization of $\mathrm{DM}(\Delta,\sigma)$ is extremely heavy numerically [3, 20, 21].

4.1. Analysis of a Gaussian Distribution. In the first experiment, we consider that the input belongs to a Gaussian cluster $\Delta_g^d \in \Re^d$ centred in the origin of the axes and with unitary variance. In this experiment, we intend to show the effects of the application of our method on the quality of the obtained maps, by using a Butterworth (BW) LF and changing the bandwidth of the filter and the filter order.

A classic 20×20 SOM with hexagonal neighbourhood has been trained where model vectors have been initialized using random initialization.

We also trained a set of maps with our method by adopting different learning filters. Training uses the same number of steps and the same learning parameters employed for the basic SOM, but the learning filter is a low-pass Butterworth filter, and a different map is trained for various values of the cut-off frequency $\omega_0 \in (0.1, 0.8)$ and filter order $N = 2, 3, 4, 5$. Each training procedure is repeated several times with different random initializations in order to avoid possible topological defects, and to obtain statistically significant results. Figures 2 and 3 show the averaged quantization error $\mathrm{QE}(\Delta_g^{10})$ and distortion measure $\mathrm{DM}(\Delta_g^{10}, 1)$ at the end of the training of each trained map, when the dimensionality of the Gaussian input is $d = 10$.

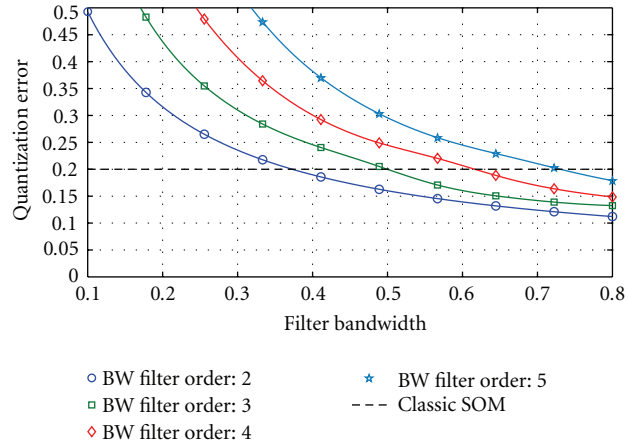


Figure 2: Averaged quantization error of trained maps with Gaussian input. Butterworth LF are used for different values of the filter order and bandwidth.

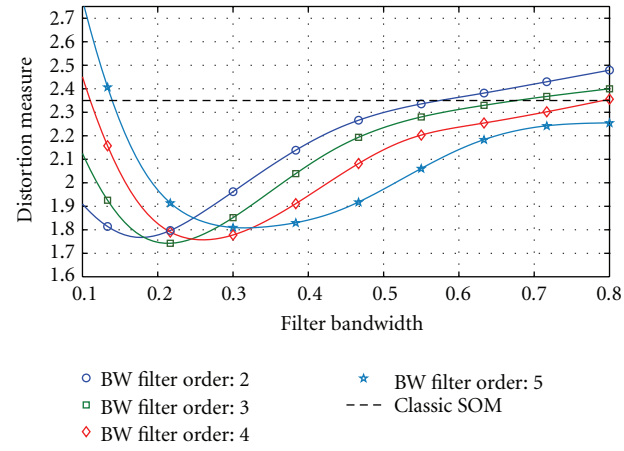


Figure 3: Averaged distortion measure of trained maps with Gaussian input. Butterworth LF are used for different values of the filter order and bandwidth.

We calculate also the percentage relative gains (PRG), shown in Figure 4, of the results obtained with the proposed method with respect to the classic SOM both for the averaged quantization error and distortion measure. The PRG are defined as

$$\begin{aligned} \mathrm{PRG}_{\mathrm{QE}} &= 100 \frac{\mathrm{QE}\left(\Delta_g^{10}\right)^{\mathrm{SOM}} - \mathrm{QE}\left(\Delta_g^{10}\right)^{\mathrm{BWLF}}}{\mathrm{QE}\left(\Delta_g^{10}\right)^{\mathrm{SOM}}}, \\ \mathrm{PRG}_{\mathrm{DM}} &= 100 \frac{\mathrm{DM}\left(\Delta_g^{10}\right)^{\mathrm{SOM}} - \mathrm{DM}\left(\Delta_g^{10}\right)^{\mathrm{BWLF}}}{\mathrm{DM}\left(\Delta_g^{10}\right)^{\mathrm{SOM}}}. \end{aligned} \tag{25}$$

It can be observed that the QE decreases when the filter bandwidth ω_0 is increased, while the QE increases when the

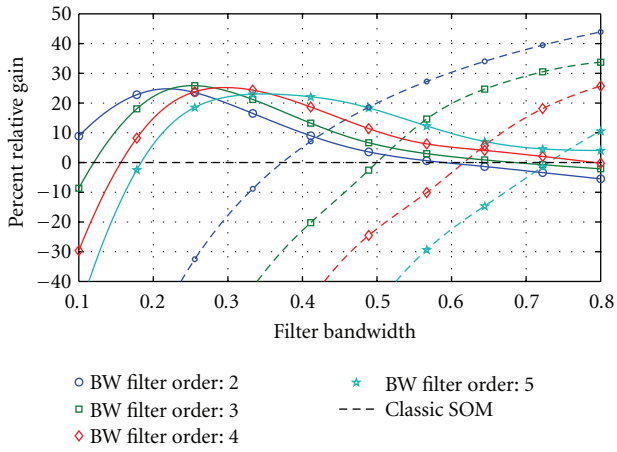


Figure 4: Percent relative gain with respect to the classic SOM quality indexes. Solid line: DM relative gain. Dashed line: QE relative gain. The line at the zero level represents the classic SOM performance.

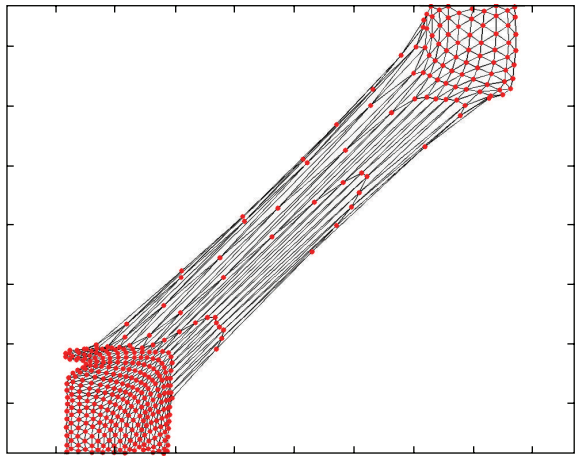


Figure 5: Model vectors of classic SOM for a clustered distribution. The distortion measure is 0.672.

filter order is increased. Hence, the better QE are obtained for high-filter bandwidth and low-filter order. For higher values of ω_0 the BW filters of order 2 and 3 can reach better QE than the classic SOM. The opposite situation can be noticed observing the DM curves. Increasing the filter bandwidth, for $\omega_0 > 0.3$, the DM decreases, while it improves by increasing the filter order. The use of the BW learning filter of order $N = 5$ leads to trained maps with better DM than classic SOM for all the used values of the filter bandwidth. The trade-off between QE and DM is a common problem that arises in the formation of SOMs. The proposed algorithm furnishes new instruments to directly control the quality of the map. Augmenting the filter order for fixed filter bandwidth produces maps with improved distortion measure but affects the quantization error, while increasing the filter bandwidth produces opposite trends. Depending on the particular analysis to be carried out, different weights can be given by the user to these features

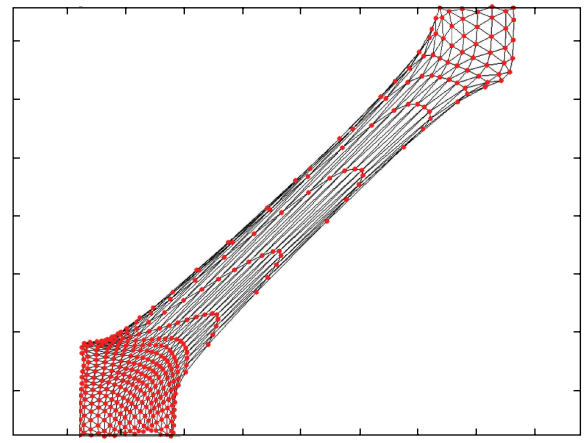


Figure 6: Model vectors of a map trained with a fourth order Butterworth learning filter. The distortion measure is 0.572.

of the map. If a map that strictly fits the data is crucial than the minimization of quantization, error is the priority and filters with high bandwidth and low order can be used. If a smoother mapping is needed than higher-order filters can be used. Furthermore, it is noticeable that every BW filter has a range of bandwidth where both QE and DM perform better than the classic SOM. In this sense, we can state that the proposed model gives better self-organization performances than the basic SOM.

If the main scope of the SOM analysis is cluster visualization, then, the use of higher-order filters brings enhanced visualization results when the E-matrix or the A-matrix visualization tools are used as shown in the following examples.

4.2. Analysis of a Clustered Distribution. In the second experiment, we consider that the input distribution is formed by two clusters in $\Re^2$, where the cluster 1 has a uniform probability density function $p(x \in 1) = 1/3$ and the cluster 2 has a uniform probability density function $p(x \in 2) = 2/3$.

A classic 20×20 SOM with hexagonal neighbourhood has been trained where model vectors have been initialized using random initialization. The final state of the model vectors after 100000 presentations is shown in Figure 5. In the trained map, an undesired curvature is present in the grid distribution of the model vectors at cluster 2. This is due to the fact that the two principal axes of the input distribution have different lengths, while the map has a 20×20 square dimension. In this case, a rectangular map may behave better, but we may be interested in searching a 20×20 map that does not exhibit this kind of distortion. In classic SOM, this can be done by adjusting the learning parameters that define $h_{ci}(t)$, then it is not a trivial task to choose these parameters in order to improve the map quality. On the other hand, in the proposed method we can obtain the desired feature by suitably selecting the filter order and cut-off frequency as shown in the previous example. By using a Butterworth filter of order $N = 4$ and cut-off frequency $\omega_0 = 0.4$, without changing the annealing schemes for the kernel function, the map in Figure 6 is obtained.

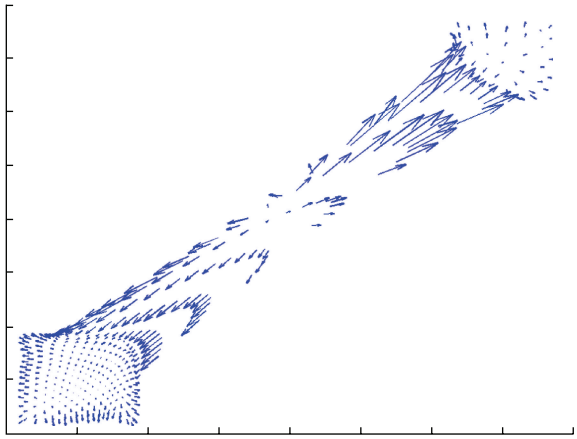


Figure 7: Trajectory vectors $\mathbf{e}_m^i$ in the input space relative to the map in Figure 6.

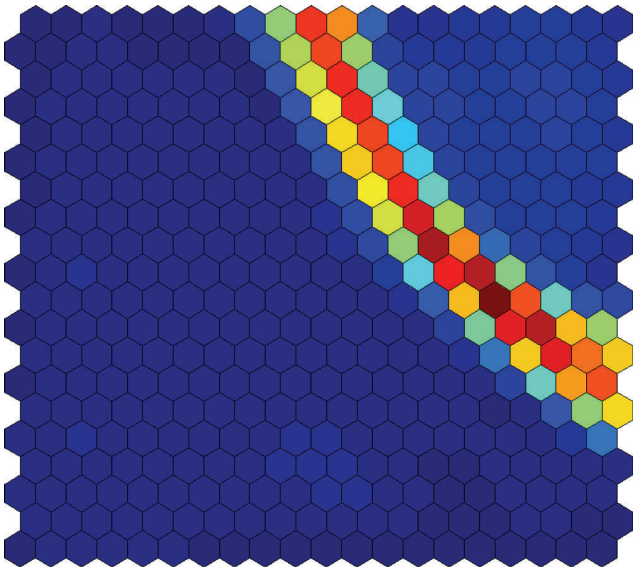


Figure 8: U-matrix of the map of Figure 6.

The model vectors shown in Figure 6 are the centers of the receptive fields $\mathbf{m}^i$ calculated as in (6). At the end of the training, we also have, for each cell i, the vectors $\mathbf{r}_k^i(t)$, $\mathbf{m}_k^i(t)$, $k = 1 \cdots N$. In this 2D example they can be visualized directly in the input space. In particular, for each cell i, the trajectory vector $\mathbf{e}^i$ is calculated as in (21) and it is shown as an arrow that starts from $\mathbf{m}^i$.

Figure 7 shows the vectors $\mathbf{e}^i$ for the map in Figure 6. The trajectory vectors clearly indicate the zones of attraction of the input space. In the classic SOM, each cell has one model vector and this kind of visualization is not possible.

Now let us consider the visualization of the map on the output grid. A popular tool for visualizing the clusters on the map is the U-matrix. We calculate the U-matrix and the E-matrix for the map of Figure 6 obtaining Figures 8 and 9, respectively. Also the A-matrix, the projections of the trajectory vectors in the output space, is displayed in Figure 9. The E-matrix in Figure 9 shows the norms of the

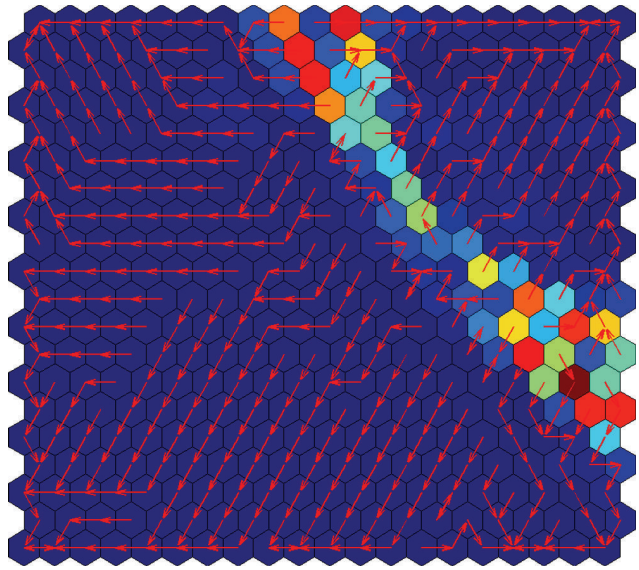
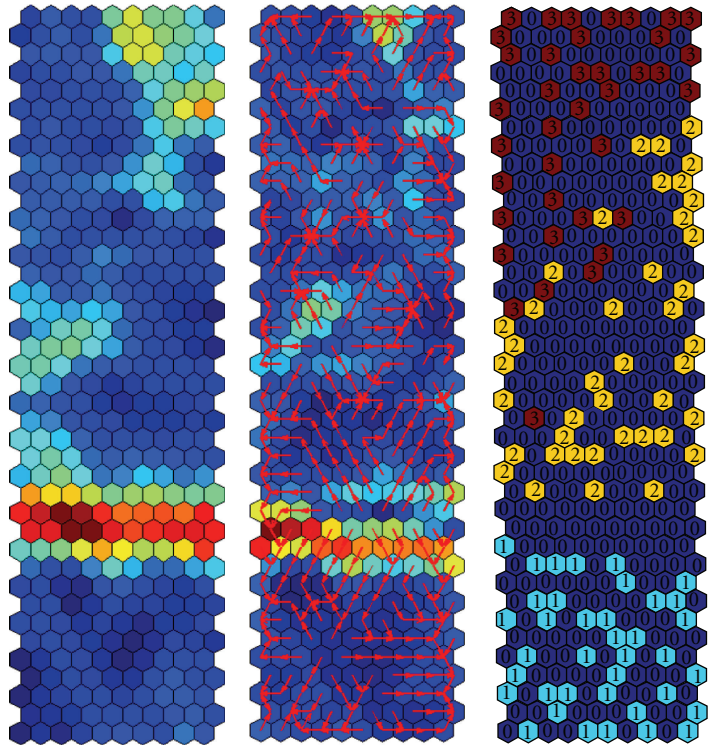


Figure 9: E-matrix (colors) and A-matrix (arrows) of the map of Figure 6.

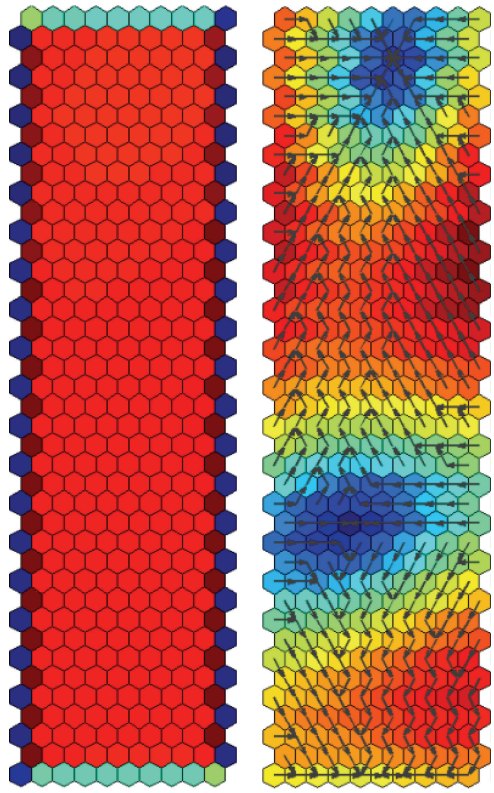
vectors $\mathbf{e}^i$ of Figure 7, and the A-matrix shows their directions projected on the topographic surface. The U-matrix gives information of the distances between neighbouring model vectors, whereas the E-matrix gives information of the final trajectory of the model vector of each cell independently. The figures show that similar information can be extracted by the two matrices, attesting that the trajectory vectors $\mathbf{e}^i$ of the interpolative units (model vectors that remain between clusters) have higher final values than units that fall inside more dense areas. This confirm that the E-matrix can be used as a cluster visualization tool. Furthermore, it has additional features with respect to the U-matrix since some information on the map quality can be derived from E-matrix observing how the model vectors fit the underlying distribution.

4.3. Analysis of the 4D Anderson's Iris Dataset. We show now the potentiality of the proposed method for visualizing 4D data, analysing the well-known Anderson's Iris dataset [22]. This dataset is composed of 150 patterns of four real variables, and each pattern belongs to one of three different classes. A first map obtained from a classic SOM of grid dimensions 40×10 and a second map obtained by using a second order Butterworth LF are considered. Note that the number of neurons is higher than the number of samples in the data, then the SOM is used as a nonlinear regression-interpolation method. We consider here results that show similar quantization error and the distortion measure of the two maps. Therefore, equivalent information is achievable from the final distribution of the model vectors as shown in Figure 10. However, it is important to evidence the situation of a very short training length duration, where the number of presentations is limited to one or two epochs. In this case, the U-matrix does not give significant information, as can be seen from Figure 11(a), but from the E-matrix and trajectory vectors a rough, but correct, indication of the presence of



(a) (b) (c)

Figure 10: (a) U-matrix of the map trained with the classic SOM. (b) E-matrix of the map trained with a second-order Butterworth LF. (c) Calibration of the map in (b).



(a) (b)

Figure 11: (a) U-matrix, and (b) E-matrix of a map trained with a short training duration.

two zones of attraction is easily obtained, as shown in Figure 11(b). Then, from this "rough" information it is possible to change some map parameters (number of neurons, filter order, etc.) and immediately optimize the learning process.

5. Computational Aspects

Considering the numerical complexity of the proposed model, it is important to note that the more demanding operations are the comparisons needed to find the winner unit. In the proposed model, the search of the winner is accomplished as in classical SOM, so that if no shortcut winner search is used, the numerical complexity is $o(D^2)$ where D is the number of map units. The proposed model increases the number of updating operations per unit: when the classic SOM needs 1 scalar-vector multiplications at each update step, the proposed method needs $4N + 1$ scalar-vector multiplications, so that the increase of the computational load is linear with the filter order N. With regard to the memory requirements, the proposed model needs $2N$ times the memory needed by the classic SOM. The increased requirements of computational resources are acceptable when the order of the filter is kept below $N = 10$, whereas the benefits of the proposed model are noticeable even for $N = 2$ or $N = 3$.

6. Conclusions

The presented work shows the feasibility of implementing the SOM learning with a higher-order difference equation that takes into account the data of the previous steps of the update process. The incremental update rule proposed is based on a discrete difference implementation of a suitably designed low-pass digital filter. The proposed model is a framework that allows the flexibility to design different learning strategies for the incremental training of various algorithms related to vector quantization, including the topology preserving methods such as SOMs. It is shown that this approach allows a simple and flexible control of the SOM formation. This gives better performances indexes with respect to a classical SOM at a cost of a slightly higher computational cost. Moreover, the memory of previous data used in the filter implementation can be used at the end of the training for visualizing the training trajectories. This novel visualization tools are very useful to understand the dynamic of the map formation and can be used for visual cluster analysis.

Acknowledgment

This work was supported in part by the Italian Ministry of University (MIUR) under a Program for the Development of Research of National Interest (PRIN Grant no. 20072347AY).

References

[1] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1–3, pp. 1–6, 1998.

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

[3] T. Kohonen, *Self-Organizing Maps*, Springer, Berlin, Germany, 3rd edition, 2001.

[4] S. Kaski, T. Honkela, K. Lagus, and T. Kohonen, "WEBSOM—self-organizing maps of document collections," *Neurocomputing*, vol. 21, no. 1–3, pp. 101–117, 1998.

[5] T. Kohonen and P. Somervuo, "Self-organizing maps of symbol strings," *Neurocomputing*, vol. 21, no. 1–3, pp. 19–30, 1998.

[6] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas, "Engineering applications of the self-organizing map," *Proceedings of the IEEE*, vol. 84, no. 10, pp. 1358–1383, 1996.

[7] E. Erwin, K. Obermayer, and K. Schulten, "Self-organizing maps: ordering, convergence properties and energy functions," *Biological Cybernetics*, vol. 67, no. 1, pp. 47–55, 1992.

[8] E. Meŕenyi, A. Jain, and T. Villmann, "Explicit magnification control of self-organizing maps for "Forbidden" data," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 786–797, 2007.

[9] Y. Zheng and J. F. Greenleaf, "The effect of concave and convex weight adjustments on self-organizing maps," *IEEE Transactions on Neural Networks*, vol. 7, no. 1, pp. 87–96, 1996.

[10] J. Cho, A. R. C. Paiva, S. P. Kim, J. C. Sanchez, and J. C. Príncipe, "Self-organizing maps with dynamic learning for signal reconstruction," *Neural Networks*, vol. 20, no. 2, pp. 274–284, 2007.

[11] Y. M. Cheung and L. T. Law, "Rival-model penalized self-organizing map," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 289–295, 2007.

[12] S. Cho and J. Seok, "Self-organizing map with time-invariant learning rate and its exponential stability analysis," *Neurocomputing*, vol. 19, no. 1–3, pp. 1–11, 1998.

[13] D. Theofilou, V. Steuber, and E. De Schutter, "Novelty detection in a Kohonen-like network with a long-term depression learning rule," *Neurocomputing*, vol. 52-54, pp. 411–417, 2003.

[14] E. Berglund and J. Sitte, "The parameterless self-organizing map algorithm," *IEEE Transactions on Neural Networks*, vol. 17, no. 2, pp. 305–316, 2006.

[15] H. Shah-Hosseini and R. Safabakhsh, "TASOM: a new time adaptive self-organizing map," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 33, no. 2, pp. 271–282, 2003.

[16] K. Haese, "Self-organizing feature maps with self-adjusting learning parameters," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1270–1278, 1998.

[17] A. Antoniou, *Digital Filters: Analysis, Design, and Applications*, McGraw-Hill, New York, NY, USA, 1993.

[18] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "'Neural-gas' network for vector quantization and its application to time-series prediction," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 558–569, 1993.

[19] A. Ultsch, "U*-matrix: a tool to visualize clusters in high dimensional data," Tech. Rep., University of Marburg, 2000.

[20] D. Lepetz, M. Némoz-Gaillard, and M. Aupetit, "Concerning the differentiability of the energy function in vector quantization algorithms," *Neural Networks*, vol. 20, no. 5, pp. 621–630, 2007.

[21] J. Rynkiewicz, "Self-organizing map algorithm and distortion measure," *Neural Networks*, vol. 19, no. 6-7, pp. 830–837, 2006.

[22] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, pp. 179–188, 1936.