Topology preservation in topographic maps

Susana Vegas-Azcárate
Statistics and Decision Sciences Group
University Rey Juan Carlos
28933 Móstoles, Spain
susana.vegas@urjc.es

Temujin Gautama
Laboratorium voor Neuro- en Psychofysiologie
Katholieke Universiteit Leuven
Campus Gasthuisberg, Herestraat 49, B-3000 Leuven, Belgium
temujin.gautama@philips.com

Marc M. Van Hulle
Laboratorium voor Neuro- en Psychofysiologie
Katholieke Universiteit Leuven
Campus Gasthuisberg, Herestraat 49, B-3000 Leuven, Belgium
marc@neuro.kuleuven.ac.be

Abstract - *The construction of topographic models for fitting a model to the data set requires selection of appropriate values for its parameters. Since the parameters control the flexibility of the model, the generalization problem is directly related to parameter selection. This paper addresses this problem, using a novel criterion for monitoring topographic maps in which the key observation is that we can measure how close to equiprobabilistic a given map is.*

Key words - Self-organizing maps, monitoring studies.

Acknowledgement - *The authors are partially supported by grants from the local CAM Government and the European Council, as well as from Spanish and European agencies and the DMR Foundation's Decision Engineering Lab.*

1 Introduction

Although cluster analysis, non-parametric regression and high-dimensional data visualization are the main statistical applications of the SOM structure, it can also be regarded as a tool for generating non-parametric models of the sampling probability density. In the view of Kohonen (2001), these models should be accurate except perhaps for the fine structure in the target density [Bauer et al. (1996)].

Intuition and common sense may provide 'rules of thumb' in parameters selection, but it would be desirable to have principled methods for making these choices. Topology preservation measures do not depend on local stretching of the lattice but on large-scale violations of the topographic ordering. These metrics, introduced by Bauer and Pawelzik (1992) and further studied by Villmann et al. (1997) among others, can not be used for monitoring the degree of topology-preservation achieved during learning, due to their computational cost. Here we review some aspects of map regularization.

The topographic product metric [Bauer and Pawelzik (1992)] develops the following scheme. Let r_i be the lattice coordinate of neuron i, and let i_k^Q and i_k^V be the kth nearest neighbours of neuron i defined in terms of the Euclidean distance, the first in lattice coordinates ($\mathtt{d}$) and the second in the input space (d). Define the following ratios,

$$Q_1(i,j) = \frac{d[\mathbf{w}_i, \mathbf{w}_{i_j^Q}]}{d[\mathbf{w}_i, \mathbf{w}_{i_j^V}]}, \tag{1}$$

$$Q_2(i,j) = \frac{\mathtt{d}[r_i, r_{i_j^Q}]}{\mathtt{d}[r_i, r_{i_j^V}]}. \tag{2}$$

This way, $Q_1(i,j) = Q_2(i,j) = 1$ only when the nearest neighbour of order j in the input and output space coincide. To avoid the sensitiveness to local variations in the input density, Bauer and Pawelzik (1992) developed the following equation,

$$T(i,j) = \left(\prod_{s=1}^{j} Q_1(i,s) \cdot Q_1(i,s) \right)^{\frac{1}{2 \cdot j}} \tag{3}$$

that results in the definition of the topographic product TP,

$$TP = \frac{1}{M(M-1)} \sum_{i=1}^{M} \sum_{j=1}^{M-1} \log(T(i,j)). \tag{4}$$

Hence, this value is as close as possible to zero for maximum lattice disentangling.

Villmann et al. (1997) have proved that the topographic product can not take into account the shape of the data manifold, since this metric only use neurons' positions. Hence, nonlinear data manifold cannot be distinguished from a folding due to a topographic mismatch between dimensionality of data and lattice. They introduce a measure for the degree of topology preservation of the lattice for linear and nonlinear data manifolds, namely, a topographic function. Haese (1998) use this topographic function to deal with the problem of automatic parameter estimation. Unfortunately, this function is much more complex to implement and requires much more computational effort that the topographic product— a parallelizable iterative procedure, similar to a competitive Hebbian learning procedure, is needed.

Algorithm 1
Self-Organizing Map, sequential version (SOM).

Initialize the weight vectors, $\mathbf{w}_i^0$— using, for example, random samples drawn from the data.
repeat for each iteration, t,
Select a data point, $\mathbf{v}_n$— at random or cycling through the data set.
Find the best matching unit for $\mathbf{v}_n$,

$$i_n^* = \arg\min_i \left\{ \left\| \mathbf{v}_n - \mathbf{w}_i^{t-1} \right\| \right\}. \tag{6}$$

Update weight vectors using a learning rate η and a neighbourhood function $\Lambda(\cdot)$

$$\mathbf{w}_i^t = \mathbf{w}_i^{t-1} + \eta^t \cdot \Lambda_{i_n^* i}^t \cdot (\mathbf{v}_n - \mathbf{w}_i^{t-1}). \tag{7}$$

Decrease the value of η and the width of the neighbourhood function.
until t reaches the maximum number or iterations allowed, say T.

2 SOM training algorithms

The cooperative stage in the Self-Organizing Map [Kohonen (2001)] arises since the algorithm updates not only the winning weight vector, but also those of its nearest lattice neighbours. The neighbourhood function is typically chosen to be a Gaussian function,

$$\Lambda_{i_n^* i} \equiv \Lambda_{i_n^* i}(\sigma_\Lambda(t)) = \exp\left(-\frac{(r_{i_n^*} - r_i)^2}{2 \cdot \sigma_\Lambda(t)^2} \right), \tag{5}$$

where r_i represents the lattice coordinates of the neuron pointing to $\mathbf{w}_i$ and $\sigma_\Lambda(t)$ the monotonically decreasing neighbourhood range. As the width of the neighbourhood function gradually decreases, so does the influence neurons have on their neighbours. Hence, the way the range function decreases is critical.

Thus, the SOM carries out a kind of vector quantization in which the weight vectors constitute the nodes of a flexible network that is fitted to the manifold underlying the input samples. If data independently sampled from the unknown underlying density is assumed available for analysis, the SOM is trained as in Algorithm 1.

When the neighbourhood function vanishes only the weight vector of the winner is updated, and Kohonen's learning rule of Equation 7 becomes identical to the standard unsupervised competitive learning rule (UCL), that is,

$$\Delta \mathbf{w}_{i^*} = \eta \cdot (\mathbf{v}_n - \mathbf{w}_{i^*}). \tag{8}$$

This algorithm is also called the zero-order topology SOM. Moreover, the generalized Lloyd algorithm [Lloyd (1957, 1982)] can be related to the batch version of the UCL rule and the zero-order topology SOM algorithm [Luttrell (1989, 1990)].

Algorithm 2
Self-Organizing Map, batch version (SOM-batch).

Initialize the weight vectors, $\mathbf{w}_i^0$.
repeat for each iteration, t,
  for each data point, $\mathbf{v}_n$, **do**
    Find its the best matching unit, see Equation 6.
  end for.
  Update weight vectors by

$$\mathbf{w}_i^t = \frac{\sum_{n=1}^{N} \mathbf{v}_n \cdot \Lambda_{i_n^* i}^t}{\sum_{n=1}^{N} \Lambda_{i_n^* i}^t}. \tag{9}$$

  Decrease the width of the neighbourhood function.
until t reaches T.

2.1 The Batch SOM

The original version of the SOM is a *sequential* algorithm, which makes a separate update for each data point, taken one at a time. There is also a batch version of the SOM algorithm (SOM-batch) [Kohonen (2001)], for which each update of model parameters is based on all data points. The batch version of Kohonen' SOM algorithm resembles the Linde-Buzo-Gray algorithm for vector quantization [Linde et al. (1980)], which, if a batch computation is considered, is usually called the *k-means* algorithm. Moreover, SOM-batch corresponds to the Heskes' approach when $\beta \to \infty$ and the neighbourhood function verify $\Lambda_{ij} \in \{0, 1\}$. SOM-batch algorithm is summarized in Algorithm 2.
Since SOM-batch contains no learning rate parameter, no convergence problems arise, so more stable asymptotic values for the weight vectors are obtained [Kohonen (2001)]. As in the sequential SOM, a usual choice is a Gaussian-shaped neighbourhood function, see Equation 5.

2.2 Kernel-based Maximum Entropy Learning Rule

The kernel-based Maximum Entropy Learning Rule (kMER) [Van Hulle (2000a)] was introduced as an unsupervised competitive learning rule for non-parametric density estimation. Its main purpose is to obtain equiprobabilistic topographic maps on regular, fixed-topology lattices. Here, neurons receptive fields are radially overlapping symmetric kernels, whose radii are adapted to the local input density together with the weight vectors that define the kernel centroids.

A sequential version of kMER is developed in Van Hulle (1998, 2000a), and can be seen in Algorithm 3. As in the standard SOM algorithm, the neighbourhood function Λ decreases over time to achieve a topographically organized lattice. This function is a critical factor towards the generation of topology-preserving mappings, in as far as it achieves the cooperation and specialization of neighbouring neurons for similar input signals.

There are substantial differences between kMER and original SOM algorithms. Pro-

Algorithm 3
Kernel-based Maximum Entropy Learning Rule (kMER), sequential version.

Initialize the weight vectors $\mathbf{w}_i^0$ and the radii σ_i^0 of the receptive fields.
repeat for each iteration, t,
Select a data point, $\mathbf{v}_n$.
Find the best matching unit for $\mathbf{v}_n$, see Equation 6.
Determine active neurons— the neurons that have a receptive field in which the current input $\mathbf{v}_n$ falls.
Update weight vectors, employing a learning rate η, a neighbourhood function Λ, a sing function Sgn and a fuzzy code membership Ξ,

$$\mathbf{w}_i^t = \mathbf{w}_i^{t-1} + \eta^t \cdot \sum_{j=1}^{M} \Lambda_{i_n^* i}^t \cdot \Xi_j(\mathbf{v}_n) \cdot Sng(\mathbf{v}_n - \mathbf{w}_i^{t-1}). \tag{10}$$

Update radii, using a scale factor ρ and a characteristic function ϑ,

$$\sigma_i^t = \sigma_i^{t-1} + \eta^t \left(\frac{\rho}{M-\rho}(1 - \vartheta_i(\mathbf{v}_n)) - \vartheta_i(\mathbf{v}_n) \right). \tag{11}$$

Decrease the value of η and the width of the neighbourhood function.
until t reaches T.

bably the most relevant is that kMER can minimize an energy function during topographic map formation, so as to produce an equiprobabilistic map. But there are others, such as the optimized criterion (entropy maximization versus distortion minimization) or the type of used receptive fields (overlapping kernel-based receptive fields versus non-overlapping Voronoi receptive fields. Function Ξ of Equation 10 in kMER expands in an effective way the winner-take-all scheme defined by the minimum Euclidean distance rule of the original SOM algorithm. In kMER the receptive fields can overlap and their radii are not directly dependent on other neuron weights. Hence, several neurons can be activated for a given input— there is more than one winner in each competition) or the quality of performance in density estimation (more fair weight distribution versus output unit under-utilization problems. kMER leads to an heteroscedastic and homogeneous mixture density model while SOM only provides a good discrete skeleton for reconstructing the initial density.)

3 Heuristics for SOM algorithms

In the case of the standard SOM, several heuristics have been suggested to guide the choice of σ_Λ. Interesting criteria for the evaluation of the degree of ordering have been developed by Kohonen (1996); Villmann et al. (1994); Zrehen (1993). Kaski and Lagus (1996) and Lampinen and Kostiainen (1999) ideas are next presented.

Kaski and Lagus (1996) provide a wide review on literature, while proposing a goodness measure based on the second best matching unit for a given input vector $\mathbf{v}$. Let us denote

the first and second best matching units for input vector $\mathbf{v}_n$ as i_n^* and i_n^{**}, respectively. Let G_n^J denote a path along the grid, that is, a subset of nodes $\{i_0, i_1, ..., i_J\}$ where each node is an immediate neighbour of the next. Then define a path to link the first and second best matching units, that is, $i_0 = i_n^*$ and $i_J = i_n^{**}$. The basic distance $d(\mathbf{v}_n)$ is defined in terms of the shortest path of this kind :

$$d(\mathbf{v}_n) = \|\mathbf{v}_n - \mathbf{w}_{i_0}\| + \min_i \sum_{j=0}^{J-1} \left\|\mathbf{w}_{i_j} - \mathbf{w}_{i_{j+1}}\right\|. \tag{12}$$

This measure combines a measure of fit, in the first summand, with a measure of continuity, in the second summand. It is now possible to take an average of these values $d(\mathbf{v}_n)$ along the training sample, say $C = \frac{1}{N}\sum_{n=1}^{n} d(\mathbf{v}_n)$. Smaller values of C should be preferred. In Kaski and Lagus (1996) words, "the proposed measure of the goodness of a map can be used to choose maps that do not fold unnecessarily in the input space while representing the input data distribution". It is shown that C can detect the best 1D map when several alternatives, produced by different end range σ_Λ, are tried out. No 2D examples are given in Kaski and Lagus (1996) work. We note that computation of the various paths G_n^J may be computationally demanding in general.

Lampinen and Kostiainen (1999) argue that if the SOM is used as a representation of the joint probability density of the data, then care must be taken not to overfit the model to the data sample. They propose a generalization measure based on the disagreement between training and validation data in the trained SOM. For each of the d variables and each of the M neurons, let $AV_1(i,j)$ denote the average of variable j taken over the data items from the training sample won by unit i. In the same way, $AV_2(i,j)$ is the average of variable j taken over the data items from the validation sample won by unit i. The average

$$AV = \frac{1}{M \cdot d} \sum_{i=1}^{M} \sum_{j=1}^{d} \left(AV_1(i,j) - AV_2(i,j)\right)^2 \tag{13}$$

measures overfitting as a systematic deviation in the projections of training and validation data onto the map [Lampinen and Kostiainen (1999)]. The generalization measure proposed by Lampinen and Kostiainen (1999) includes also another summand involving the similarly defined variances, $Var_1(i,j)$ and $Var_2(i,j)$. The resulting measure can be used for selecting the best generalizing map from several candidates during the search for the optimal hyperparameters. Still, Lampinen and Kostiainen (1999) point out that the main regularization issue is far from settled : "the map may start to overfit as soon as the neighbourhood is reduced to any practical level, indicating that some other forms of regularization may also be needed".

kMER also needs to monitor the degree of topology-preservation achieved during learning [Van Hulle (2000a); Van Hulle and Gautama (2002a,b)]. Van Hulle (2000a) has introduced a methodology based on a neighbourhood cooling scheme to help kMER reach a topology-preservation state.

3.1 Overlap variability, a heuristic for kMER algorithm

For the concrete case of kMER algorithm, which uses overlapping, spherical quantization regions, Van Hulle (2000a) has developed the Overlap Variability metric OV. This

Algorithm 4
A monitoring tool for kMER algorithm based on the overlap variability.

Train the map with a constant neighbourhood range.
Store the obtained disentangled lattice, say Q^0.
Starting from Q^0, perform one complete training with $\gamma^1 = 1$.
Determine the number of epochs, t_{ov}^1, and the corresponding range, $\sigma_{\Lambda,ov}^1$,
for which OV^1 is minimal.
repeat for each monitoring run, $j > 1$,
Perform a new training, starting from Q^0, with $T^j = 2 \cdot t_{ov}^{j-1}$ and

$$\gamma^j = -\frac{\log \frac{0.9 \cdot \sigma_{\Lambda,ov}^{j-1}}{\sigma_{\Lambda(0)}}}{2 \cdot \sigma_{\Lambda(0)}} \tag{15}$$

to cool at a slower rate, but only run the simulation as long as necessary.
Determine the epoch, t_{ov}^j, and the range, $\sigma_{\Lambda,ov}^j$, for which OV^j is minimal.
until $OV^j << OV^{j-1}$.
Do a complete run with $T_{ov} = t_{ov}^j$ and $\gamma_{ov} = \gamma^j$.

metric takes advantage of the overlap to assess, in a heuristic way, the degree of topology-preservation achieved in kMER map. Hence, provided that neurons are equiprobabilistic, a given kMER map is more likely to be disentangled if the number of neurons activated by a given input is constant over the data set. Moreover, if that number is constant the map is also locally smooth. When topological defects arise, the number of activated neurons will be higher than expected in a disentangled area of the lattice. Specifically, OV metric consists of dividing the variability in the number of active neurons by the mean number of activated neurons, and hence, it does not depend on the positions of the weight vectors.

Apart from developing such a metric, Van Hulle (2000b) has presented a monitoring algorithm to control the degree of overlap during learning, yielding a kMER map free of topological defects. He defines the following neighbourhood cooling scheme,

$$\sigma_\Lambda(t) = \sigma_\Lambda(0) \cdot \exp\left(-2 \cdot \sigma_\Lambda(0) \cdot \frac{t}{T_{ov}} \cdot \gamma_{ov}\right), \tag{14}$$

where $\sigma_\Lambda(0)$ is the initial neighbourhood range, γ_{ov} the parameter that controls the slope of the cooling scheme and T_{ov} the number of cycles needed to reach the minimum of the overlap variability metric.

The monitoring tool for kMER algorithm can be seen in Algorithm 4, where superscripts refer to the monitoring run in each case.

3.2 UDL stopping policy

This paper explores the novel monitoring criterion based on the uniformity of the dataload vector, a criterion called UDL— Uniform DataLoad vector [Vegas-Azcárate and Muruzábal (2005); Muruzábal and Vegas-Azcárate (2005)]. Specifically, our UDL criterion

Algorithm 5
UDL monitoring scheme.

Train the map with a constant neighbourhood range.
Store the obtained disentangled lattice, Q^0.
Starting from Q^0, perform one complete training with $\gamma^1 = 1$.
Determine the number of epochs, t^1_{udl}, and the corresponding range, $\sigma^1_{\Lambda,udl}$, for which the speed of decrease of SD^1 function is nearly zero.
repeat, for each monitoring run, $j > 1$,
Perform a new training, starting from Q^0, with $T^j = 2 \cdot t^{j-1}_{udl}$ and

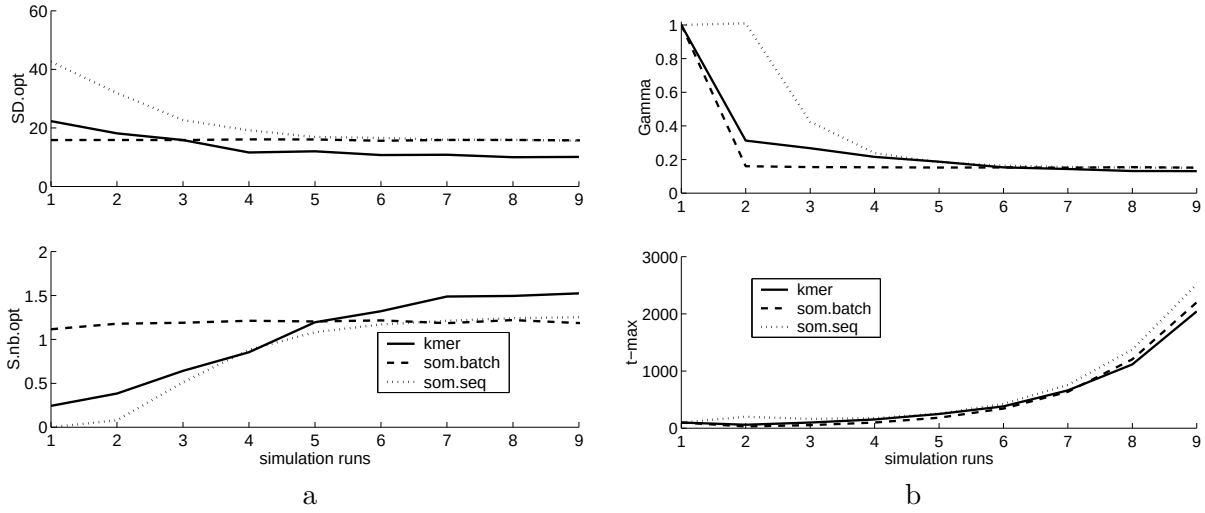
$$\gamma^j = -\frac{\ln \frac{0.9 \cdot \sigma^{j-1}_{\Lambda,udl}}{\sigma_{\Lambda(0)}}}{2 \cdot \sigma_{\Lambda(0)}} \tag{16}$$

to cool at a slower rate, but only run the simulation as long as necessary.
Determine the epoch, t^j_{udl}, and the range, $\sigma^j_{\Lambda,udl}$, for which the speed of decrease of SD^j function is nearly zero.
until $\sigma^j_{\Lambda,udl} \simeq \sigma^{j-1}_{\Lambda,udl}$.
Do a complete run with $T_{udl} = t^j_{udl}$ and $\gamma_{udl} = \gamma^j$.

looks for the moment at which the speed of decrease of the dataload standard deviation function is nearly zero. Pointer density in the trained equiprobabilistic map serves as an estimate of the density underlying the data. Thus, each neuron would cover about the same proportion of the data, leading to a uniform dataload vector. It appears that the stochastic Gaussian behaviour in the equiprobabilistic case can be approximately detected when it is first reached [Vegas-Azcárate and Muruzábal (2005); Muruzábal and Vegas-Azcárate (2005)]. Hence, the associated UDL stopping rule could be stated as follows : quit as soon as the trained map shows the first signs of having reached the reference Gaussian DL distribution—a moment referred to as the UDL stage. In other words, quite as soon as SD function reaches its stability.

Furthermore, minor gains in quantization error brought by training beyond the UDL stage seem to enforce the loss of useful organization, implying fuzzier displays for analysis. Indeed, the UDL stage also signals approximately the beginning of the fine-tuning phase in quantization error [Vegas-Azcárate and Muruzábal (2005); Muruzábal and Vegas-Azcárate (2005)]. Note that during the learning process, the best matching unit of each datum is calculated, so the overhead in obtaining the dataloads and its standard deviation is minimal.

The problem now is to obtain t_{udl} or σ^{udl}_{Λ}, since a complete long enough training has to be done. Indeed, this is not possible with real-world data where 'long enough' is not known. To solve the 'long enough' problem, a scheme similar to the one presented in [Van Hulle (2000a)] is developed here. In it, the rate at which the neighbourhood function range decreases is adjusted during training, ensuring that the final range value is the one desired, σ^{udl}_{Λ}.



a b

FIG. 1 – 4M-50D data set. (a) Optimal values of SD in each of the monitoring run in the upper plot, and stopping criterion in the lower ; (d) Gamma and t_{max} values.

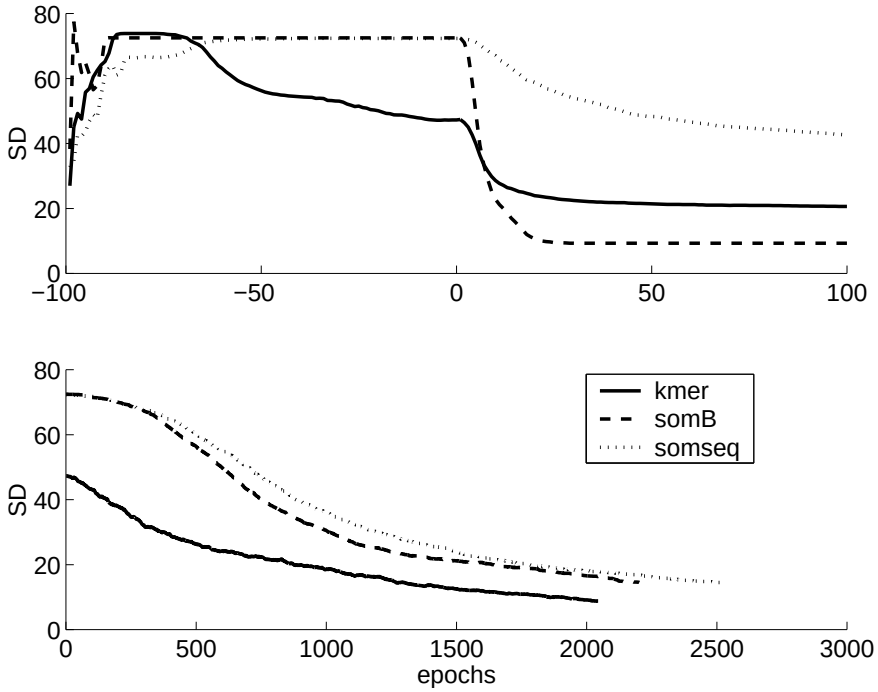


FIG. 2 – 4M-50D data set. Initial and first runs for kMER (solid line), SOM-batch (dashed line) and sequential SOM (dotted line) algorithms, in the upper plot ; and last monitoring runs in the lower.

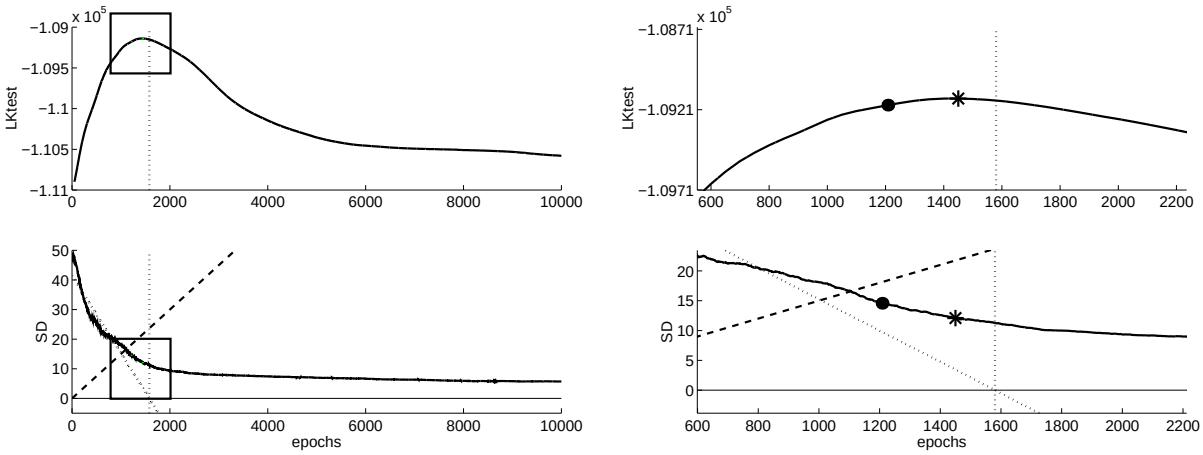


FIG. 3 – kMER on 4M-50D data set. (a) LKtest (upper plot) and SD (lower plot) with the bisector (dashed), the regression (thick dotted) and X-axis (thin solid) lines; (b) zooms of the squared areas of (a), highlighting the maximum of $LKtest$ (asterisks) and the optimal of the monitoring process (big dot).

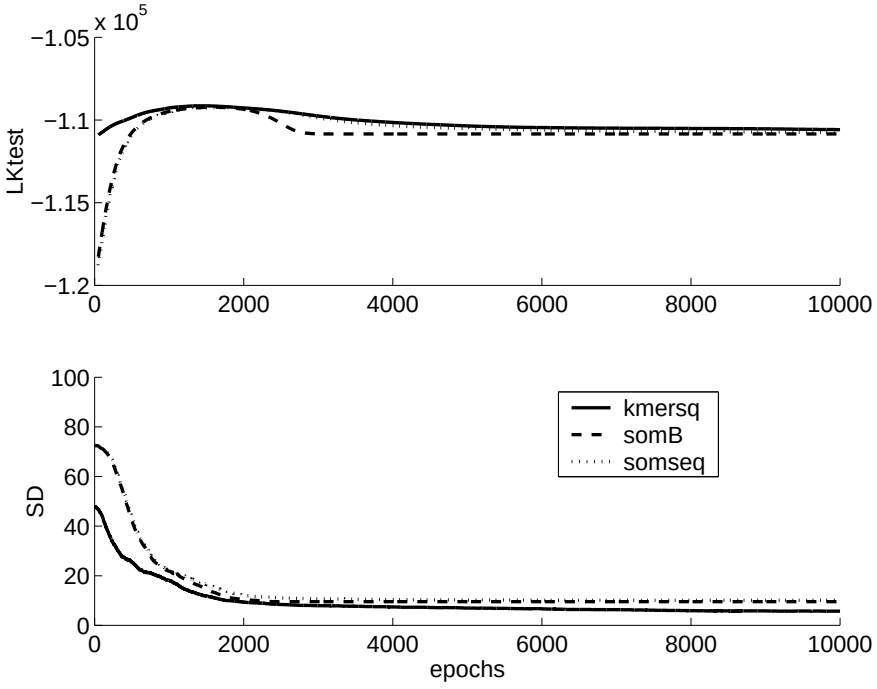


FIG. 4 – 4M-50D data set. Predictive power of kMER (solid line), SOM-batch (dashed line) and sequential SOM (dotted line) algorithms.

Hence, in a finite and affordable number of cycles the same optimal map as the one achieved in an infinitely slow training is obtained. The neighbourhood cooling scheme is developed following,

$$\sigma_\Lambda(t) = \sigma_\Lambda(0) \cdot \exp\left(-2 \cdot \sigma_\Lambda(0) \cdot \frac{t}{T_{udl}} \cdot \gamma_{udl}\right), \tag{17}$$

where γ_{udl} and T_{udl} provide the map that has a range value of σ_Λ^{udl}. Algorithm 5 reflects the proposed UDL monitoring scheme.

The monitoring processes converge, since in a slow enough training the neighbourhood range value at which SD function reaches its stability does exist [Vegas-Azcárate and Muruzábal (2005); Muruzábal and Vegas-Azcárate (2005)]. Hence, if the training algorithm employs a neighbourhood function to ensure topographic ordering, as in the case of SOM-like and kMER algorithms, it is possible to use UDL monitoring scheme.

4 A thorough synthetic example

We consider a thorough synthetic example to study in detail the new proposed UDL monitoring scheme on sequential SOM, SOM-batch and kMER learning rules. The data set consists of four 50D spherical Gaussians, namely 4M-50D, with all the mixture components well-separated, although one of them is more separated from the others.

In Figure 1 the different parameters of the complete monitoring processes are shown. The optimal values of SD function all along the monitoring process reach the expected stability (top plot of Figure 1a), since the optimal neighbourhood range values are selected to converge to $\sigma_\Lambda^{udl} \approx \sigma_\Lambda^{udl}$ (see bottom plot of Figure 1a). In Figure 1a bottom plot it is easily noted that simulation run number 7 provides the optimal neuron configuration for kMER— the stability of σ_Λ^{udl} is reached at that point.

For sequential SOM it is better to wait until run number 8, and for SOM-batch it seems to be enough to run the two first simulations— note that SOM-batch learns with the mean of each Voronoi region so its convergence is always faster. The γ^j values on top plot of Figure 1b also reflect the desired and expected stability— no more changes in the cooling scheme are needed when the optimal neighbourhood range value is reached. t_{udl}^j values on bottom plot of Figure 1b also go after the exponential awaiting shape— the aim of the monitoring process is to reproduce an infinitely slow training with the desired neighbourhood range value.

In Figure 2a the initial, first and last simulation runs are presented. Note that sequential and batch SOM algorithms develop an initial disentangled lattice with the same value that has SD function ($SD(0)$), while in kMER learning rule this initial map seems to be a better map— the slower SD value the better the map. From the first run, it is noticed that sequential SOM learns slower and the batch map performs the best progress.

The aim of working with synthetic data is to check the obtained results. Hence, in Figure 3 kMER learning rule on 4M-50D data set for a long training of 10,000 epochs is presented. It can be appreciated the notable resemblance between the neighbourhood range value at the maximum of $LKtest$ (asterisk) and the one obtained at the end of monitoring run number 7 (big dot). Indeed, quite similar neighbourhood range values at the maximum of $LKtest$ and at the area of SD function stability are obtained.

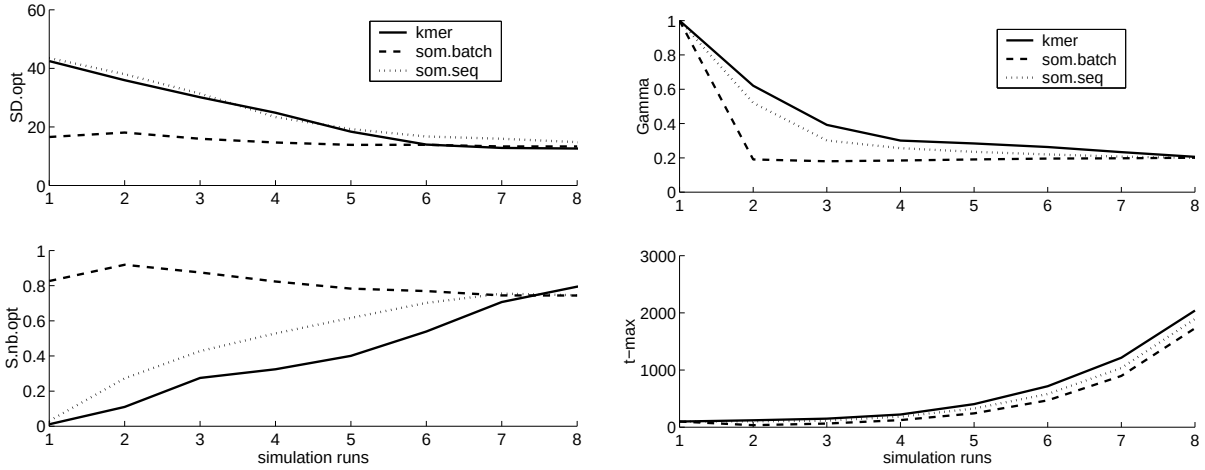


FIG. 5 – Mfeat data set. (a) Optimal values of SD in each of the monitoring runs in the upper plot, and stopping criterion in the lower ; (b) Gamma and t_{max} values.

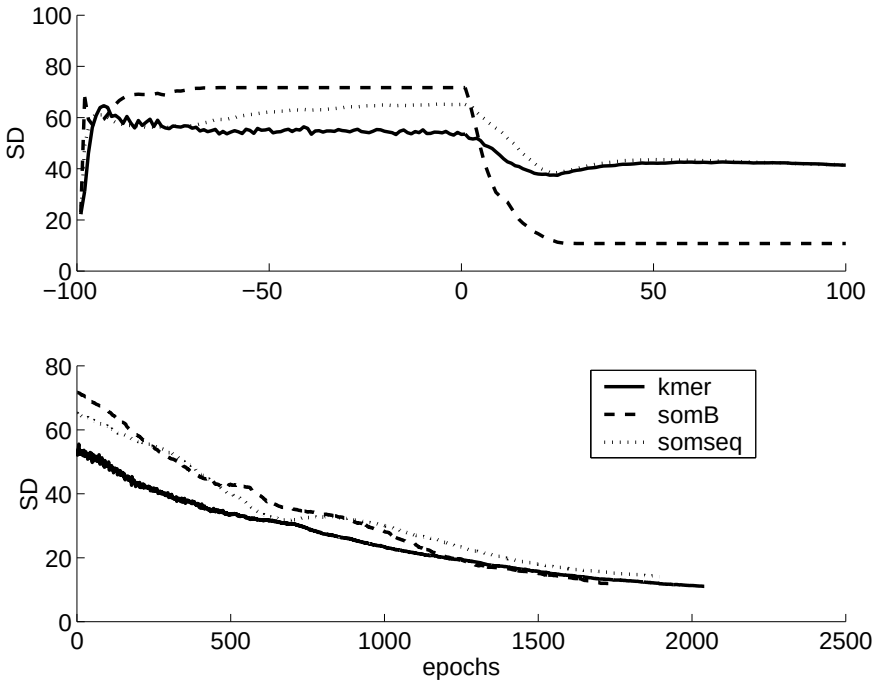


FIG. 6 – Mfeat data set. Initial and first runs for kMER (solid line), SOM-batch (dashed line) and sequential SOM (dotted line) algorithms, in the upper plot ; and last monitoring runs in the lower.

Furthermore, UDL monitoring process reaches the optimal range value at $\sum_{j=0}^{7} t_{max}^{j} = 1,804$ epochs, while 10,000 or more epochs are needed otherwise. Note that data live in a 50 dimensional space, so training 10,000 epochs is very time consuming. Moreover, some test training has to be done to verify if 10,000 is long enough or not.

A complete comparison of the predictive log-likelihoods achieved with each algorithm is presented in Figure 4 top plot. kMER predictive log-likelihood (solid line) is derived from the mixture of variable kernels developed by its learning rule. SOM-type $LKtest$ functions are performed with a variable kernel approach. Here, dashed line for SOM-batch and dotted line for sequential SOM. Notice that the neighbourhood range value that makes $LKtest$ function optimal is the same for the three algorithms (around 1,800 epochs out of 10,000), although kMER algorithm provides the highest predictive value. Figure 4 bottom plot presents SD functions ; these illustrate once more the close relationship between predictive log-likelihood and dataloads standard deviation functions.

5 A real-world example

We now consider the Multiple Features database from the UCI repository [Blake and Merz (1998)] to validate the whole UDL approach. This data set (referred to as Mfeat) consists of features of handwritten numerals $(0, \ldots, 9)$, with 200 patterns per class for a total of 2,000 patterns. Hence, in this case $d = 649$, $N = 2000$ and we look for 10 decision classes.

Figure 6a illustrates the initial, first and last simulation runs, while Figure 5 shows the parameters of the UDL monitoring process. As in the synthetic example, the batch map achieves the fastest learning. Sequential SOM and kMER learning rules seem to follow the same pattern of development, although the slower SD values of kMER lead to a more representative final map. Again, the optimal values of SD function along the monitoring process reach the expected stability (top plot of Figure 5a). On bottom plot of Figure 5a we can appreciate that, simulation run number 8 provides the optimal neuron configuration for kMER, for sequential SOM it is better stop in run number 7, for SOM-batch it is enough to run the two first simulations. Note also that γ^j values on top plot of Figure 5b reflect the desired stability, while t_{max} values (Figure 5b bottom plot) adopt the expected exponential shape.

6 Discussion

Current theoretical frameworks provide no formal assistance for choosing initial values and decrementing schedules for the neighbourhood range function. Only for cases in which input and output space have the same dimension, the global order of the lattice can be uniquely characterized. Moreover, measures of self-organization, such as the topographic product or the topographic function, have their limitations. Furthermore, for any topographic map formation algorithm based on fixed-topology lattices, topological defects may occur during the final stages of training. According to our experience, topographic maps had never been monitored before to reach automatically the maximum predictive likelihood state.

Références

Bauer, H. U., Der, R. and Hermann, M. (1996). Controlling the magnification factor of self-organizing feature maps, in *Neural Computation*, vol. 8, pp. 757–771.

Bauer, H.-U. and Pawelzik, K. (1992). Quantifying the neighborhood preservation of self-organizing feature maps, *IEEE Trans. Neural Networks*, **3** : 570–579.

Blake, C. and Merz, C. (1998). UCI repository of machine learning databases, `http ://www.ics.uci.edu/mlearn/MLRepository.html`.

Haese, K. (1998). Self-organizing feature map with self-adjusting learning parameters, *IEEE Transactions on Neural Network*, **9(6)** : 1270–1278.

Kaski, S. and Lagus, K. (1996). Comparing self-organizing maps, in *Proceedings of ICANN'96, International Conference on Artificial Neural Networks, Lecture Notes in Computer Science*, edited by von der Malsburg, C. and Sendhoff, B., vol. 1112, pp. 809–814, Springer, Berlin.

Kohonen, T. (1996). Self-organizing maps of symbol strings, in *Technical Report A42*, Laboratory of Computer and Information Science, Helsinki University of Technology, Finland.

Kohonen, T. (2001). *Self-Organizing Maps*, Berlin : Springer-Verlag, 3rd extended ed.

Lampinen, J. and Kostiainen, T. (1999). Overtraining and model selection with the self-organizing map, in *Proc. IJCNN'99, Washington, DC, USA*.

Linde, Y., Buzo, A. and Gray, R. (1980). An algorithm for vector quantizer design, *IEEE Trans. Comunication*, **COM-28** : 84–95.

Lloyd, S. (1957). Least squares quantization in PCM, *Institute of Mathematical Statistics Meeting, Atlantic City, NJ*.

Lloyd, S. P. (1982). Least squared quantization in pcm, *IEEE Trans. Inform. Theory IT*, **28(2)**.

Luttrell, S. (1989). Self-organization : A derivation from first principles of a class of learning algorithms, in *Proc. IEEE Int. Joint Conf. on Neural Networks (IJCNN89)(Washington, DC)*, vol. I, pp. 495–498, IEEE Press.

Luttrell, S. P. (1990). Derivation of a class of training algorithms, *IEEE Trans. Neural Networks*, **1** : 229–232.

Muruzábal, J. and Vegas-Azcárate, S. (2005). On equiprobabilistic maps and plausible density estimation, in *5th Workshop On Self-Organizing Maps, Paris*.

Van Hulle, M. M. (1998). Kernel-based equiprobabilistic topographic map formation, *Neural Computation*, **10(7)** : 1847–1871.

Van Hulle, M. M. (2000a). *Faithful representations and topographic maps : From distortion- to information-based self-organization*, New York : Wiley.

Van Hulle, M. M. (2000b). Monitoring the formation of kernel-based topographic maps, in *IEEE Neural Network for Signal Processing Workshop 2000, Sidney*, pp. 241–250.

Van Hulle, M. M. and Gautama, T. (2002a). Monitoring the formation of kernel-based topographic maps with application to hierarchical clustering of music signals, *J. VLSI Signal Processing Systems for Signal, Image, and Video Technology*, **32** : 119–134.

Van Hulle, M. M. and Gautama, T. (2002b). Optimal smoothing of kernel-based topographic maps with application to density-based clustering shapes.

Vegas-Azcárate, S. and Muruzábal, J. (2005). On cluster analysis via neuron proximity in monitored self-organizing maps, in *Workshop on Biosignal Processing and Classification, Barcelona, Spain*.

Villmann, T., Der, R., Herrmann, M. and Martinetz, T. (1997). Topology preservation in self-organizing feature maps : Exact definition and measurement, *IEEE Trans. Neural Networks*, **8(2)** : 256–266.

Villmann, T., Der, R. and Martinetz, T. (1994). A new quantitative measure of topology preservation in Kohonen's feature maps, *In Proc. of the IEEE Int. Conf. on Neural Networks (ICNN'94)*, pp. 645–648.

Zrehen, S. (1993). Analyzing Kohonen maps with geometry, *Proc. Int. Conf. on Artificial Neural Networks (ICANN'93)*, pp. 609–612.