HAL
open science

Dynamic Self-Organising Map

Nicolas P. Rougier, Yann Boniface

To cite this version:

Nicolas P. Rougier, Yann Boniface. Dynamic Self-Organising Map. Neurocomputing, 2011, 74 (11), pp.1840-1847. 10.1016/j.neucom.2010.06.034 . inria-00495827

HAL Id: inria-00495827
https://inria.hal.science/inria-00495827v1

Submitted on 29 Jun 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Dynamic Self-Organising Map

Nicolas Rougier*[1] and Yann Boniface[2]

[1]LORIA/INRIA Nancy - Grand Est Research Centre, 54600 Villers-lès-Nancy, France
[2]LORIA/Université Nancy 2, 54015 Nancy Cedex, France

May 26, 2010

Abstract

We present in this paper a variation of the self-organising map algorithm where the original time-dependent (learning rate and neighbourhood) learning function is replaced by a time-invariant one. This allows for on-line and continuous learning on both static and dynamic data distributions. One of the property of the newly proposed algorithm is that it does not fit the magnification law and the achieved vector density is not directly proportional to the density of the distribution as found in most vector quantisation algorithms. From a biological point of view, this algorithm sheds light on cortical plasticity seen as a dynamic and tight coupling between the environment and the model.
Keywords: self organisation, on-line, cortical plasticity, dynamic

1 Introduction

Vector quantisation (VQ) refers to the modelling of a probability density function into a discrete set of prototype vectors (sometimes called the codebook) such that any point drawn from the associated distribution can be associated to a prototype vector. Most VQ algorithms try to match the density through the density of their codebook: high density regions of the distribution tend to have more associated prototypes than low density region. This generally allows to minimise the loss of information (or distortion) as measured by the mean quadratic error. For a complete picture, it is to be noted that there also exists some cases where only a partition of the space occupied by the data (regardless of their density) is necessary. In this case, one wants to achieve a regular quantification *a priori* of the probability density function. For example, in some classification problems, one wants to achieve a discrimination of data in term of classes and thus needs only to draw frontiers between data regardless of their respective density.

Vector quantisation can be achieved using several methods such as variations of the k-means method [1], Linde–Buzo–Gray (LBG) algorithm [2] or neural network models such as the self-organising map (SOM) [3], neural gas (NG) [4] and growing neural gas (GNG) [5]. Among all these methods, the SOM algorithm is certainly the most famous in the field of computational neurosciences since it can give a biologically and plausible account on the organisation of receptive fields in sensory areas where adjacent neurons shares similar representations. The stability and the quality of such self-organisation depends heavily on a decreasing learning rate as well as a decreasing neighbourhood function. This is quite congruent with the idea of a critical period in the early years of development where most sensory or motor properties are acquired and

*Corresponding author: Nicolas.Rougier@loria.fr

stabilised [6–8]. However, this fails to explain cortical plasticity since we know that the cortex has the capacity to re-organise itself in face of lesions or deficits [9–11]. The question is then to know to what extent it is possible to have both stable and dynamic representations ?

Quite obviously, this cannot be achieved using SOM-like algorithms that depends on a time decreasing learning rate and/or neighbourhood function (SOM, NG, GNG) and, despite the huge amount of literature [12, 13] around self-organising maps and Kohonen-typed networks (more than 7000 works listed in [14]), there is is surprisingly and comparatively very little work dealing with online learning (also referred as incremental or lifelong learning). Furthermore, most of these works are based on incremental models, that is, networks that create and/or delete nodes as necessary. For example, the modified GNG model [15] is able to follow non-stationary distributions by creating nodes like in a regular GNG and deleting them when they have a too small *utility* parameter. Similarly, the evolving self-organising map (ESOM) [16, 17] is based on an incremental network quite similar to GNG that creates dynamically based on the measure of the distance of the winner to the data (but the new node is created at exact data point instead of the mid-point as in GNG). Self-organising incremental neural network (SOINN) [18] and its enhanced version (ESOINN) [19] are also based on an incremental structure where the first version is using a two layers network while the enhanced version proposed a single layer network. One noticeable result is the model proposed by [20] which does not rely on a incremental structure but is based on the Butterworth decay scheme that does not decay parameters to zero. The model works in two phases, an initial phase (approximately ten epochs) is used to establish a rough global topology thanks to a very large neighbourhood and the second phase uses a small neighbourhood phase to train the network. Unfortunately, the size of the neighbourhood in the second phase has to be adapted to the expected density of the data.

Without judging performances of these models, we do not think they give a satisfactory answer to our initial question and we propose instead to answer by considering a tight coupling between the environment and representations. If the environment is stable, representations should remain stable and if the environment suddenly changes, representations must dynamically adapt themselves and stabilise again onto the new environment. We thus modified the original SOM algorithm in order to make its learning rule and neighbourhood independent of time. This results in a tight coupling between the environment and the model that ensure both stability and plasticity. In next section, we formally describe the dynamic self-organising map in the context of vector quantisation and both neural gas and self-organising map are formally described in order to underline differences between the three algorithms. The next section re-introduces the model from a more behavioural point of view and main experimental results are introduced using either low or high dimensional data and offers side-to-side comparison with other algorithms. Results concerning dynamic distributions are also introduced in the case of dynamic self-organising map in order to illustrate the coupling between the distribution and the model. Finally, we discuss the relevancy of such a model in the context of computational neurosciences and embodied cognition.

2 Definitions

Let us consider a probability density function $f(x)$ on a compact manifold $\Omega \in \mathbb{R}^d$. A vector quantisation (VQ) is a function Φ from Ω to a finite subset of n code words $\{\mathbf{w}_i \in \mathbb{R}^d\}_{1 \leq i \leq n}$ that form the codebook. A cluster is defined as $C_i \stackrel{\text{def}}{=} \{x \in \Omega | \Phi(x) = \mathbf{w}_i\}$, which forms a partition

of Ω and the distortion of the VQ is measured by the mean quadratic error

$$\xi = \sum_{i=1}^{n} \int_{C_i} \|x - \mathbf{w}_i\|^2 f(x) dx. \tag{2.1}$$

If the function f is unknown and a finite set $\{x_i\}$ of p non biased observations is available, the distortion error may be empirically estimated by

$$\hat{\xi} = \frac{1}{p} \sum_{i=1}^{n} \sum_{x_j \in C_i} \|x_j - \mathbf{w}_i\|^2. \tag{2.2}$$

Neural maps define a special type of vector quantifiers whose most common approaches are the Self-Organising Map (SOM) [3], Elastic Net (EN) [21], Neural Gas (NG) [4] and Growing Neural Gas (GNG) [22]. In the following, we will use definitions and notations introduced by [23] where a neural map is defined as the projection from a manifold $\Omega \subset \mathbb{R}^d$ onto a set $\mathcal{N}$ of n *neurons* which is formally written as $\Phi : \Omega \to \mathcal{N}$. Each neuron i is associated with a code word $\mathbf{w}_i \in \mathbb{R}^d$, all of which established the set $\{\mathbf{w}_i\}_{i \in \mathcal{N}}$ that is referred as the codebook. The mapping from Ω to $\mathcal{N}$ is a closest-neighbour winner-take-all rule such that any vector $\mathbf{v} \in \Omega$ is mapped to a neuron i with the code $\mathbf{w}_\mathbf{v}$ being closest to the actual presented stimulus vector $\mathbf{v}$,

$$\Phi : \mathbf{v} \mapsto \arg\min_{i \in \mathcal{N}} (\|\mathbf{v} - \mathbf{w}_i\|). \tag{2.3}$$

The neuron $\mathbf{w}_\mathbf{v}$ is called the *winning element* and the set $C_i = \{x \in \Omega | \Phi(x) = \mathbf{w}_i\}$ is called the *receptive field* of the neuron i. The geometry corresponds to a Voronoï diagram of the space with $\mathbf{w}_i$ as the center.

2.1 Self-Organising Maps (SOM)

SOM is a neural map equipped with a structure (usually a hypercube or hexagonal lattice) and each element i is assigned a fixed position $\mathbf{p}_i$ in $\mathbb{R}^q$ where q is the dimension of the lattice (usually 1 or 2). The learning process is an iterative process between time $t = 0$ and time $t = t_f \in \mathbb{N}^+$ where vectors $\mathbf{v} \in \Omega$ are sequentially presented to the map with respect to the probability density function f. For each presented vector $\mathbf{v}$ at time t, a winner $s \in \mathcal{N}$ is determined according to equation (2.3). All codes $\mathbf{w}_i$ from the codebook are shifted towards $\mathbf{v}$ according to

$$\Delta \mathbf{w}_i = \varepsilon(t)\ h_\sigma(t, i, s)\ (\mathbf{v} - \mathbf{w}_i) \tag{2.4}$$

with $h_\sigma(t, i, j)$ being a neighbourhood function of the form

$$h_\sigma(t, i, j) = e^{-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|^2}{2\sigma(t)^2}}. \tag{2.5}$$

where $\varepsilon(t) \in \mathbb{R}$ is the learning rate and $\sigma(t) \in \mathbb{R}$ is the width of the neighbourhood defined as

$$\sigma(t) = \sigma_i \left(\frac{\sigma_f}{\sigma_i}\right)^{t/t_f}, \text{ with } \varepsilon(t) = \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i}\right)^{t/t_f}, \tag{2.6}$$

while σ_i and σ_f are respectively the initial and final neighbourhood width and ε_i and ε_f are respectively the initial and final learning rate. We usually have $\sigma_f \ll \sigma_i$ and $\varepsilon_f \ll \varepsilon_i$.

2.2 Neural Gas (NG)

In the case of NG, the learning process is an iterative process between time $t = 0$ and time $t = t_f \in \mathbb{N}^+$ where vectors $\mathbf{v} \in \Omega$ are sequentially presented to the map with respect to the probability density function f. For each presented vector $\mathbf{v}$ at time t, neurons are ordered according to their respective distance to $\mathbf{v}$ (closest distances map to lower ranks) and assigned a rank $k_i(\mathbf{v})$. All codes $\mathbf{w}_i$ from the codebook are shifted towards $\mathbf{v}$ according to

$$\Delta \mathbf{w}_i = \varepsilon(t)\ h_\lambda(t, i, \mathbf{v})\ (\mathbf{v} - \mathbf{w}_i) \tag{2.7}$$

with $h_\lambda(t, i, \mathbf{v})$ being a neighbourhood function of the form:

$$h_\lambda(t, i, \mathbf{v}) = e^{-\frac{k_i(\mathbf{v})}{\lambda(t)}} \tag{2.8}$$

where $\varepsilon(t) \in \mathbb{R}$ is the learning rate and $\lambda(t) \in \mathbb{R}$ is the width of the neighbourhood defined as

$$\lambda(t) = \lambda_i \left(\frac{\lambda_f}{\lambda_i}\right)^{t/t_f}, \text{ with } \varepsilon(t) = \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i}\right)^{t/t_f}, \tag{2.9}$$

while λ_i and λ_f are respectively the initial and final neighbourhood and ε_i and ε_f are respectively the initial and final learning rate. We usually have $\lambda_f \ll \lambda_i$ and $\varepsilon_f \ll \varepsilon_i$.

2.3 Dynamic Self-Organising Map (DSOM)

DSOM is a neural map equipped with a structure (a hypercube or hexagonal lattice) and each neuron i is assigned a fixed position $\mathbf{p}_i$ in $\mathbb{R}^q$ where q is the dimension of the lattice (usually 1 or 2). The learning process is an iterative process where vectors $\mathbf{v} \in \Omega$ are sequentially presented to the map with respect to the probability density function f. For each presented vector $\mathbf{v}$, a winner $s \in \mathcal{N}$ is determined according to equation (2.3). All codes $\mathbf{w}_i$ from the codebook $\mathbf{W}$ are shifted towards $\mathbf{v}$ according to

$$\Delta \mathbf{w}_i = \varepsilon \|\mathbf{v} - \mathbf{w}_i\|_\Omega\ h_\eta(i, s, \mathbf{v})\ (\mathbf{v} - \mathbf{w}_i) \tag{2.10}$$

with ε being a constant learning rate and $h_\eta(i, s, \mathbf{v})$ being a neighbourhood function of the form

$$h_\eta(i, s, \mathbf{v}) = e^{-\frac{1}{\eta^2}\frac{\|\mathbf{p}_i - \mathbf{p}_s\|^2}{\|\mathbf{v} - \mathbf{w}_s\|_\Omega^2}} \tag{2.11}$$

where η is the *elasticity* or *plasticity* parameter. If $\mathbf{v} = \mathbf{w}_s$, then $h_\eta(i, s, \mathbf{v}) = 0$

3 Model

As we explained in the introduction, the DSOM algorithm is essentially a variation of the SOM algorithm where the time dependency has been removed. Regular learning function (2.4) and neighbourhood function (2.5) have been respectively replaced by equations (2.10) and (2.11) which reflect two main ideas:

- If a neuron is close enough to the data, there is no need for others to learn anything: the winner can represent the data.
- If there is no neuron close enough to the data, any neuron learns the data according to its own distance to the data.

This draws several consequences on the notion of neighbourhood that is now dynamic and leads to a qualitatively different self-organisation that can be controlled using a free elasticity parameter.

3.1 Dynamic neighbourhood

Learning rate is modulated using the closeness of the winner to the data. The figure 1 represents this learning rate modulation as a function of a data $\mathbf{v}$, a neuron i (with code $\mathbf{w}_i$) and a winner s (with code $\mathbf{w}_s$). If the winner s is very close or equal to $\mathbf{v}$ (bottom line on the figure), learning rate of any neuron different from the winner s is zero and only the winner actually learns the new data. When the winner s is very far from the data (top line), any neuron benefits from a large learning rate and learns the new data (modulated by their own distance to the data but this extra modulation is not represented on the figure). This notion of closeness of the winner to the data is thus critical for the algorithm and modifies considerably both the notion of neighbourhood and the final codebook. Most VQ tries to capture data density through the density of their codebook as introduced in [23] where authors considers the generalised error

$$E_\gamma = \int_\Omega \|\mathbf{w}_s - \mathbf{v}\|^\gamma P(\mathbf{v}) d\mathbf{v} \tag{3.1}$$

and introduces the relation $P(\mathbf{w}) \propto \rho(\mathbf{w})^\alpha$ with $\rho(\mathbf{w})$ being the weight vector density and α being the *magnification exponent* or *magnification factor*. If we consider the intrinsic (or Hausdorff) dimension d of the data, the relation between magnification and d is given by $\alpha = \frac{d}{d+\gamma}$ and an ideal VQ achieves a magnification factor of 1. However, DSOM algorithm clearly states that if a neuron is already close enough to a presented data, there is no need for the neighbours to learn anything and this results in a codebook that does not follow the magnification law as illustrated on figure 2 for three very simple two-dimensional non homogeneous distributions. Said differently, what is actually mapped by the DSOM is the structure or support of the distribution (Ω using notations introduced in section 2) rather than the density.

3.2 Elasticity

The DSOM algorithm is not parameter free since we need to control when a neuron may be considered to be close enough to a data such that it prevents learning for its neighbours. This is the role of the elasticity parameter that modulates the strength of the coupling between neurons as shown on figure 3 for a simple two-dimensional normal distribution. This notion of elasticity shares some common concepts with the Adaptive Resonance Theory (ART) as it has been introduced in [24]. In the ART model, the vigilance parameter has a critical influence on learning since it controls the actual partition of the input space: high vigilance level produces high number of very precise memories while low vigilance level produces fewer and more generic memories. This is very similar to the elasticity parameter: if elasticity is high, neurons tend to pack themselves very tightly together (code vectors are relatively close) while a lower elasticity allows for looser coupling between neurons. However, in the case of ART, the vigilance parameter also governs the number of final prototypes since they can be created on demand. In the case of DSOM, the number of prototypes (i.e. neurons) is fixed and they are supposed to span the whole input space to ensure convergence. Consequently, there exists a relation between the diameter of the support (defined as the maximum distance between any two points in Ω), the number of neurons and the elasticity parameter. In the one hand, if elasticity is too high, neurons cannot span the whole space and the DSOM algorithm does not converge, in the other hand, if elasticity is too low, coupling between neurons is weak and may prevent self-organisation to occur: code-vectors are evenly spread on the support but they do not respect the neighbourhood relationship anymore. There certainly exists an optimal elasticity for a given distribution but we did not yet investigate fully this relationship and we do not have formal results. As a preliminary work, we have studied the relationship between elasticity and the initial conditions in the one dimensional case using a very simple experimental

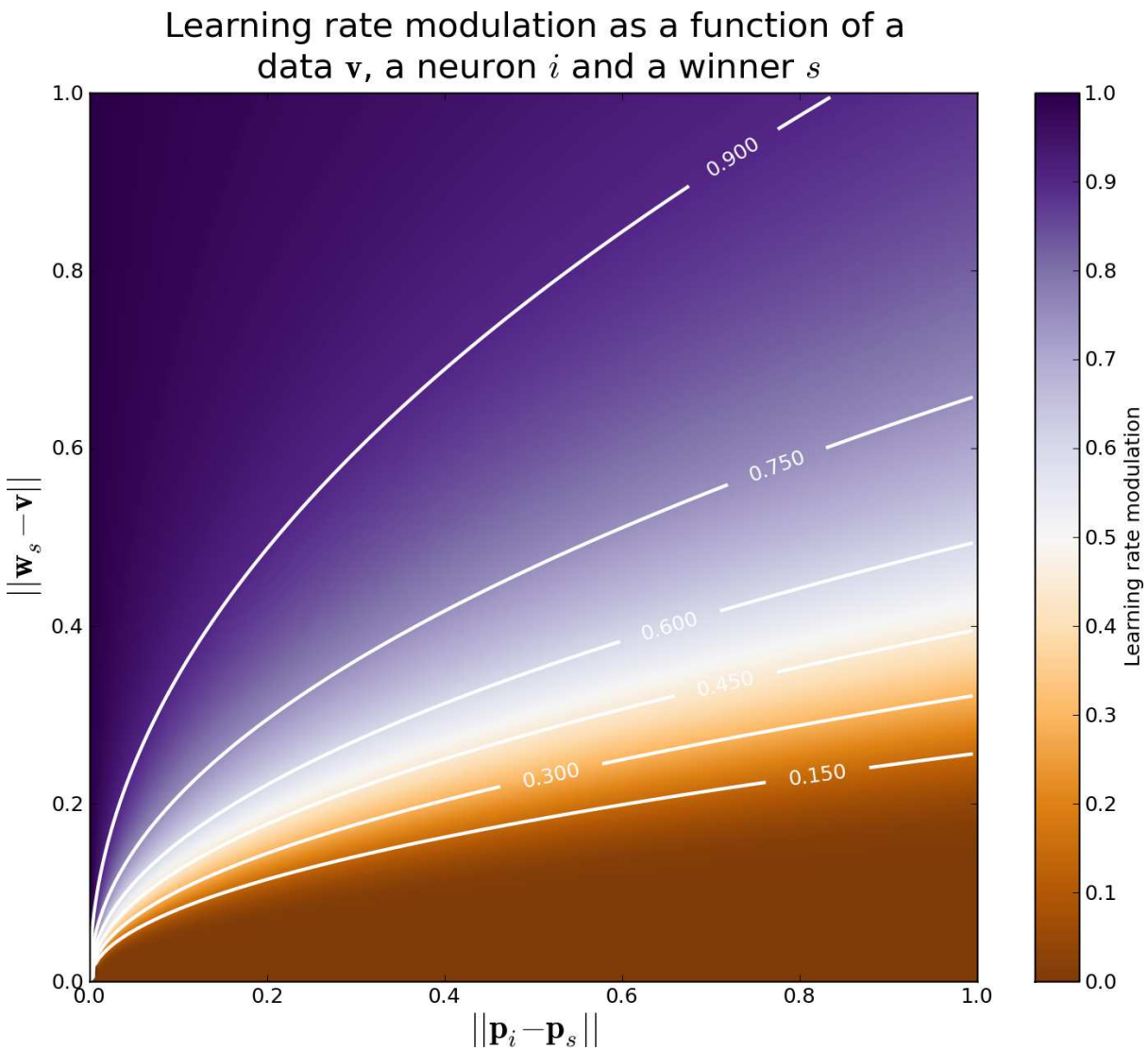

Figure 1: At each presented data $\mathbf{v}$, the learning rate of each neuron i is modulated according to both the distance $\|\mathbf{w}_s - \mathbf{v}\|$ (which represents the distance between the winner s and the presented data $\mathbf{v}$) and the distance $\|\mathbf{p}_i - \mathbf{p}_s\|$ (which represent the distance between code words of neuron i and neuron s). If the winner s is very close or equal to $\mathbf{v}$ (bottom line on the figure), learning rate of any neuron different from the winner s is zero and only the winner actually learns the new data. When the winner s is very far from the data (top line), any neuron benefits from a large learning rate and learns the new data (modulated by their own distance to the data but this extra modulation is not represented on the figure).

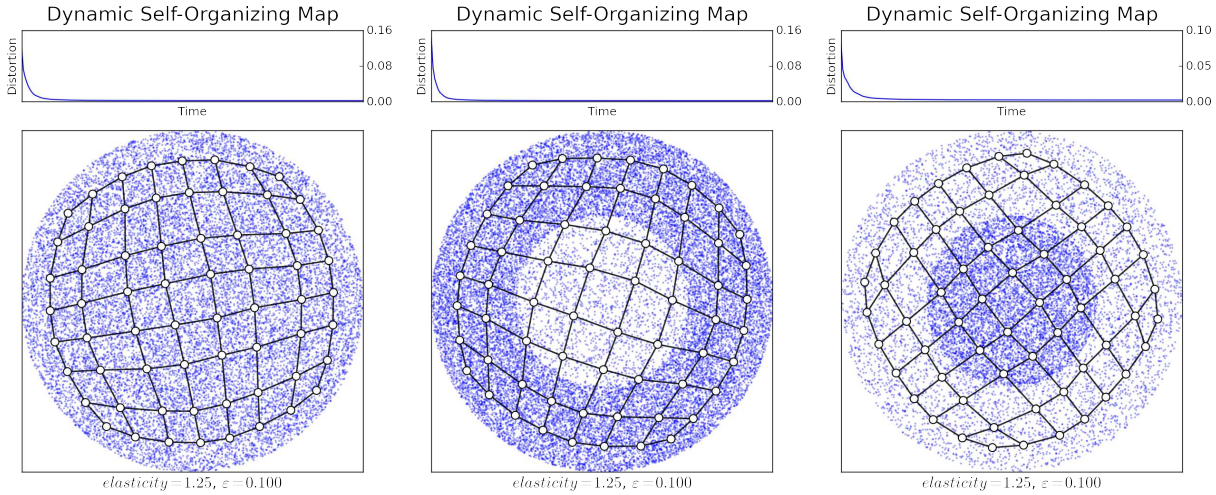

Figure 2: Three DSOM have been trained on a disc distribution using different density areas. **Left.** The density is uniform all over the disc (0.25). **Center**. Outer ring has higher density (.4) than inner disc (.1). **Right**. Outer ring has lower density (.1) than inner disc (.4). Despite these different density distributions, the three DSOM self-organise onto the support of the distribution (the whole disc) and does not try to match density.

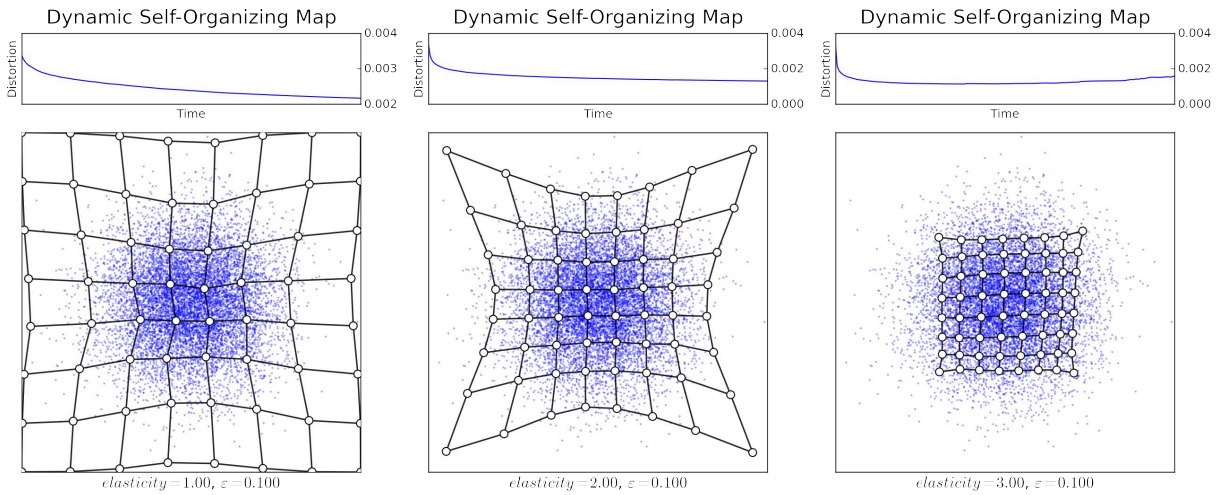

Figure 3: Three DSOM with respective elasticity equal to 1, 1.5 and 2 have been trained for 20 000 iteration on a normal distribution using a regular grid covering the $[0,1]^2$ segment as initialisation. Low elasticity leads to loose coupling between neurons while higher elasticity results in a tight coupling between neurons.

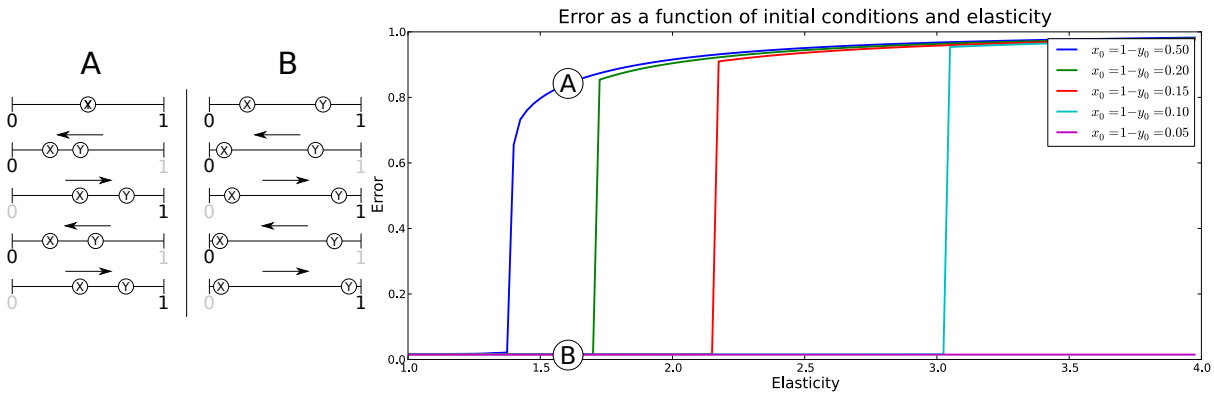

Figure 4: Several one-dimensional DSOM with two nodes have been trained for 2500 epochs using a dataset of two samples (0 and 1) that were presented alternatively. Each point of each curve represents the error of a network with given elasticity and initial conditions. Point A represents a case where elasticity is too high and makes the network to oscillate while point B represents a case where elasticity was low enough to allow the network to properly converge (towards $x = 0$ and $y = 1$).

setup where the dataset is made of only two samples (one at 0 and the other at 1) as explained on figure 4. This figure clearly shows a discontinuity in the error when elasticity is varying from 1.0 to 4.0 but at different places for different initial conditions. The reason comes from the dependency of the learning to the distance between the winner node and the presented data. When this difference is large, a large correction of weights occur on all networks nodes and this is only attenuated by their distance to the winner and the network elasticity. In the presented experimental setup, data (0 and 1) were presented alternatively and lead to a convergence when elasticity was low enough and to an oscillatory behaviour (not visible on the figure) when elasticity was too high. This oscillatory behaviour can be understood most simply when looking at scheme A on the figure. Each correction made to the network in one way is immediately counter-balanced in the other way when next data is presented. This preliminary study lead us to think that the choice of an optimal elasticity not only depends on the size of the network and the size of the support but also on the initial conditions. If we were to generalise from the simple study above, the initial configuration of the network should cover the entire support as much as possible to reduce elasticity dependency.

3.3 Convergence

It is well known that the convergence of the Kohonen algorithm has not be proved in the general case [25] even though some conditional convergence properties have been established in the one-dimensional case [26]. Furthermore, in the case of continuous input, it has been shown that there does not exist an associated energy function [27] and in the case of a finite set of training patterns, the energy function is highly discontinuous [28]. In the case of the dynamic SOM, the proof of convergence is straightforward since we can exhibit at least one case where the DSOM does not converge, when the number of nodes is less then the number of data as illustrated on figure 5. Most generally, in case where the number of nodes is less than the total number of presented data, we can predict that the dynamic SOM will not converge. Moreover, a similar problem occurs if the number of nodes is exactly equal to the number of data and if nodes are initially distributed uniquely on each data. In such an initial setup, the learning parameter is

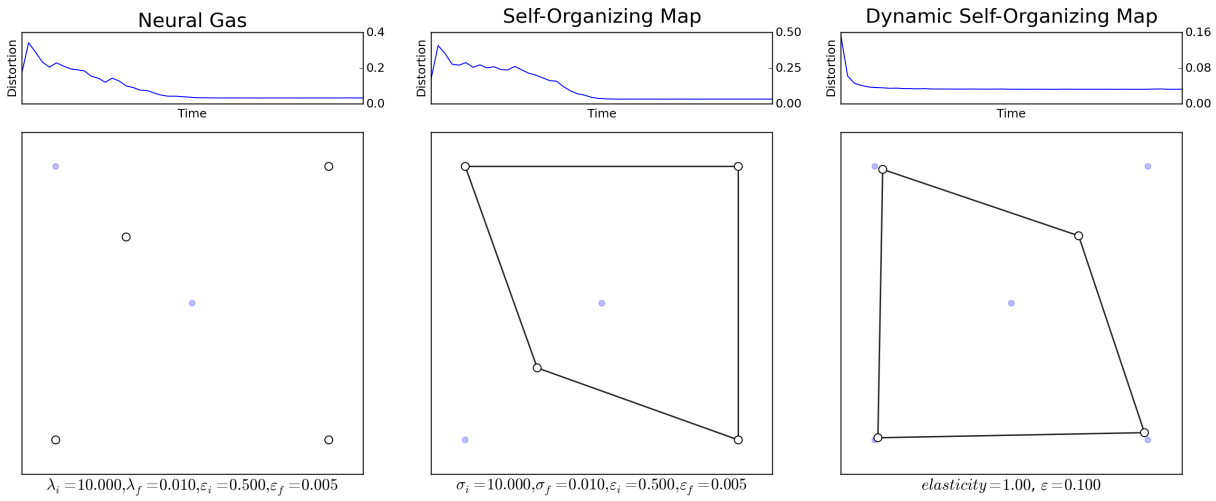

Figure 5: Due to its dynamic nature, the dynamic SOM cannot converge when the number of nodes (4 here) is less than the number of data (5 here). NG and SOM can converge on an approximated solution thanks to both their decaying learning rate and neighborhood and this explains why three nodes are exactly aligned with their corresponding data while the last node found a mid-distance position. In the case of DSOM and because of the constant learning rate, every node is moving at each presented data and thus cannot converge at all.

zero for any presented data and this prevents the network to learn anything at all. We could say that it does converge in such a case (network is frozen) but if the initial configuration does not correspond to a proper unfolded one, the answer would not be really satisfactory. A proof of convergence would then require to identify configurations (initial conditions, size, elasticity, learning rate) where the network may have chances to converge but we think this is currently out of the scope of this paper.

4 Experimental results

We report in this section some experimental results we obtained on different types of distribution that aim at illustrating DSOM principles. We do not have yet formal results about convergence and/or quality of the codebook. As a consequence, these results do not pretend to prove anything and are introduced mainly to illustrate qualitative behaviour of the algorithm.
Unless stated otherwise, the learning procedure in following examples is:

1. A distribution is chosen (normal, uniform, etc.)
2. A discrete sample set of samples is drawn from the distribution
3. Model learns for n iterations
4. At each iteration, a sample is picked randomly and uniformly in the discrete sample set
5. Distortion is measured on whole sample set every 100 iterations using equation (2.2).

The distortion error is plotted above each graphics to show rate of convergence.

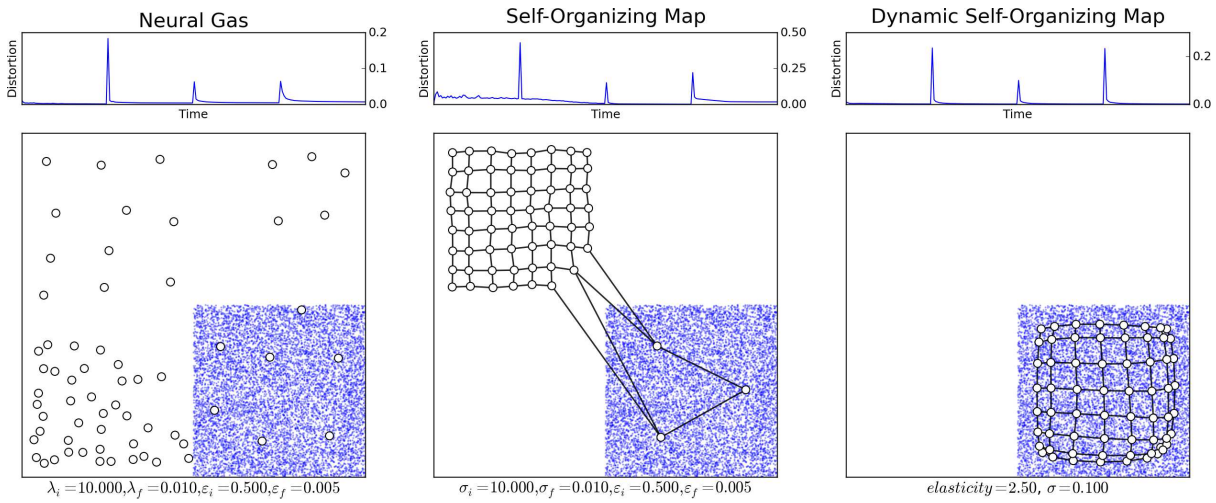

Figure 6: Three networks (NG, SOM, DSOM) have been trained for 20 000 iterations on a dynamic distribution that vary along time: a uniform distribution (1) on $[0.0, 0.5] \times [0.0, 0.5]$ from iterations 0 to 5000, a uniform distribution (2) on $[0.5, 1.0] \times [0.5, 1.0]$ from iterations 5000 to 10000, a uniform distribution (3) on $[0.0, 0.5] \times [0.5, 1.0]$ from iterations 10000 to 15000 and a final uniform distribution (4) on $[0.5, 1.0] \times [0.0, 0.5]$ from iterations 15000 to 20000.

4.1 Non-stationary distributions

In order to study dynamic aspect of the DSOM algorithm, three networks (NG, SOM, DSOM) have been trained for 20 000 iterations on a dynamic distribution that vary along time: a uniform distribution (1) on $[0.0, 0.5] \times [0.0, 0.5]$ from iterations 0 to 5000, a uniform distribution (2) on $[0.5, 1.0] \times [0.5, 1.0]$ from iterations 5000 to 10000, a uniform distribution (3) on $[0.0, 0.5] \times [0.5, 1.0]$ from iterations 10000 to 15000 and a final uniform distribution (4) on $[0.5, 1.0] \times [0.0, 0.5]$ from iterations 15000 to 20000. NG shows some difficulties in tracking various changes and the final state reflects the history of the distribution: there are many code words within the first distribution and very few in the final one. In the case of SOM, the algorithm can almost cope with the dynamic nature of the distributions as long as its learning rate and neighbourhood function are large enough to move the codebook into the new data region. This is the case for distributions (1) to (3) but the final change makes the SOM network unable to map the final distribution as expected because of the time dependency of the algorithm. In the case of DSOM, the network is able to accurately track each successive distribution with a short transient error correlated to the distribution change. We think this behaviour reflects cortical plasticity seen as a tight coupling between the model and the environment.

4.2 High-dimensional distributions

Until now, we have considered only trivial two-dimensional distributions whose intrinsic dimension matched the topography of the network. We now consider higher dimensional distribution with unknown intrinsic dimension. Using the standard Lena grey-level image as a source input, samples of 8×8 pixels have been draw uniformly from the image and presented to the different networks. 1000 such samples have been drawn and all three networks have learnt during 10 000 iterations. As illustrated on figure 7, the strong influence of neighbourhood in the case of SOM leads to a final codebook where vectors tend to be very homogeneous and composed of a mean value with little variations around this mean value. In the case of NG, things are

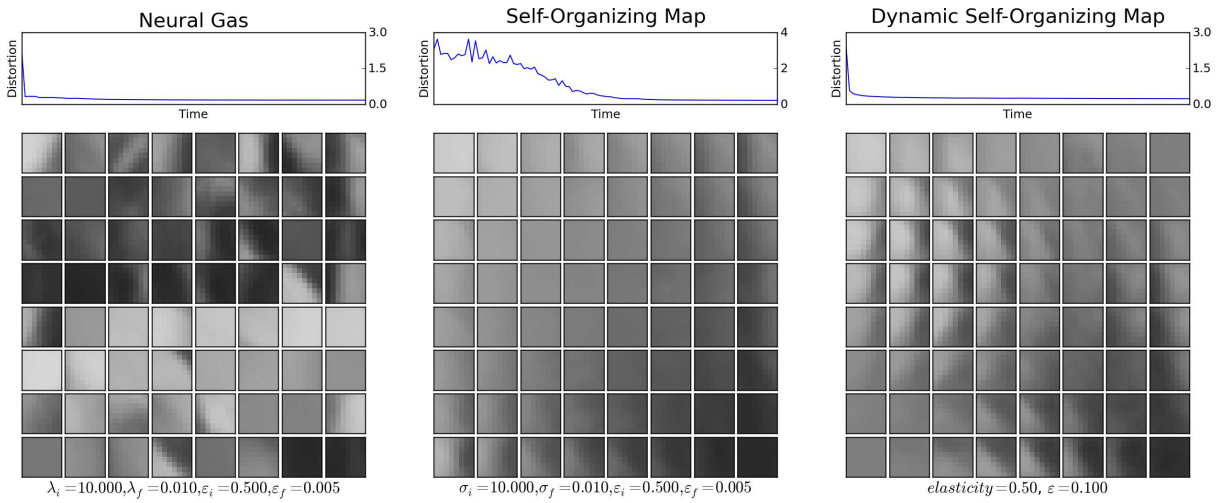

Figure 7: Three networks (NG, SOM, DSOM) have been trained for 20 000 iterations on 1000 samples of size 8×8 pixels that have been drawn uniformly from the standard lena grey image.

different because of the absence of topographic constraints: NG converges rapidly toward a stable solution made of qualitatively different filters, part of them are quite homogeneous like in SOM but some others clearly possess a greater internal variety. In the case of DSOM, we can also check on the figure a greater variety of filters that are self-organised. The meaning of such a greater variety of filters in the case of DSOM is difficult to appreciate. In the one hand, if we were to reconstruct the original image using those filters, we would certainly obtain a larger distortion error. In the other hand, if those filters were supposed to extract useful information from the image, they would certainly give a better account of the structure of the image.

5 Conclusion

One of the major problem of most neural map algorithms is the necessity to have a finite set of observations to perform adaptive learning starting from a set of initial parameters (learning rate, neighbourhood or temperature) at time t_i down to a set of final parameters at time t_f. In the framework of signal processing or data analysis, this may be acceptable as long as we can generate a finite set of samples in order to learn it off-line. However, from a more behavioural point of view, this is not always possible to have access to a finite set and we must face on-line learning. As explained in the introduction, if we consider the existence of a critical period in the early years of development, the problem may be solved using decreasing learning rate and neighbourhood over an extended period of time. But if this may explain to some extents the development of early sensory filters, this fails at explaining cortical plasticity at a more broad level. As explained in [29], we know today that "cortical representations are not fixed entities, but rather, are dynamic and are continuously modified by experience". How can we achieve both stability and reactivity ?

We proposed to answer this question by introducing a variant of the original SOM learning algorithm where time depency has been removed. With no available formal proof of convergence and based on several experiments in both two-dimensional, high-dimensional cases and dynamic cases, we think this new algorithm allows for on-line and continuous learning ensuring a tight coupling to the environment. However, the resulting codebook does not fit data den-

sity as expected in most VQ algorithms. This could be a serious drawback in the framework of signal processing or data compression but may be a desirable property from a behavioural point fo view. For example let us consider a picture of a (very) snowy landscape with a small tree in the middle. If we want to mimic visual exploration of the scene using eye saccades, we can randomly pick small patches within the image and present them to the model. Not very surprisingly, the vast majority of these patches would be essentially white (possibly with some variations) because the whole image is mainly white. From a pure VQ point of view, the codebook would reflect this density by having a vast majority of its representations into the white domain and if the tree is small enough, we could even have only white representation within the codebook. While this would serve data compression, how much is it relevant in general ? We do not have the answer in the general case but we think this must be decided explicitely depending on task. DSOM allows such explicit decision since it maps the structure of the data rather than their density. This means that in a more general framework, we could expect an external structure to attach some kind of motivation for each data that would modulate its learning. If some region of the perceptive space is judged behaviourally relevant, model could develop precise representations in this region but if learning is driven solely by data density (like in most VQ), such modulation would certainly be strongly attenuated or not possible at all.

Acknowledgement. This work has received useful corrections and comments by Thierry Viéville and support from the MAPS ANR grant.

A Notations

Ω : a compact manifold of $\mathbb{R}^d$ where $d \in \mathbb{N}^+$

$f(x)$: a probability density function (*pdf*) $\Omega \to \mathbb{R}$

$\{x_i\}$: a set of p non-biased observations of f.

$\mathcal{N}$: a set of n elements, $n \in \mathbb{N}^+$.

Φ : a function defined from $\Omega \to \mathcal{N}$

$\mathbf{w}_i \in \mathbb{R}^d$: code word associated to an element i of $\mathcal{N}$

$\{\mathbf{w}_i\}$: codebook associated to $\mathcal{N}$

C_i : cluster associated to element i such that $C_i = \{x \in \Omega | \Phi(x) = \mathbf{w}_i\}$

$\|x\|$: euclidean norm defined over $\mathbb{R}^d$

$\|x\|_\Omega$: normalised euclidean norm defined over Ω as $x \mapsto \frac{\|x\|}{\max_{y,z\in\Omega}(\|y-z\|)}$

ξ : distortion error defined as $\sum_{i=1}^n \int_{C_i} \|x - \mathbf{w}_i\|^2 f(x)dx$

$\hat{\xi}$: estimated distortion error defined as $\frac{1}{p}\sum_{i=1}^n \sum_{x_j \in C_i} \|x_j - \mathbf{w}_i\|^2$

$\varepsilon(t)$: learning rate at time t

$\lambda(t)$ **or** $\sigma(t)$: neighbourhood width at time t

η : elasticity or plasticity

B Online resources

Python code sources

http://www.loria.fr/~rougier/DSOM/dsom.tgz

Movie of self-organisation onto a sphere surface

http://www.loria.fr/~rougier/DSOM/sphere.avi

Movie of self-organisation onto a cube surface

http://www.loria.fr/~rougier/DSOM/cube.avi

Movie of self-reorganisation from sphere to cube surface

http://www.loria.fr/~rougier/DSOM/sphere-cube.avi

Movie of self-reorganisation from one sphere to two spheres surface

http://www.loria.fr/~rougier/DSOM/sphere-spheres.avi

References

[1] J. B. Macqueen, Some methods of classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281–297.

[2] A. B. Y. Linde, R. Gray, An algorithm for vector quantization design, IEEE Trans. on Communications COM-28 (1980) 84–95.

[3] T. Kohonen, Self-organized formation of topologically correct feature maps, Biological Cybernetics 43 (1982) 59–69.

[4] T. M. Martinetz, S. G. Berkovich, K. J. Schulten, Neural-gas network for vector quantization and its application to time-series prediction, IEEE Trans. on Neural Networks 4 (4) (1993) 558–569.

[5] B. Fritzke, A growing neural gas network learns topologies, in: G. Tesauro, D. Touretzky, T. Leen (Eds.), Advances in Neural Information Processing Systems 7, MIT Press, Cambridge MA, 1995, pp. 625–632.

[6] D. Hubel, T. Wiesel, Receptive fields and functional architecture in two non-striate visual areas (18 and 19) of the cat, Journal of Neurophysiology 28 (1965) 229–289.

[7] D. Hubel, T. Wiesel, The period of susceptibility to the physiological effects of unilateral eye closure in kittens., Journal of Physiology 206 (1970) 419–436.

[8] N. Daw, Mechanisms of plasticity in the visual cortex, Investigative Ophthalmology 35 (1994) 4168–4179.

[9] P. B. y Rita, C. Collins, F. Saunders, B. White, L. Scadden, Vision substitution by tactile image projection, Nature 221 (1969) 963–964.

[10] P. B. y Rita, Brain Mechanisms in Sensory Substitution, Academic Press New York, 1972.

[11] V. Ramachandran, D. Rogers-Ramachandran, M. Stewart, Perceptual correlates of massive cortical reorganization, Science 258 (1992) 1159–1160.

[12] M. Oja, S. Kaski, T. Kohonen, Bibliography of self-organizing map (som) papers: 1998-2001 addendum, Neural Computing Surveys 3 (2003) 1–156.

[13] S. Kaski, J. Kangas, T. Kohonen, Bibliography of self-organizing map (som) papers: 1981-1997, Neural Computing Surveys 1 (1998) 102–320.

[14] M. Pöllä, T. Honkela, T. Kohonen, Bibliography of self-organizing map (som) papers: 2002-2005 addendum, Tech. rep., Information and Computer Science, Helsinki University of Technology (2009).

[15] B. Fritzke, A self-organizing network that can follow non-stationary distributions, in: ICANN, 1997, pp. 613–618.

[16] D. Deng, N. Kasabov, Esom: An algorithm to evolve self-organizing maps from on-line data streams, in: Proc. of IJCNN'2000, Vol. VI, Como, Italy, 2000, pp. 3–8.

[17] D. Deng, N. Kasabov, On-line pattern analysis by evolving self-organizing maps, Neurocomputing 51 (2003) 87–103.

[18] S. Furao, O. Hasegawa, An incremental network for on-line unsupervised classification and topology learning, Neural Networks 19 (1) (2006) 90–106.

[19] S. Furao, T. Ogura, O. Hasegawa, An enhanced self-organizing incremental neural network for online unsupervised learning, Neural Networks 20 (8) (2007) 893–903.

[20] R. Keith-Magee, Learning and development in kohonen-style self-organising maps, Ph.D. thesis, Curtin University of Technology (2001).

[21] R. Durbin, D. Willshaw, An analogue approach to the travelling salesman problem, Nature 326 (1987) 689–691.

[22] B. Fritzke, Fast learning with incremental RBF networks, Neural Processing Letters 1 (1) (1994) 2–5.

[23] T. Villman, J. Claussen, Magnification control in self-organizing maps and neural gas, Neural Computation 18 (2006) 446–449.

[24] S. Grossberg, Competitive learning: From interactive activation to adaptive resonance, Cognitive Science 11 (1) (1987) 23–63.

[25] M. Cottrell, J. F. G. Pagès, Theoretical aspects of the som algorithm, Neurocomputing 21 (1998) 119–138.

[26] M. Cottrell, J. Fort, Etude d'un algorithme d'auto-organisation, Annales Institut Henri Poincaré 23 (1) (1987) 1–20.

[27] E. Erwin, K. Obermayer, K. Schulten, Self-organizing maps: Ordering, convergence properties and energy functions, Biological Cybernetics 67 (1992) 47–55.

[28] T. Heskes, Energy functions for self-organizing maps, in: E. Oja, S. Kaski (Eds.), Kohonen Maps, Elsevier, Amsterdam, 1999, pp. 303–315.

[29] D. Buonomano, M. Merzenich, Cortical plasticity: From synapses to maps, Annual Review of Neuroscience 21 (1998) 149–186.