Growing Hierarchical Probabilistic Self-Organizing Graphs

Ezequiel López-Rubio and Esteban José Palomo

***Abstract*— Since the introduction of the growing hierarchical self-organizing map, much work has been done on self-organizing neural models with a dynamic structure. These models allow adjusting the layers of the model to the features of the input dataset. Here we propose a new self-organizing model which is based on a probabilistic mixture of multivariate Gaussian components. The learning rule is derived from the stochastic approximation framework, and a probabilistic criterion is used to control the growth of the model. Moreover, the model is able to adapt to the topology of each layer, so that a hierarchy of dynamic graphs is built. This overcomes the limitations of the self-organizing maps with a fixed topology, and gives rise to a faithful visualization method for high-dimensional data.**

***Index Terms*— Classification, hierarchical self-organization, unsupervised learning, visualization, web mining.**

I. Introduction

DATA analysis is the process of extracting useful information from data. Two kinds of processes can be considered, namely, exploratory and confirmatory data analysis, which focus on the discovery of new knowledge in data and the confirmation of existing hypotheses, respectively. Unsupervised learning methods constitute an important type of exploratory data analysis in several fields such as machine learning, data mining, and pattern recognition. Data clustering is a fundamental unsupervised learning method that has been widely researched across varied disciplines over several decades [1].

The self-organizing map (SOM) [2] has been extensively used as an important tool for data clustering since it performs a mapping between high-dimensional data and a 2-D representation space, where data with similar properties are close together in the representation space, thus preserving the topology of the input data in terms of distances in the output space. This way, the SOM represents the structure of the data as faithfully as possible, facilitating the understanding of such structures. However, this self-organizing model presents a fixed architecture that has to be established prior to training. This task is not always easy, since it requires knowledge about the input data to choose the number of neurons and their arrangement. Moreover, the SOM is not able to capture and represent the inherent hierarchical relations that may present input data. In fact, hierarchical relations are common in many application domains.

The field of self-organization has the growing hierarchical self-organizing map (GHSOM) [3] as one of its most outstanding models. Self-organizing neural networks with a dynamic topology have been proposed [4], [5], although they do not feature a hierarchical structure. This model tries to face the previously commented problems related to the SOM. It consists of several independent growing SOMs [6] arranged in layers, where the number of layers, maps, and neurons are determined during the unsupervised learning process, reflecting the possible hierarchical relations among input data. However, this neural model also shows a number of drawbacks due to its prespecified topology. Firstly, since each map is a growing grid, the topology is always a 2-D rectangular grid and this topology may not be the best that fits the input data. This rigid topology is not a desirable feature of a self-organizing system, as stated in [7]. Secondly, each time the map grows, a row or a column of neurons must be inserted in order to preserve the rectangular topology. As a result, this causes the addition of a large number of neurons without any need, especially with very large maps. Furthermore, the growth of the GHSOM is controlled by two distinct parameters τ_1 and τ_2, which hampers the tuning of the training procedure.

This paper proposes the growing hierarchical probabilistic self-organizing graph (GHPSOG) to solve these limitations present in most self-organizing systems. This new model has a hierarchical architecture composed of dynamic graphs, thus representing the features of the input data with more flexibility and plasticity than fixed-topology maps and also improving visualization capabilities and understanding of the model. On the other hand, our self-organizing model is based on a probabilistic mixture of multivariate Gaussian components. This way, the topology and distribution of the input data is faithfully captured. Moreover, a probabilistic criterion is used to control the growth of the entire architecture. This criterion is tuned by a sole parameter τ, which makes the training simple to tune and understand.

The rest of this paper is organized as follows. Section II presents and explains our new model. Further remarks about stochastic approximation are provided in Appendix A. The visualization methods that can be used with the GHPSOG are commented in Section III. Section IV provides a discussion about the most outstanding features of the GHPSOG model.

Manuscript received July 7, 2010; revised March 28, 2011; accepted March 28, 2011. Date of publication May 12, 2011; date of current version July 7, 2011. This work was supported in part by the Ministry of Science and Innovation of Spain, "Probabilistic Self-Organizing Models for the Restoration of Lossy Compressed Images and Video Project," under Grant TIN2010-15351.

The authors are with the Department of Computer Languages and Computer Science, University of Málaga, Málaga 29071, Spain (e-mail: ezeqlr@lcc.uma.es; ejpalomo@lcc.uma.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2011.2138159

Experimental results are shown in Section V. Finally, Section VI concludes this paper.

II. Model

A GHPSOG is defined as a tree with a variable number of siblings on each node. Each node stores a self-organizing graph with a variable number of units and a flexible topology. Every graph is the child of a unit in the upper layer, except for the top level (root) graph. The definition of the model is divided in several subsections. First we present the basic model for a graph with a fixed number of units and no children (Section II-A), and the corresponding learning scheme (Sections II-B to II-D). Then we explain how the model grows (Section II-E).

A. Basic Model

Every graph of a GHPSOG is associated with a mixture of multivariate Gaussian probability distributions (MoG). In particular, each unit of a graph stores a component of the mixture. The observed (input) space dimension is D, and the likelihood of the observed data $\mathbf{t}$ is modeled by the mixture probability density function of the graph

$$p(\mathbf{t}) = \sum_{i=1}^{H} \pi_i \, p(\mathbf{t} \mid i) \tag{1}$$

where H is the number of mixture components (units), and π_i are the prior probabilities or mixing proportions. The multivariate Gaussian probability density associated with each mixture component i is given by

$$p(\mathbf{t} \mid i) = (2\pi)^{-D/2} \left(\det(\mathbf{C}_i)\right)^{-1/2} \exp\left(-\frac{1}{2}\left(\mathbf{t}-\boldsymbol{\mu}_i\right)^T \mathbf{C}_i^{-1}\left(\mathbf{t}-\boldsymbol{\mu}_i\right)\right) \tag{2}$$

where $\boldsymbol{\mu}_i$ and $\mathbf{C}_i$ are the mean vector and the covariance matrix of mixture component i, respectively.

Along with this probability distribution, there is an adjacency matrix $\mathbf{Z}$ of dimension $H \times H$, whose components z_{ij} are Heskes' confusion probabilities [8]. Here, z_{ij} is the probability that mixture component j has the maximum posterior probability of having generated an input sample $\mathbf{t}$, while in fact $\mathbf{t}$ was generated by mixture component i. Adjacency matrix $\mathbf{Z}$ represents a topology among the mixture components which is adapted to the input data through the learning process, as we see in the next subsection.

B. Learning by Stochastic Approximation

When a new sample $\mathbf{t}_n$ has been presented to the network at time step n, its information should be used to update the mixture components. If we want to update mixture component i with the information from sample $\mathbf{t}_n$, an online learning method for an MoG is required. This strategy has been examined by Sato and Ishii [9] for general (non-self-organizing) Gaussian principal components analysis (PCA) mixtures. A Bayesian model selection is carried out to decide which mixture component has the most posterior probability R_{ni} of having generated the sample if the *a priori* probabilities are assumed to be approximately equal (this assumption is aimed to avoid dead units; see the following subsection). This component is called the *winning unit* j, and it is computed as follows:

$$Winner(n) = \arg\max_i \{p(\mathbf{t}_n \mid i)\} = \arg\max_i \{R_{ni}\} \tag{3}$$

$$R_{ni} = \frac{p(\mathbf{t}_n \mid i)}{\sum_{h=1}^{H} p(\mathbf{t}_n \mid h)} \approx P(i \mid \mathbf{t}_n). \tag{4}$$

Note that we are assuming equal *a priori* responsibilities in (4), this is fully discussed in Section II-C.

Our online method generates the updated values $\boldsymbol{\mu}_i(n)$, $\mathbf{C}_i(n)$ and $\mathbf{Z}(n)$ from the new sample $\mathbf{t}_n$ and the old values $\boldsymbol{\mu}_i(n-1)$, $\mathbf{C}_i(n-1)$ and $\mathbf{Z}(n-1)$. The application of Robbins–Monro stochastic approximation algorithm [10] yields the following update equations for each mixture component i (see Appendix A):

$$\pi_i(n) = (1-\epsilon(n,i))\,\pi_i(n-1) + \epsilon(n,i)\,R_{ni} \tag{5}$$

$$\mathbf{m}_i(n) = (1-\epsilon(n,i))\,\mathbf{m}_i(n-1) + \epsilon(n,i)\,R_{ni}\mathbf{t}_n \tag{6}$$

$$\boldsymbol{\mu}_i(n) = \frac{\mathbf{m}_i(n)}{\pi_i(n)} \tag{7}$$

$$\mathbf{M}_i(n) = (1-\epsilon(n,i))\,\mathbf{M}_i(n-1) + \epsilon(n,i)\,R_{ni}\left(\mathbf{t}_n-\boldsymbol{\mu}_i(n)\right)\left(\mathbf{t}_n-\boldsymbol{\mu}_i(n)\right)^T \tag{8}$$

$$\mathbf{C}_i(n) = \frac{\mathbf{M}_i(n)}{\pi_i(n)} \tag{9}$$

where $\mathbf{m}_i(n)$ and $\mathbf{M}_i(n)$ are auxiliary variables required to update the model parameters. In addition to this, $\epsilon(n,i)$ is a variable step size for the Robbins–Monro algorithm and it is chosen as

$$\epsilon(n,i) = z_{ij}(n)\,\frac{1}{an+b}, \qquad 0 < a < 1 \tag{10}$$

so that it falls smoothly and the method converges. In our experiments, we have found that selecting $b = 1/a$ often yields good results because the network remains plastic long enough: that is, the parameters do not converge too early. So, we have used

$$\epsilon(n,i) = z_{ij}(n)\,\frac{1}{\epsilon_0 n + \frac{1}{\epsilon_0}}, \qquad 0 < \epsilon_0 < 1 \tag{11}$$

where ϵ_0 is near zero because higher values produce an excessive plasticity (variability) of the estimations.

On the other hand, we need a learning rule for the adjacency matrix $\mathbf{Z}$. Only the elements of the jth column of $\mathbf{Z}$ are changed, and they are updated as follows:

$$z_{ij}(n) = (1-\eta(n,i))\,z_{ij}(n-1) + \eta(n,i)\,\psi(n,i) \tag{12}$$

$$\eta(n,i) = \frac{1}{\epsilon_0 n + \frac{1}{\epsilon_0}} \tag{13}$$

where $\eta(n)$ is a variable step size with a decay pattern similar to that of $\epsilon(n,i)$, and ψ is a monotonic function of the rank of the ith unit in the list of all units ordered by R_{ni}. The function ψ models Heskes' confusion probabilities at a particular time

step n. In particular, it is required that

$$\forall n, \quad \sum_{i=1}^{H} \psi(n,i) = 1 \tag{14}$$

$$\forall n \; \forall i,k \in \{1,\ldots,H\}, \; \psi(n,i) \leq \psi(n,k) \Leftrightarrow R_{ni} \leq R_{nk}. \tag{15}$$

Note that the update rule (12) keeps the jth column of $\mathbf{Z}$ normalized in the 1-norm. Hence, z_{ij} are probabilities (Heskes' confusion probabilities) estimated as running averages of the instantaneous values modeled by ψ

$$z_{ij} = P(i \mid j = Winner(n)). \tag{16}$$

In fact, (12) implements another stochastic approximation procedure with $\eta(n)$ as the variable step size instead of $\epsilon(n,i)$, which cannot be used in (12) because that would lead to instability due to a positive feedback effect on z_{ij}.

A proper selection of ψ enforces self-organization by reinforcement of the connections among units with highly correlated values of R_{ni}. In our experiments, we have used $\psi(n,\cdot) = 0.75$ for the winning unit, $\psi(n,\cdot) = 0.25$ for the unit with the second highest value of R_{ni}, and $\psi(n,\cdot) = 0$ for the rest of the units.

We have set the stochastic approximation to proceed during a number of time steps n, which is five times the number of training samples. This way, we adjust the length of the training procedure to the size of the input dataset, so that the algorithm has enough time to adapt to the input, while no computing time is unnecessarily wasted. This choice has not been subject to fine-tuning, and it has been found to yield a correct adaptation of the model across a wide range of datasets, as seen in Section V. Hence it can be expected to generalize well for other datasets.

C. Dead Unit Avoidance

Unit death is defined as the decay to zero of its *a priori* probability π_i, and it is an undesirable situation because it renders the unit useless. There is a cause of unit death in a positive feedback of the *a priori* probability. If π_i grows, then the posterior probability $P(i \mid \mathbf{t}_n)$ also grows. Consequently, the ith mixture component wins the competition more often. In turn, this makes π_i grow. The situation is reversed when π_i diminishes (which leads to 'dead' mixture components). This problem has already been described in detail in the context of probabilistic self-organizing models [11]. Our solution is based on cutting the positive feedback path by assuming that the *a priori* probabilities are approximately equal in the computation of the posterior probabilities R_{ni}. This strategy is implemented by (4). Other mixture learning models manage this problem by pruning the mixture components with small *a priori* probabilities, but in our case this would lead to a catastrophic loss of the valuable topological information stored in the removed mixture components. Our approach works in a way similar to that of the conscience mechanism used in many competitive and self-organizing neural networks [12]–[14].

D. Initialization

Here we outline the initialization procedure to be carried out before the learning algorithm of Section II-B starts. First of all, we divide the available training set randomly into H disjoint subsets with approximately the same size, one subset per unit. Let K be number of input samples allocated to unit i. Then the mixture model of unit i is set up according to those K randomly chosen samples. Hence, we have

$$\pi_i(0) = \frac{1}{H} \tag{17}$$

$$\mathbf{m}_i(0) = \frac{\pi_i(0)}{K} \sum_{j=1}^{K} \mathbf{t}_j \tag{18}$$

$$\boldsymbol{\mu}_i(0) = \frac{\mathbf{m}_i(0)}{\pi_i(0)} \tag{19}$$

$$\mathbf{M}_i(0) = \frac{\pi_i(0)}{K} \sum_{j=1}^{K} \left(\mathbf{t}_j - \boldsymbol{\mu}_i(0)\right)\left(\mathbf{t}_j - \boldsymbol{\mu}_i(0)\right)^T \tag{20}$$

$$\mathbf{C}_i(0) = \frac{\mathbf{M}_i(0)}{\pi_i(0)}. \tag{21}$$

On the other hand, the adjacency matrix is initialized to a constant neutral value for all units

$$\mathbf{Z}_i(0) = \frac{0.8}{H}\mathbf{U} + 0.2\mathbf{I} \tag{22}$$

where $\mathbf{U}$ is a $H \times H$ matrix with all 1s, and $\mathbf{I}$ is the identity matrix of size $H \times H$. This way, every unit is connected to all the others with the same strength. The identity matrix ensures that the self-connections are stronger than the others, which improves learning.

E. Model Growth

The quantization error used for the growing criteria of the classical GHSOM is substituted here by the average negative log-likelihood of the training samples

$$ANLL = -\frac{1}{T} \sum_{n=1}^{T} \log\left(p(\mathbf{t}_n)\right) \tag{23}$$

where T is the number of training samples, and the objective is to minimize the *ANLL*. This is more in line with the probabilistic nature of our model. The graph creation procedure involves two tasks which are executed in sequence.

1) When a new graph is created, the first task to be carried out is the determination of its number of units. We start by training a graph with only one unit. When the training is finished, we compute its *ANLL*, which we denote $ANLL_1$. Then we train a graph with two units, with its corresponding $ANLL_2$. The process continues until the following stopping condition holds

$$\frac{ANLL_H - ANLL_{H+1}}{\text{abs}(ANLL_H)} \leq \tau \tag{24}$$

where τ is called the *growth control parameter*, with $0 < \tau < 1$. That is, we decide that the graph must have H units if and only if a new unit would not make the *ANLL* improve more that τ, where the improvement is

computed as a proportion. If the chosen size is $H = 1$, then the entire graph is pruned (destroyed), since it is not worthwhile to maintain a graph with only one unit.

2) When the number of units of the graph has been determined, the second and final task is executed: namely, the creation of its child graphs. Every unit of the graph is considered as a potential parent unit for a child graph. The set of training samples for the child graph comprises all the samples for which the parent unit is the winner. Consequently, a child graph is created for every unit, i.e., the graph creation procedure is recursively invoked once per unit.

This recursive procedure is initiated with only one root graph in the first level, which is trained with all the available training data. The process terminates when there are no more graphs to be created. The outcome is a hierarchy of graphs of variable size. The root graph has more than one unit, unlike the classical GHSOM.

III. Visualization

There are two visualization methods that can be used with the GHPSOG. In the figures corresponding to both cases, we have used a perceptually adequate color map, as explained in [15]. These two methods are the following.

1) *Input Space Plots*. They are aimed to represent how the GHPSOG units are distributed on the input space. They are useful for $D = 2$ or $D = 3$. For higher values of D, a dimensionality reduction could be carried out by PCA before plotting. Each unit i is represented by its unit Mahalanobis distance ellipse (for $D = 2$) or ellipsoid (for $D = 3$), which is the locus of the points $\mathbf{t}$ which satisfy

$$\left(\mathbf{t} - \boldsymbol{\mu}_i\right)^T \mathbf{C}_i^{-1} \left(\mathbf{t} - \boldsymbol{\mu}_i\right) = 1. \tag{25}$$

On the other hand, we plot the S strongest connections of each unit i (usually the three strongest, possibly including the self-connection) by straight lines between $\boldsymbol{\mu}_i$ and the mean vectors $\boldsymbol{\mu}_j$ of the units j it is connected to. In order to improve readability, the ellipses and connections are drawn in different colors depending on the level in the model hierarchy.

2) *Topology Plots*. They are designed to visualize the structure of the input distribution, as represented by the dynamic topology and the hierarchy of the GHPSOG. They can be used equally well no matter how high is the input dimension D. Therefore, it is a method to explore high dimensional datasets. The plot is built by a recursive algorithm. In order to plot a graph in a certain position and with a certain size, the first step is the extraction of a binary adjacency matrix comprising the S strongest connections of each unit (again, we use $S = 3$ in experiments). Then we use the Graphviz engine [16] to build a planar layout of the subgraph defined by the binary adjacency matrix, so that its bidimensional rendering is as clear and uncluttered as possible. After that, we follow the layout to plot the units and connections of the subgraph so that the obtained drawing is centered in the specified position and has the specified size. Finally, we invoke the plotting procedure recursively for the children of the units of the graph, with half the original size and the coordinates of the unit in the plot as the drawing position.

IV. Discussion

In this section we outline the most outstanding features of the GHPSOG model, and its relations with previous work.

1) The GHPSOG model is able to learn the topology at each level, so that the structure is dynamic not only in terms of the creation of the layers and units but also in the connections among units. This is fundamentally different from many self-organizing models [17]–[21], which are based on maps with a fixed topology (usually rectangular or hexagonal). The visualization capabilities of the model are greatly improved, since the graphs represent the features of the input distribution with more flexibility than fixed topology maps. The GHPSOG model differs from an ordinary Gaussian mixture model in that the amount of learning for each unit i is given by the posterior probabilities of its neighboring units, and not only by its own posterior probability (which would be the case of an ordinary Gaussian mixture model). This leads to the self-organization of the network, i.e., neighboring units cooperate with each other to adapt to the training data. A benefit of this cooperation is that no unit can specialize too much on a small portion of the input data, so that overfitting is avoided. Our online learning approach implies that the results vary depending on the random order of presentation of the input samples, but the standard deviation results for the classification and web mining applications (Sections V-C and V-D) demonstrate that the variability of the network performance is small. On the other hand, batch mode learning could also be employed with the help of the expectation maximization algorithm for mixtures of multivariate Gaussians, although online learning is much more common for self-organizing models [22]; this makes them more easily adaptable to real-time applications where the complete training set is not available.

2) Our proposal defines each graph as a probabilistic mixture model. Hence it allows a wider range of applications than other self-organizing models that do not define probability densities [17], [18]. For example, it is possible to carry out a Bayesian classification. Furthermore, the probabilistic model of each graph can be easily enriched by substituting the mixture component of a unit $p(\mathbf{t} \mid i)$ by the probability distribution of the child graph of that unit $p_{Child(i)}(\mathbf{t})$

$$p(\mathbf{t} \mid i) = p_{Child(i)}(\mathbf{t}). \tag{26}$$

The result of this strategy is a probability density function with more or less mixture components depending on the units that we expand. In other words, we can control the level of detail of the probabilistic model.

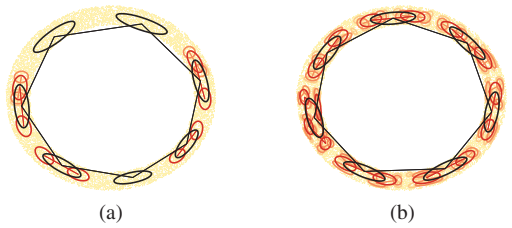


Fig. 1. Ring results. (a) $\tau = 0.02$. (b) $\tau = 0.01$.

3) The growth is controlled by only one user-defined parameter τ. This improves the ease of use with respect to the classical GHSOM, which has two distinct parameters τ_1 and τ_2. Moreover, the recursive nature of the growing algorithm of GHPSOG makes it simple to implement and understand by practitioners not specialized in neural network models. It must be noted that the variables a, b, and ϵ, the values 0.8 and 0.2 in (22), and the function ψ are not tunable, i.e., their values are fixed throughout all experiments. This is because we have found in the tests that no significant improvement can be obtained from changing them in any of the considered applications. Consequently, we regard them as constants rather than parameters. The only adjustable value is the growth control parameter τ, which is the same for all the levels of a GHPSOG network.
4) The GHPSOG model is designed to use multivariate Gaussian mixture components with full covariance matrices. This implies that the training algorithm has $O\left(D^3\right)$ complexity. If we seek to visualize data with high dimension D, it is recommended to carry out a projection of the input data on a subspace with a reduced dimensionality by PCA prior to training. When the GHPSOG model is trained, we unproject the mean vectors to the original input space to view them. This simple technique serves two purposes: it reduces the training time dramatically, and at the same time it removes the data insufficiency problem when the training data is scarce. We have used it in Section III. If the application at hand needs a very fast execution, another possibility to reduce the computational complexity is to use Gaussian mixture components with spherical covariance matrices, which leads to $O(D)$ complexity at the expense of a reduced flexibility [22].

V. Experimental Results

A. Self-Organization Experiments

Several experiments have been performed to show the self-organization capabilities of the GHPSOG. In order to evaluate the unfolding of the maps easily, first we have considered a 2-D input space with 10 000 input samples. Input samples were drawn from an input distribution in the shape of a ring. The training was done with $N = 50\,000$ time steps and two different values for the τ parameter, 0.01 and 0.02, to show the effects of this parameter over the growth process of the model. The resulting architecture is larger as τ disminishes.

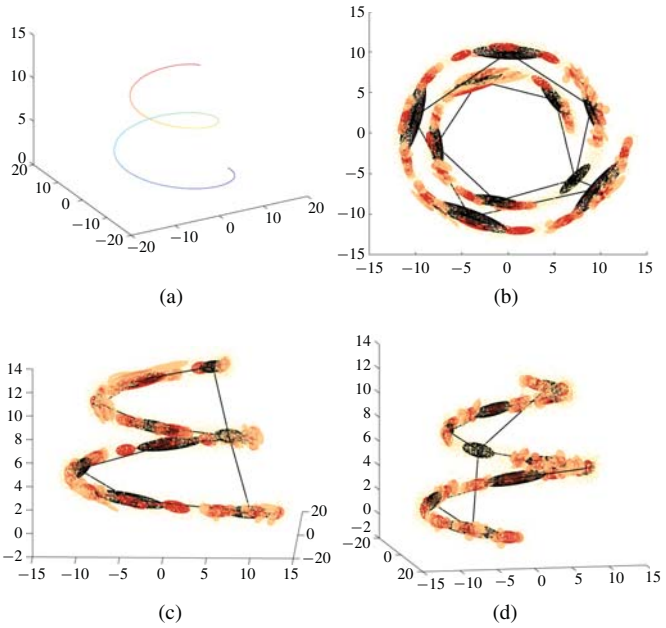


Fig. 2. (a) Tridimensional "Ueda spiral" input distribution. (b)–(d) Corresponding GHPSOG architecture in different views.

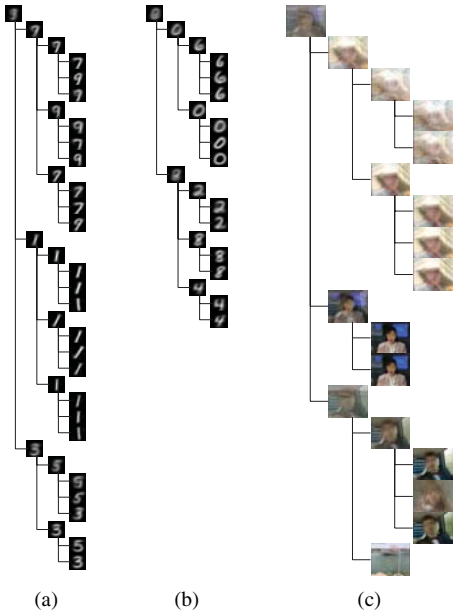
The plots show the state of the network after the training, where neurons are represented by unit Mahalanobis distance ellipses for each mixture component in different colors, depending on the layer they belong to. Connections among the mean vectors of the neurons with the strongest adjacencies are plotted with straight lines, whereas the input samples are plotted with small points. Thus, we can see what subsets of input samples are represented at a higher level of detail and the different effects on the size of the network of choosing different τ values.[1]

The resulting input space plots are shown in Fig. 1. The different colors of each mixture component indicate the layer of the hierarchy to which they belong, the upper layers are painted darker. See Section IV-1 for further information about visualization of input space plots. The plots illustrate how the GHPSOG appropriately fits to the shape of the input distribution. In addition, we can see how the resulting model is larger for $\tau = 0.01$ than for $\tau = 0.02$.

The second part of the self-organization experiments consists in a test of the unfolding of the GHPSOG model for a manifold input distribution with $D = 3$. There are well-known 3-D distributions which are based on manifolds [23]. Here, the "Ueda spiral" manifold with added Gaussian noise has been selected. Again, we have drawn 10 000 training samples from this distribution, and the GHPSOG model was trained during 50 000 time steps with $\tau = 0.002$.

The original noiseless manifold and the resulting GHPSOG model in different views are shown in Fig. 2. In the figures, each mixture component has been represented as a unit Mahalanobis distance ellipsoid, the three strongest adjacencies of the neuron as straight lines, and the training samples as small

[1]Demo videos of the training process are available as supplementary data. The source code of our method can be downloaded from http://www.lcc.uma.es/%7Eezeqlr/ghpsog/ghpsog.html.



(a) (b) (c)

Fig. 3. Visualization of high-dimensional data. MNIST (a) odd and (b) even handwritten digits, and (c) video frames. The root nodes (upper left) show the global mean of the distribution, while the rest of nodes show the mean vectors $\boldsymbol{\mu}_i$ of the units of the GHPSOG hierarchy.

points. Both ellipsoids and adjacencies have been painted with the color of the layer which they belong to, the upper layers are painted darker. In the figures, we can see how the neurons from the first layers are fewer than the rest from deeper layers and provide a worse representation: that is, the deeper the hierarchy the better adaptation of the neurons to the 3-D manifold.

B. Visualization of High-Dimensional Data

This subsection is devoted to illustrate the abilities of the GHPSOG to visualize high-dimensional datasets by discovering their hierarchical structure. This time, we have used $\tau = 0.03$ in order to get smaller hierarchies which are easier to plot and understand. Two experiments have been carried out, which we describe next.

Our first database is the handwritten digit database from MNIST [24], which has a dimensionality $D = 784$. In order to remove unnecessary information, we have reduced its dimensionality to $D = 15$ by PCA prior to training. This technique allows us to project back the GHPSOG mean vectors $\boldsymbol{\mu}_i$ to the original input space in order to visualize the structure of the dataset. Fig. 3 depicts the final state of two GHPSOG networks after training, one for odd digits [(a), 30 508 input samples] and the other for even digits [(b), 29 492 input samples). As seen, the networks are able to discover the structure of the digit datasets, so that the samples are grouped by their writing style and the digit they represent. As seen, slanted digits are distinguished from non-slanted digits very early in the hierarchical structure.

We have built our second database by taking all the frames from the *Akiyo*, *Carphone*, *Container*, *Foreman*, and *Suzie* benchmark video sequences [25]. This amounts to a total of 1432 input samples, each with dimensionality $D = 25344$ (176×144 pixel frames with three color channels). This time, we have reduced the dimensionality of the input samples to $D = 8$, since we have less variability; given the very high original dimension and the small number of input samples, the PCA has been performed by means of the eigenface method [26]. This method has small computational complexity: our unoptimized MATLAB implementation computes the PCA in 2.7 s (MNIST datasets) and 14.9 s (video dataset) on a single core of a 2.8-GHz CPU of a standard notebook PC. For very large datasets, the complexity can be further reduced by specialized techniques that do not carry out an eigenvalue decomposition, such as fast PCA [27]. Fig. 3(c) shows the final state of the GHPSOG. We can observe that the network clusters the video frames by the sequence they belong to.

It must be remarked that the GHPSOG networks are able to find the underlying groups and the intrinsic structure of these input datasets without any information about the number or characteristics of the groups present in the datasets, i.e., the learning is fully unsupervised.

C. Classification

Classification is a common application in the field of self-organization. In order to show the classification capabilities of the GHPSOG, we have chosen ten different standard benchmark datasets from the UCI Machine Learning Repository [28]. We have run 10-fold cross-validations, where each dataset was divided into as many data subsets as there are classes, so that a GHPSOG for each class of every dataset has been trained. For training, we set the τ parameter to 0.02, which yields medium-size neural architectures, and the number of epochs has been 2. In addition, for each of the 10 folds, we randomly selected the 90% data subset for training and the remaining 10% data subset for testing. In order to implement a Bayesian classifier with the GHPSOG models, the test samples $\mathbf{t}_n$ are assigned to the class i corresponding to the GHPSOG with the highest probability $P(i|\mathbf{t}_n)$.

In order to evaluate the classification performance of the GHPSOG, we have followed the methodology proposed by [29] to assess self-organizing models in classification, which considers several quality measures. The quality measure we have used is the classification accuracy, which is a measure of the amount of input samples correctly classified, where the higher the accuracy value, the better the accuracy. These values are in [0, 1] and are determined as

$$Acc = \frac{\sum_{i=1}^{N_C} N_{ii}}{N} \tag{27}$$

where N_{ii} is the number of input samples assigned to a map i which belong to class i, N_C is the number of classes (which is the same as the number of SOM), and N is the total number of input samples given by

$$N = \sum_{i=1}^{N_C} \sum_{j=1}^{N_C} N_{ij}. \tag{28}$$

The test results of the 10-fold cross-validations are given in Table I, where the standard deviations are shown in parentheses. The table also provides a comparison between our results

TABLE I
CLASSIFICATION ACCURACY (HIGHER IS BETTER). STANDARD DEVIATIONS IN PARENTHESES

	GHPSOG	SOMM	KBTM	t-GTM
BalanceScale	**0.9211 (0.0432)**	0.8868 (0.0368)	0.8672 (0.0391)	0.7276 (0.1393)
BreastCancer	0.9579 (0.0194)	0.9596 (0.0212)	0.9751 (0.0180)	**0.9827 (0.0135)**
Cloud	**1.000 (0.000)**	**1.000 (0.000)**	0.9945 (0.0038)	0.9991 (0.0020)
Dermatology	**0.9651 (0.0247)**	0.8997 (0.0399)	0.4018 (0.0750)	0.3446 (0.1443)
Liver	**0.6949 (0.0502)**	0.6289 (0.0727)	0.5843 (0.0555)	0.5852 (0.1188)
Magic	**0.8203 (0.0077)**	0.7705 (0.0062)	0.6902 (0.0096)	0.7699 (0.0057)
Shuttle	**0.9612 (0.0721)**	0.7871 (0.0060)	0.8735 (0.0037)	0.7988 (0.0122)
Spambase	**0.6888 (0.0185)**	0.6553 (0.0156)	0.6608 (0.0256)	0.6539 (0.0167)
Waveform	0.8531 (0.0082)	**0.8676 (0.0130)**	0.8063 (0.0131)	0.8163 (0.0357)
Wine	**0.9817 (0.0303)**	0.7239 (0.0836)	0.7618 (0.0614)	0.4716 (0.2039)

TABLE II
TRAINING TIMES FOR THE CLASSIFICATION EXPERIMENTS (IN SECONDS)

	GHPSOG	SOMM	KBTM	t-GTM
BalanceScale	**50.95**	356.95	2146.42	155.53
BreastCancer	**39.88**	241.83	1784.59	173.89
Cloud	**63.58**	789.45	2398.05	511.83
Dermatology	**105.57**	390.66	5697.47	117.98
Liver	**50.01**	128.07	1512.63	88.30
Magic	**2817.10**	6345.54	6670.56	4591.34
Shuttle	**5712.55**	15404.14	16161.99	11261.09
Spambase	**481.66**	1688.19	1614.83	1137.64
Waveform	**709.04**	1722.73	1828.64	1233.31
Wine	**15.53**	95.48	2425.29	48.58

TABLE III
DATASET CATEGORIES AND THEIR ASSOCIATED THEMES

ID	Dataset category	Associated theme
A	Commercial Banks	Banking and Finance
B	Building Societies	Banking and Finance
C	Insurance Agencies	Banking and Finance
D	Java	Programming Languages
E	C/C++	Programming Languages
F	Visual Basic	Programming Languages
G	Astronomy	Science
H	Biology	Science
I	Soccer	Sport
J	Motor Sport	Sport
K	Other Sports	Sport

and those achieved by other probabilistic self-organizing models. In particular, the same datasets have been presented to the self-organizing mixture model (SOMM) in [19], the joint entropy maximization kernel based on topographic maps (KBTM) in [21], and the t-GTM by [30]. The degrees of freedom of ν parameter for t-GTM has been optimized, and the best results for every considered model are shown. Note that the GHPSOG results are better than the rest of models in 8 of the 10 datasets. For the remaining two datasets, we did not outperform the other models, although the results were quite acceptable. Moreover, Table II shows the training times for the compared models. They have been measured over a single core of a 2.66-GHz CPU (64-bit $\times$ 86 architecture) with 2 GB RAM, all the models have been implemented as MATLAB scripts. As seen, GHPSOG achieves competitive results.

D. Web Mining

This designed set of experiments for the GHPSOG consists in carrying out a web mining process, allowing us to evaluate the algorithm in clustering tasks. One problem related to web document clustering is the lack of standard datasets since each research work uses its own datasets. Because of this, comparison among different works is difficult. For our experiments, the "BankSearch" benchmark dataset has been chosen to test the GHPSOG. This is a web document dataset proposed in [31] as a standard dataset against which different techniques can be benchmarked and assessed in comparison to each other.

This dataset was chosen from the Open Directory Project and Yahoo! Categories, which have already been manually categorized. It consists of 11000 documents that correspond to 11 categories (1000 documents per category). Each category has an associated theme, namely "Banking and Finance," "Programming Languages," "Science," and "Sport." The 11 categories are the following: "Commercial Banks," "Building Societies," "Insurance Agencies," "Java," "C/C++," "Visual Basic," "Astronomy," "Biology," "Soccer," "Motor Sport," and "Other Sports." The dataset categories and their associated themes are shown in Table III.

Each document has several features that represent the frequency of the most frequent words in a set of documents. To extract these features for a determined data subset, we followed the process shown by [31]. For each document, first we consider all words that appear at least once in the document. Second, stop words listed in our stop-word list were removed. Our stop-word list is the same as used in [32], which is formed by 319 words. Third, the frequency of each remaining word in the document is stored. Once all the documents of the dataset were thus processed, a master word list that contains every word in the combined data subset, associated with its overall frequency, is created. Then the

TABLE IV
CLUSTERING ACCURACY (HIGHER IS BETTER). STANDARD DEVIATIONS ARE IN PARENTHESES

		GHPSOG		GHSOM		GTM
	$h(\%)$	$\tau = 0.01$	$\tau = 0.02$	$\tau_1 = 0.1$	$\tau_1 = 0.2$	
A&I	0.5	**0.8876 (0.0282)**	0.8352 (0.0497)	0.6503 (0.0279)	0.6467 (0.0257)	0.7874 (0.0257)
	1	**0.8420 (0.0395)**	0.7617 (0.0665)	0.6657 (0.0187)	0.6504 (0.0211)	0.7109 (0.0246)
	1.5	0.7896 (0.0434)	**0.7962 (0.0739)**	0.6726 (0.0179)	0.6409 (0.0325)	0.7455 (0.0215)
	2	**0.8622 (0.0341)**	0.7858 (0.0692)	0.6656 (0.0403)	0.6490 (0.0177)	0.7345 (0.0212)
B&C	0.5	**0.8719 (0.0205)**	0.8631 (0.0231)	0.6993 (0.0809)	0.6773 (0.0265)	0.8217 (0.0168)
	1	**0.8711 (0.0189)**	0.8459 (0.0538)	0.6908 (0.0280)	0.6790 (0.0470)	0.7954 (0.0562)
	1.5	**0.8704 (0.0203)**	0.8320 (0.0587)	0.6998 (0.0339)	0.6423 (0.0410)	0.8113 (0.0340)
	2	**0.8511 (0.0465)**	0.8285 (0.0450)	0.7162 (0.0189)	0.6590 (0.0341)	0.8200 (0.0350)
All	0.5	**0.2631 (0.0088)**	0.2144 (0.0200)	0.1802 (0.0169)	0.1529 (0.0054)	0.1974 (0.0082)
	1	**0.2524 (0.0261)**	0.2237 (0.0191)	0.1574 (0.0072)	0.1623 (0.0086)	0.1880 (0.0084)

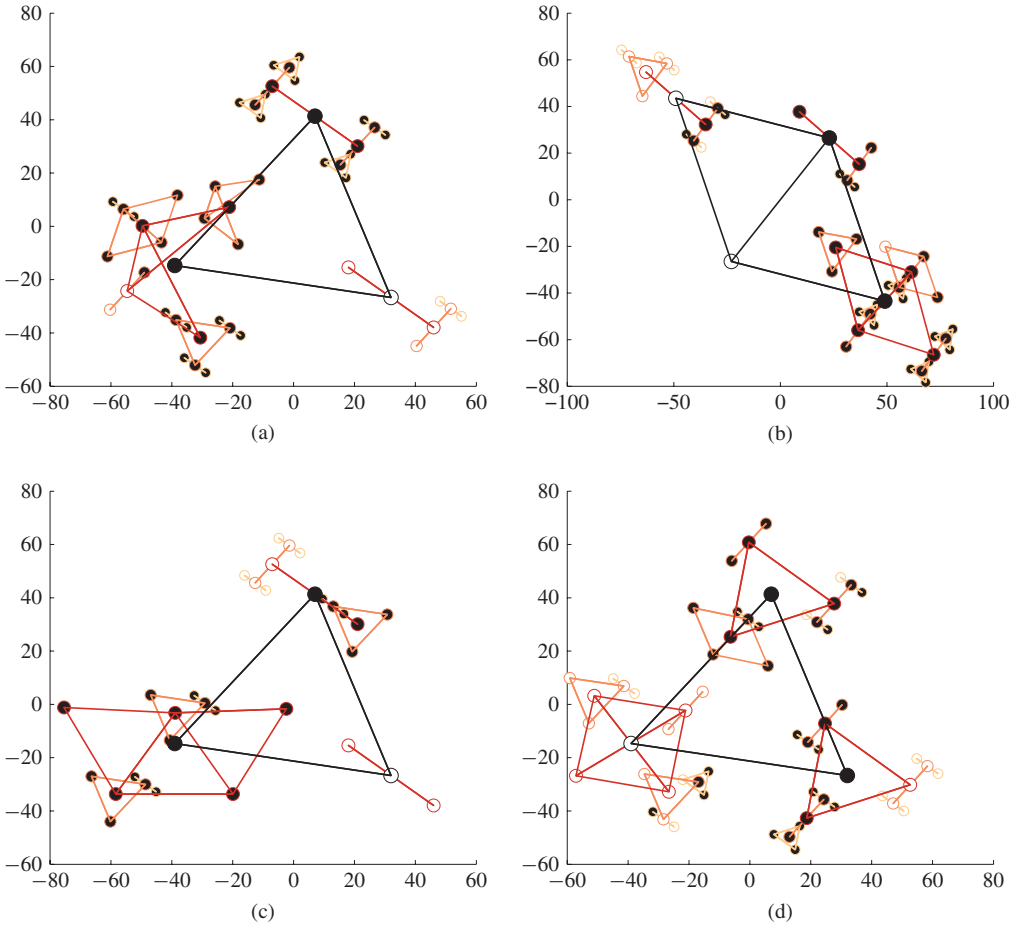


Fig. 4. Topology plots for the $A\&I$ dataset trained with $\tau = 0.02$ and (a) $h = 0.5$; (b) $h = 1$; (c) $h = 1.5$; and (d) $h = 2$. The color of the neurons indicates the class label assigned to the clusters (black for the A dataset and white for the I dataset).

master list is cut down to contain only the top $h\%$ of most frequently occurring words. Finally, a feature vector v_i was created for each document i, such that the jth element in v_i was w_{ji}/s_i, where w_{ji} is the number of occurrences in document i of the jth most frequent word in the combined data subset, and s_i is the total number of words in document i.

For our experiments, the datasets $A\&I$ ("Commercial Banks" and "Soccer") and $B\&C$ ("Building Societies" and "Insurance Agencies") were selected, where the first dataset has associated two different themes ("Finance" and "Sport") whereas the second one has associated the same theme ("Finance"). In addition, the entire dataset with the 11 categories was considered. The GHPSOG models were trained during 2000 time steps for the two first datasets and during 22 000 for the entire dataset. The training was done by using these datasets with different settings, specifically, four h values were chosen: 0.5, 1, 1.5, and 2. Also, the GHPSOG models were trained with two different τ values: 0.02 and 0.01. These

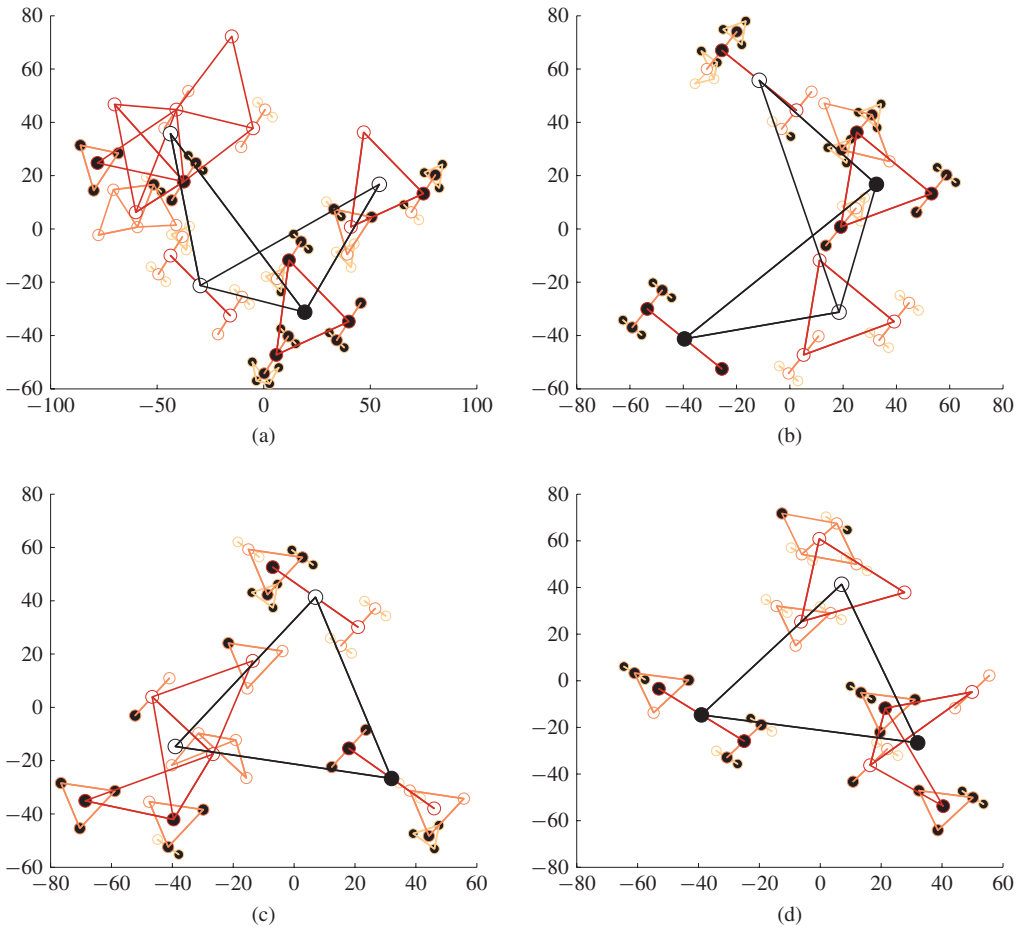


Fig. 5. Topology plots for the $B\&C$ dataset trained with $\tau = 0.02$ and (a) $h = 0.5$; (b) $h = 1$; (c) $h = 1.5$; and (d) $h = 2$. The color of the neurons indicates the class label assigned to the clusters (black for the B dataset and white for the C dataset).

TABLE V
PSNR RESULTS FOR IMAGE DEBLOCKING. THE BEST RESULT IS SHOWN IN BOLD

	JPEG	SA-DCT	FOE	GHPSOG
Baboon	19.2794	19.9998	19.7582	**20.0247**
Lake	21.1574	22.3669	22.1896	**22.5123**
Lena	23.3423	24.7947	24.8305	**25.3666**
Peppers	22.3154	**23.7945**	22.6068	23.7732
Tiffany	23.4779	24.5756	23.5629	**24.8897**

values of the τ parameter achieve a good tradeoff between the performance and the size of the network, allowing us at the same time to compare the effects of choosing different architecture sizes on the results.

In order to evaluate the performance of the GHPSOG in clustering web documents, we have again used the classification accuracy as quality measure as proposed by [29, Sec. VI-B]. To this end, each cluster has been labeled with the class that is the largest in the cluster, so the cluster represents that class. For each data subset, we have run 10-fold cross-validations for each model and dataset setup. Results achieved by our model are given in Table IV. A comparison between our model, the GHSOM model, and the GTM is also provided in the Table. We chose the GHSOM [3] and the GTM [33] models in order to compare the GHPSOG with another growing hierarchical self-organizing model and another probabilistic model, respectively.

The GHSOM models were trained with same datasets used for the GHPSOG. The GHSOM has two parameters, τ_1 and τ_2, to control the growing of maps and the expansion of neurons into maps, respectively. For this experiment, we run simulations with values for the τ_1 parameter 0.1 and 0.2, since τ_1 is the most decisive parameter in the GHSOM model. We fixed the τ_2 parameter to 0.3, since it yields architecture sizes similar to those obtained for the GHPSOG models. As shown in Table IV, the GHPSOG achieves better results than the GHSOM for web document clustering. Note that the results are better for $\tau = 0.01$, which generates bigger neural architectures in terms of neurons than those obtained for $\tau = 0.02$. Moreover, the results for $h = 0.5$ are generally better than for other h values.

Finally, some topology plots are shown to visualize the structure of the input distribution as represented by the GHPSOG (see Section IV-2). As an example, we chose to represent the topology of the GHPSOG models trained with the four setups of the $A\&I$ and $B\&C$ datasets for $\tau = 0.02$. These topology plots are shown in Figs. 4 and 5. In the figures, each neuron is represented by a circle, and the three strongest

TABLE VI
MAE RESULTS FOR IMAGE DEBLOCKING. THE BEST RESULT IS SHOWN IN BOLD

	JPEG	SA-DCT	FOE	GHPSOG
Baboon	21.1876	19.4878	20.1489	**19.4768**
Lake	16.9985	14.6086	14.9088	**14.3133**
Lena	13.3468	11.3367	11.1059	**10.4907**
Peppers	14.4370	**12.0603**	13.9406	12.0674
Tiffany	11.9943	10.5609	11.7163	**10.2788**

TABLE VII
SSIM RESULTS FOR IMAGE DEBLOCKING. THE BEST RESULT IS SHOWN IN BOLD

	JPEG	SA-DCT	FOE	GHPSOG
Baboon	0.4262	0.4319	0.4294	**0.4416**
Lake	0.5637	0.6288	0.6194	**0.6336**
Lena	0.6445	0.7180	0.7125	**0.7254**
Peppers	0.5539	0.6499	0.6373	**0.6593**
Tiffany	0.6440	0.7230	0.7060	**0.7297**

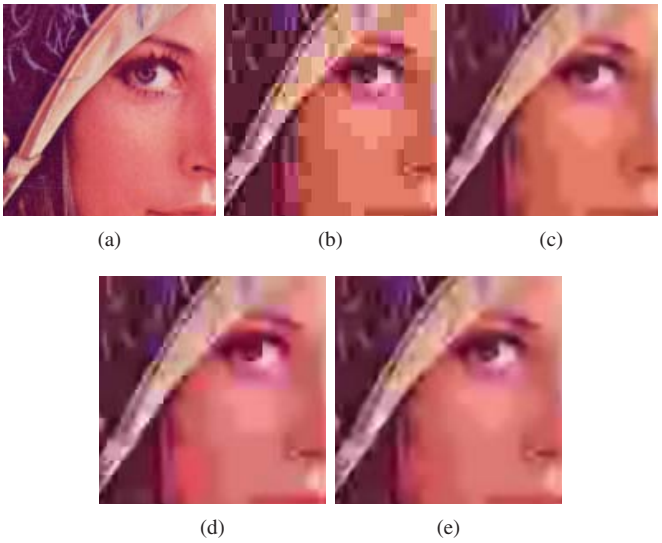


Fig. 6. Image deblocking results for *Lena* (detail). (a) Original. (b) JPEG. (c) SA-DCT. (d) FOE. (c) GHPSOG.

adjacencies of each neuron are represented by straight lines. Both the colors of the neuron edges and their connections have the color of the layer to which they belong. We show only the first four layers to simplify the visualization and understanding of the plots, the upper layers are painted darker. The color of the neuron surfaces are black or white, depending on whether the class assigned to the neuron is either A or I for the first dataset, or B or C for the second, respectively.

E. Image Deblocking

There is a wide range of techniques to reduce the blocking artifacts caused by JPEG lossy image compression [34]–[36]. Here we employ the GHPSOG model to provide an alternative approach for this problem. Let us assume that we have two deblocked images coming from two distinct deblocking algorithms, here we will consider the shape-adaptive discrete cosine transform (SA-DCT, [35]) and the fields of experts (FOE, [36]). First we clip the enhanced DCT values so that they fulfill the *narrow quantization constraint* (NQC, [34]), i.e., we ensure that the obtained image belongs to a reduced version of the set of images which could have given rise to the input JPEG file

$$\hat{q}_j(\mathbf{x}_i) = \min\left(q_j(\mathbf{x}_i) + 0.3, \max\left(q_j(\mathbf{x}_i) - 0.3, \tilde{q}_j(\mathbf{x}_i)\right)\right) \tag{29}$$

where $\mathbf{x}_i$ comprises the coordinates of a 8×8 pixel JPEG compression block, $j \in \{1, ..., 64\}$ is the index into the block, $q_j(\mathbf{x}_i)$ is the JPEG quantized DCT value, $\tilde{q}_j(\mathbf{x}_i)$ is the enhanced value, and $\hat{q}_j(\mathbf{x}_i)$ is the clipped value. Since the NQC defines a convex hypervolume in the DCT domain, we might consider alternative solutions that lie in the straight line segment which connects the two clipped solutions, all of which satisfy (29)

$$\dot{\mathbf{q}}(\mathbf{x}_i) = \hat{\mathbf{q}}_{SA-DCT}(\mathbf{x}_i) + \lambda\left(\hat{\mathbf{q}}_{FOE}(\mathbf{x}_i) - \hat{\mathbf{q}}_{SA-DCT}(\mathbf{x}_i)\right), \ \lambda \in [0, 1]. \tag{30}$$

In the experiments, we have chosen 10 alternative solutions equally spaced on the line segment. In order to decide which of them is the best, we train a GHPSOG with 64-dimensional vectors $\mathbf{q}(\mathbf{x}_i)$ of JPEG quantized coefficients. This way, the GHPSOG builds an estimator of the probability density function $p(\mathbf{q}(\mathbf{x}_i))$ of the distribution of quantized DCT blocks. The network training is carried out offline, so that it does not slow down the deblocking process. On the other hand, the training set is obtained from images that are not going to be deblocked to avoid any overfitting of the probabilistic model. The selected image is that which minimizes the log-likelihood L of observing the restored coefficients $\dot{\mathbf{q}}(\mathbf{x}_i)$, given the distribution of JPEG quantized DCT blocks

$$L = \sum_{i=1}^{B} \log p(\dot{\mathbf{q}}(\mathbf{x}_i)) \tag{31}$$

where B is the number of blocks in the image. This way, we are selecting the image that departs the most from the distribution of artifact-full quantized blocks $\mathbf{q}(\mathbf{x}_i)$. Next we smooth the selected image by iterative steering kernel regression [37], [38], and finally we clip the DCT coefficients to enforce (29) so as to avoid excessive departure from the compressed image.

We have tested our method over several benchmark images from USC-SIPI image repository [39]. The F16 image has been used to train the GHPSOG model, and then the restoration performance has been measured over five other images for JPEG compression quality parameter $Q = 4$.

For the quantitative assessment, three standard criteria have been chosen [40]: namely, the power signal-to-noise ratio (PSNR) (higher is better), the mean absolute error (MAE) (lower is better), and the structural similarity index (SSIM) (higher is better). Tables V–VII demonstrate that our approach consistently outperforms the two deblocking algorithms it is based on, namely SA-DCT and FOE. The qualitative results for *Lena* (Fig. 6) confirm this.

VI. Conclusion

We have presented a new probabilistic self-organizing model, which is based on the stochastic approximation framework. It is able to build a hierarchy of self-organizing graphs, each with a dynamically learnt topology. These features enhance the capabilities of the model to represent an input distribution faithfully. Moreover, visualization of high-dimensional datasets is facilitated. We have designed two visualization methods for this purpose, which highlight the learnt hierarchy and connections among units.

We have proven that our proposal is able to compensate for a large class of transformations on the input data; this is particularly useful for data with unusual scaling. Finally, experimental results have been carried out, which demonstrate its self-organization ability and its suitability for classification and web mining applications.

Appendix

Stochastic Approximation

Here we derive an stochastic approximation algorithm to learn the parameters of the probabilistic mixture corresponding to a graph. Let $\boldsymbol{\Theta}_i = \left(\pi_i, \boldsymbol{\mu}_i, \mathbf{C}_i\right)$ be a vector comprising the parameters for mixture component i. The following derivation relies on the methodology proposed in [41] and [42]. Let $\varphi\left(\boldsymbol{\Theta}_i, \mathbf{t}\right)$ be an arbitrary function of $\boldsymbol{\Theta}_i$ and the input sample $\mathbf{t}$. Then we define the weighted mean of $\varphi\left(\boldsymbol{\Theta}_i, \mathbf{t}\right)$ as

$$< \varphi >_i \;\; = \;\; E\left[P\left(i \mid \mathbf{t}\right) \varphi\left(\boldsymbol{\Theta}_i, \mathbf{t}\right)\right]. \tag{32}$$

This allows expressing the conditional expectation of $\varphi\left(\boldsymbol{\Theta}_i, \mathbf{t}\right)$ as follows:

$$\begin{aligned} \frac{< \varphi >_i}{< 1 >_i} &= \frac{E\left[P\left(i \mid \mathbf{t}\right) \varphi\left(\boldsymbol{\Theta}_i, \mathbf{t}\right)\right]}{E\left[P\left(i \mid \mathbf{t}\right)\right]} \\ &= \frac{\int p\left(\mathbf{t}\right) P\left(i \mid \mathbf{t}\right) \varphi\left(\boldsymbol{\Theta}_i, \mathbf{t}\right) d\mathbf{t}}{\pi_i} \\ &= \int \frac{p\left(\mathbf{t}\right) P\left(i \mid \mathbf{t}\right)}{\pi_i} \varphi\left(\boldsymbol{\Theta}_i, \mathbf{t}\right) d\mathbf{t} \\ &= \int p\left(\mathbf{t} \mid i\right) \varphi\left(\boldsymbol{\Theta}_i, \mathbf{t}\right) d\mathbf{t} = E\left[\varphi\left(\boldsymbol{\Theta}_i, \mathbf{t}\right) \mid i\right]. \end{aligned} \tag{33}$$

Therefore, we can rewrite the mixture parameters in terms of the weighted means $< \varphi >_i$

$$\pi_i = < 1 >_i \tag{34}$$

$$\boldsymbol{\mu}_i = \frac{< \mathbf{t} >_i}{< 1 >_i} \tag{35}$$

$$\mathbf{C}_i = \frac{< \left(\mathbf{t} - \boldsymbol{\mu}_i\right) \left(\mathbf{t} - \boldsymbol{\mu}_i\right)^T >_i}{< 1 >_i}. \tag{36}$$

If we have n samples (finite case), the linear least squares approximation for $< \varphi >_i$ is

$$< \varphi >_i = \frac{1}{n} \sum_{j=1}^{n} P\left(i \mid \mathbf{t}_j\right) \varphi\left(\boldsymbol{\Theta}_i, \mathbf{t}_j\right) \approx \frac{1}{n} \sum_{j=1}^{n} R_{ji} \varphi\left(\boldsymbol{\Theta}_i, \mathbf{t}_j\right) \tag{37}$$

where we have used the approximation (4) to the posterior probability that mixture component i generated the sample $\mathbf{t}_j$.

As $n \rightarrow \infty$, the approximation of (37) is closer to the exact value given by (32). Next, we apply the Robbins–Monro stochastic approximation algorithm [10, Sec. 1.1] to estimate iteratively the value of the weighted means $< \varphi >_i$

$$< \varphi >_i (0) = R_{0i} \varphi\left(\boldsymbol{\Theta}_i, \mathbf{t}_0\right) \tag{38}$$

$$\begin{aligned} < \varphi >_i (n) = \; & < \varphi >_i (n-1) + \epsilon\left(n, i\right) \left(R_{ni} \varphi\left(\boldsymbol{\Theta}_i, \mathbf{t}_n\right)\right. \\ & \left. - < \varphi >_i (n-1)\right) \end{aligned} \tag{39}$$

where $\epsilon\left(n, i\right)$ is the variable step size. Equation (39) is more conveniently written as

$$\begin{aligned} < \varphi >_i (n) = \; & \left(1 - \epsilon\left(n, i\right)\right) < \varphi >_i (n-1) \\ & + \epsilon\left(n, i\right) R_{ni} \varphi\left(\boldsymbol{\Theta}_i, \mathbf{t}_n\right). \end{aligned} \tag{40}$$

Now we derive an online stochastic approximation algorithm by applying (40) to (34)–(36). First we need to define two auxiliary variables

$$\mathbf{m}_i = < \mathbf{t} >_i \tag{41}$$

$$\mathbf{M}_i = < \left(\mathbf{t} - \boldsymbol{\mu}_i\right) \left(\mathbf{t} - \boldsymbol{\mu}_i\right)^T >_i . \tag{42}$$

The corresponding update equations are

$$\mathbf{m}\left(n\right) = \left(1 - \epsilon\left(n, i\right)\right) \mathbf{m}_i \left(n-1\right) + \epsilon\left(n, i\right) R_{ni} \mathbf{t}_n \tag{43}$$

$$\begin{aligned} \mathbf{M}_i \left(n\right) = \; & \left(1 - \epsilon\left(n, i\right)\right) \mathbf{M}_i \left(n-1\right) \\ & + \epsilon\left(n, i\right) R_{ni} \left(\mathbf{t}_n - \boldsymbol{\mu}_i \left(n\right)\right) \left(\mathbf{t}_n - \boldsymbol{\mu}_i \left(n\right)\right)^T. \end{aligned} \tag{44}$$

Then, we are ready to rewrite (34)–(36) as

$$\pi_i \left(n\right) = \left(1 - \epsilon\left(n, i\right)\right) \pi_i \left(n-1\right) + \epsilon\left(n, i\right) R_{ni} \tag{45}$$

$$\mu_i \left(n\right) = \frac{\mathbf{m}_i \left(n\right)}{\pi_i \left(n\right)} \tag{46}$$

$$\mathbf{C}_i \left(n\right) = \frac{\mathbf{M}_i \left(n\right)}{\pi_i \left(n\right)}. \tag{47}$$

References

[1] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ: Prentice-Hall, 1988.

[2] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cybern.*, vol. 43, no. 1, pp. 59–69, 1982.

[3] A. Rauber, D. Merkl, and M. Dittenbach, "The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data," *IEEE Trans. Neural Netw.*, vol. 13, no. 6, pp. 1331–1341, Nov. 2002.

[4] E. López-Rubio, J. Muñoz-Pérez, and J. A. Gómez-Ruiz, "Self-organizing dynamic graphs," *Neural Process. Lett.*, vol. 16, no. 2, pp. 93–109, Oct. 2002.

[5] E. López-Rubio, J. M. Ortiz-de-Lazcano-Lobato, and M. C. Vargas-González, "Probabilistic self-organizing graphs," in *Proc. 10th Int. Work-Conf. Artif. Neural Netw.*, vol. 1, 2009, pp. 180–187.

[6] B. Fritzke, "Growing grid—A self-organizing network with constant neighborhood range and adaptation strength," *Neural Process. Lett.*, vol. 2, no. 5, pp. 9–13, 1995.

[7] C. Malsburg, "Network self-organization," in *An Introduction to Neural and Electronic Networks*, S. F. Zornetzer, J. L. Davis, and C. Lau, Eds. New York: Academic, 1990, ch. 22, pp. 421–432.

[8] T. Heskes, "Self-organizing maps, vector quantization, and mixture modeling," *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1299–1305, Nov. 2001.

[9] M. Sato and S. Ishii, "On-line EM algorithm for the normalized Gaussian network," *Neural Comput.*, vol. 12, no. 2, pp. 407–432, Feb. 2000.

[10] H. J. Kushner and G. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*. New York: Springer-Verlag, 2003.

[11] S.-S. Cheng, H.-C. Fu, and H.-M. Wang, "Model-based clustering by probabilistic self-organizing maps," *IEEE Trans. Neural Netw.*, vol. 20, no. 5, pp. 805–826, May 2009.

[12] D. Desieno, "Adding a conscience to competitive learning," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 1. San Diego, CA, Jul. 1988, pp. 117–124.

[13] D. Bacciu and A. Starita, "Competitive repetition suppression (CoRe) clustering: A biologically inspired learning model with application to robust clustering," *IEEE Trans. Neural Netw.*, vol. 19, no. 11, pp. 1922–1941, Nov. 2008.
[14] R. Rizzo and A. Chella, "A comparison between habituation and conscience mechanism in self-organizing maps," *IEEE Trans. Neural Netw.*, vol. 17, no. 3, pp. 807–810, May 2006.
[15] D. Borland and R. M. Taylor, "Rainbow color map (still) considered harmful," *IEEE Comput. Graph. Appl.*, vol. 27, no. 2, pp. 14–17, Mar.–Apr. 2007.
[16] E. R. Gansner and S. C. North, "An open graph visualization system and its applications to software engineering," *Softw.–Pract. Exper.*, vol. 30, no. 11, pp. 1203–1233, 1999.
[17] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990.
[18] T. Kohonen, "Emergence of invariant-feature detectors in the adaptive-subspace self-organizing map," *Biol. Cybern.*, vol. 75, no. 4, pp. 281–291, 1996.
[19] J. J. Verbeek, N. Vlassis, and B. J. A. Kröse, "Self-organizing mixture models," *Neurocomputing*, vol. 63, pp. 99–123, Jan. 2005.
[20] M. M. Van Hulle, "Maximum likelihood topographic map formation," *Neural Comput.*, vol. 17, no. 3, pp. 503–513, Mar. 2005.
[21] M. M. Van Hulle, "Joint entropy maximization in kernel-based topographic maps," *Neural Comput.*, vol. 14, no. 8, pp. 1887–1906, Aug. 2002.
[22] E. López-Rubio, "Probabilistic self-organizing maps for continuous data," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1543–1554, Oct. 2010.
[23] T. Wittman. (2005). *MANIfold Learning MATLAB Demo*. Dept. Math., Univ. California, Los Angeles [Online]. Available: http://www.math.ucla.edu/∼wittman/mani/index.html
[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[25] M. Reisslein, L. J. Karam, P. Seeling, F. H. P. Fitzek, and T. K. Madsen. (2010, Dec.). *YUV Video Sequences* [Online]. Available: http://trace.eas.asu.edu/yuv/index.html
[26] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognit. Neurosci.*, vol. 3, no. 1, pp. 71–86, 1991.
[27] A. Sharma and K. K. Paliwal, "Fast principal component analysis using fixed-point algorithm," *Pattern Recognit. Lett.*, vol. 28, no. 10, pp. 1151–1155, Jul. 2007.
[28] A. Asuncion and D. J. Newman. (2007). *UCI Machine Learning Repository*. School Inf. Comput. Sci., Univ. California, Irvine [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html
[29] V. Moschou, D. Ververidis, and C. Kotropoulos, "Assessment of self-organizing map variants for clustering with application to redistribution of emotional speech patterns," *Neurocomputing*, vol. 71, nos. 1–3, pp. 147–156, Dec. 2007.
[30] A. Vellido, "Missing data imputation through GTM as a mixture of t-distributions," *Neural Netw.*, vol. 19, no. 10, pp. 1624–1635, Dec. 2006.
[31] M. P. Sinka and D. W. Corne, "The BankSearch web document dataset: Investigating unsupervised clustering and category similarity," *J. Netw. Comput. Appl.*, vol. 28, no. 2, pp. 129–146, Apr. 2005.
[32] C. J. Van Rijsbergen, *Information Retrieval*, 2nd ed. Oxford, U.K.: Butterworths, 1979.
[33] C. M. Bishop and M. Svensén, "The generative topographic mapping," *Neural Comput.*, vol. 10, no. 1, pp. 215–234, 1998.
[34] G. Zhai, W. Zhang, X. Yang, W. Lin, and Y. Xu, "Efficient deblocking with coefficient regularization, shape-adaptive filtering, and quantization constraint," *IEEE Trans. Multimedia*, vol. 10, no. 5, pp. 735–745, Aug. 2008.
[35] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images," *IEEE Trans. Image Process.*, vol. 16, no. 5, pp. 1395–1411, May 2007.
[36] D. Sun and W.-K. Cham, "Postprocessing of low bit-rate block DCT coded images based on a fields of experts prior," *IEEE Trans. Image Process.*, vol. 16, no. 11, pp. 2743–2751, Nov. 2007.
[37] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Trans. Image Process.*, vol. 16, no. 2, pp. 349–366, Feb. 2007.
[38] E. López-Rubio, "Restoration of images corrupted by Gaussian and uniform impulsive noise," *Pattern Recognit.*, vol. 43, no. 5, pp. 1835–1846, May 2010.
[39] A. Weber. (2010). *The USC-SIPI Image Database* [Online]. Avilable: http://sipi.usc.edu/database/
[40] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Proccess.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[41] E. López-Rubio, J. M. Ortiz-de-Lazcano-Lobato, and D. López-Rodríguez, "Probabilistic PCA self-organizing maps," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1474–1489, Sep. 2009.
[42] E. López-Rubio, "Multivariate Student-t self-organizing maps," *Neural Netw.*, vol. 22, no. 10, pp. 1432–1447, Dec. 2009.

Ezequiel López-Rubio was born in 1976. He received the M.Sc. and Ph.D. degrees (with honors) in computer engineering from the University of Málaga, Málaga, Spain, in 1999 and 2002, respectively.

He joined the Department of Computer Languages and Computer Science, University of Málaga, in 2000, where he is currently an Associate Professor of computer science and artificial intelligence. His current research interests include unsupervised learning, pattern recognition, and image processing.

Esteban José Palomo was born in 1982. He received the B.Sc. and M.Sc. degrees in computer engineering from the University of Málaga, Málaga, Spain, in 2006, and 2008, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Languages and Computer Science, University of Málaga.

His current research interests include neural networks, self-organization, data mining, image/video processing, and network security.