Probabilistic PCA Self-Organizing Maps

Ezequiel López-Rubio, Juan Miguel Ortiz-de-Lazcano-Lobato, and Domingo López-Rodríguez

***Abstract*—In this paper, we present a probabilistic neural model, which extends Kohonen's self-organizing map (SOM) by performing a probabilistic principal component analysis (PPCA) at each neuron. Several SOMs have been proposed in the literature to capture the local principal subspaces, but our approach offers a probabilistic model while it has a low complexity on the dimensionality of the input space. This allows to process very high-dimensional data to obtain reliable estimations of the probability densities which are based on the PPCA framework. Experimental results are presented, which show the map formation capabilities of the proposal with high-dimensional data, and its potential in image and video compression applications.**

***Index Terms*—Competitive learning, dimensionality reduction, handwritten digit recognition, probabilistic principal component analysis (PPCA), self-organizing maps (SOMs), unsupervised learning.**

I. Introduction

THE concept of self-organization seems to explain several neural structures of the brain that perform invariant feature detection [15]. These structures inspired the proposal of computational maps designed to explore multidimensional data. The original self-organizing map (SOM) was proposed by Kohonen [22], where each neuron had a weight vector to represent a point of the input space. It was followed by the adaptive subspace self-organizing map (ASSOM), which was first presented as an invariant feature detector [23]. This property has been further studied [24], and its relations with wavelets and Gabor filters have been reported [25], [36], [37].

Each neuron of an ASSOM network represents a subset of the input data with a vector basis. This vector basis is adapted so that the data points of the subset are as close as possible to its spanned vector subspace. Hence, a description of the local geometry of the input data is built. The concept of a neuron which represents a linear subspace can be traced to the subspace classifier by Oja [33]. The minimization of the mean squared error (MSE) of the projection errors on the subspaces leads naturally to the principal components analysis (PCA). The ASSOM model extends these ideas by considering the self-organization of subspace neurons, which can also be found in Dony and Haykin's optimally adaptive transform coding [13].

The ASSOM has been successfully applied to the handwritten digit recognition problem [65] which many neural network researchers have addressed [47]. Also, it has been used for texture segmentation [43]. This work is related with a supervised variant of the ASSOM, called supervised adaptive-subspace self-organizing map (SASSOM), first proposed by Ruiz del Solar and Köppen [42].

Finally, SOM networks are adequate to create topographic maps, which are representations of the input space. This ability is inherited by the ASSOM network, which has been taken as a standard for comparison with other algorithms that make these maps [55].

The ASSOM does not define any probability model on the input data. This is not the case for the generative topographic mapping (GTM) by Bishop *et al.* [8], which is a constrained mixture of Gaussians. A *latent space* is defined with a reduced dimensionality, and a lattice of units is set up in the latent space.

The ASSOM lacks the ability to learn the local mean vectors. The PCA SOM [30] solves this problem by learning both the mean vector and the covariance matrix at each neuron. The self-organizing mixture networks (SOMNs), by Yin and Allinson [64], also follow this line. Unfortunately, the use of the full covariance matrix makes them computationally heavy for high-dimensional data sets.

The self-organizing mixture model (SOMM) by Verbeek *et al.* [59] uses a version of the expectation–maximization (EM) method to produce an extension of the SOM where a mixture of restricted Gaussians is defined. Nevertheless, it has some scalability problems when the size of the map grows.

Other families of SOMs include kernel-based topographic maps [56]–[58], where Gaussian kernels are defined around a centroid, and topographic independent component analysis (ICA) [19], which introduces the use of ICA instead of PCA.

Our aim here is to develop a self-organizing model with online learning of the local subspaces of an input distribution, which is based on the probabilistic PCA (PPCA) framework. Furthermore, our proposal has a low complexity both in the size of the map and in the input space dimension, so that it is suited for high-dimensional data. This sort of data sets is common in certain typical applications of SOMs such as exploratory data analysis [12] and content-based data retrieval [27], [28].

The outline of this paper is as follows. Section II explores the links between subspace methods, density modeling, and self-organization. In Section III, we present the enhanced model, called probabilistic principal component analysis self-organizing map (PPCASOM). Section IV is devoted to complexity analysis. A discussion of the differences among known models and our proposal is carried out in Section V. Finally, computational results are shown in Section VI.

Manuscript received September 05, 2007; revised July 08, 2008, November 12, 2008, April 06, 2009, and May 17, 2009; accepted June 09, 2009. First published August 18, 2009; current version published September 02, 2009. This work was supported in part by the Ministry of Education and Science of Spain under Projects TIC2003-03067 and TIN2006-07362, and by the Autonomous Government of Andalusia (Spain) under Projects P06-TIC-01615 and P07-TIC-02800.

E. López-Rubio and J. M. Ortiz-de-Lazcano-Lobato are with the Department of Computer Languages and Computer Science, University of Málaga, 29071 Málaga, Spain (e-mails: ezeqlr@lcc.uma.es; jmortiz@lcc.uma.es).

D. López-Rodríguez is with the Department of Applied Mathematics, University of Málaga, 29071 Málaga, Spain (e-mail: dlopez@ctima.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.2009.2025888

II. Subspace Density Modeling and Self-Organization

As we have seen, high-dimensional spaces arise naturally in many pattern recognition and machine learning areas. Conventional PCA is defined as an orthogonal linear transformation where the most variance of the original data comes to lie in the first coordinates of the transformed data. The transformation is given by $\mathbf{z} = \mathbf{U}^T(\mathbf{t} - \boldsymbol{\mu})$, where $\mathbf{t}$ is the original data, and $\mathbf{z}$ is the transformed data, both of dimension d. As known, $\boldsymbol{\mu}$ is the mean vector, and $\mathbf{U}$ is the $d \times d$ matrix of eigenvectors of the data covariance matrix $\mathbf{C}$, which is decomposed as $\mathbf{C} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$. The eigenvalues in the diagonal matrix $\boldsymbol{\Lambda}$ are sorted in decreasing order. Hence, conventional PCA shows that the optimal linear dimensionality reduction to q dimensions, with $q < d$, involves an orthogonal projection on the principal directions given by the eigenvectors corresponding to the q largest eigenvalues of $\mathbf{C}$: $\mathbf{z}_q = \mathbf{U}_q^T(\mathbf{t} - \boldsymbol{\mu})$. The optimality is in the least squares sense, and the optimal reconstruction error is $E_r = \|\mathbf{t} - \boldsymbol{\mu} - \mathbf{z}_q\mathbf{U}_q\|^2$. This starting point has given rise to many dimensionality reduction techniques, in particular, PPCA and subspace methods [61].

PPCA [53] models the probability distribution of the input data by a multivariate Gaussian. This extends conventional PCA, which does not define any probability model (two examples of non-Gaussian PCA for mixture modeling can be found in [66] and [39]). In order to achieve dimensionality reduction, PPCA considers that the input data come from a linear transformation of a q-dimensional vector of latent variables. This yields a probability density model which corresponds to the q-dimensional principal subspace of the data, with the addition of isotropic noise in all directions. The optimal dimensionality reduction can be obtained by projection of the original data on the principal subspace, just as in conventional PCA. Hence, the trailing $d - q$ directions are considered noisy and they are not even estimated.

On the other hand, the subspace method by Moghaddam and Pentland [35] splits the input space into two orthogonal subspaces: the principal subspace of dimension q and its complementary subspace of dimension $d - q$. The probability density for the original input space is built by the product of two independent Gaussians, one for each subspace. The original data are projected on the two subspaces, but the projection on the complementary subspace is not explicitly computed.

Hence, we obtain a procedure which has similarities to PPCA, and in fact has been proven to be equivalent [61]. The two approaches need to estimate the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\mathbf{C}$. The main difficulty is the robust estimation of $\mathbf{C}$, since a plain maximum-likelihood estimator will fail to produce a full rank $\mathbf{C}$ because of the small sample size with respect to the high dimensionality of the input space. The solution is to restrict $\mathbf{C}$ so that it has less degrees of freedom, in order to obtain robust estimations. The subspace method projects the original data into the eigenspace of $\mathbf{C}$, and then uses only the q leading directions. The remaining $d - q$ projections are estimated by a parameter which can be shown to correspond to PPCA's isotropic noise parameter. Therefore, the two methods are seen as different ways to implement a robust estimator of $\mathbf{C}$ for high-dimensional data with comparatively small sample sizes.

As we see, the search for linear subspaces where the data lie is guided in a probabilistic framework by the PCA principal directions associated with the q leading eigenvalues of the covariance matrix of a Gaussian distribution. These models are easily extended to mixtures of Gaussians. Then, we have H linear transformations, one per mixture component. In principle, the mixture components are not bound to each other, and they are adjusted to optimize some objective function, such as the log-likelihood of the input data. Nevertheless, it is more convenient in data visualization applications to enforce the self-organization of the subspaces. This yields faithful representations of the input distribution, which can be used to explore the structure of complex high-dimensional data sets. Under the robust estimation perspective considered before, self-organization of linear subspaces can be regarded as a way to impose additional conditions in the estimated covariance matrices, since every covariance matrix is constrained to be similar to those of the neighboring units. This procedure effectively reduces the variability of the covariance matrices, which leads to increased robustness against the relative lack of input samples for large d. Furthermore, the number of input samples which contribute significantly to the estimation of a particular covariance matrix is increased, as the information is shared among neighboring units, and this leads to more robustness.

Other approaches to restrict Gaussian mixture models include enforcing the covariance matrices to lie in a low-dimensional matrix subspace, as shown in [11]. These restrictions alleviate the data insufficiency problem, which prevents reliable estimation of full covariance mixture models.

Perhaps the best known SOM model which learns linear subspaces is the ASSOM. It does not define a probability density model, but its objective function is the average expected projection error. So, it is closely related to the reconstruction error of PCA, since each ASSOM neuron stores a q-dimensional orthonormal vector base which could be identified with $\mathbf{U}_q$. The dimensionality reduction capability of these maps is somehow reduced by the absence of a mean vector in the neurons, which is equivalent to assume $\boldsymbol{\mu} = 0$.

Other self-organizing models which do not learn subspaces, such as the SOMM and kernel-based topographic maps, assume Gaussian densities. Hence, we can think of them as models where the number of retained principal directions is $q = 0$. In fact, they enforce $\mathbf{U} = \mathbf{I}$, and no PCA transformation is performed. This implies that their covariance matrices are diagonal, and they only learn the variances in each direction given by the eigenvalue matrix $\boldsymbol{\Lambda}$. We can conclude that many SOM proposals include certain elements of subspace-based probability density modeling, but not completely. Some of them do not define probability densities, and others do not learn subspaces.

It should not be inferred from the discussion above that subspace modeling is only suitable for the minimization of the reconstruction error. In supervised classification applications, the relevant features for discrimination are commonly confined to low-dimensional subspaces of the input space. In this context, the objective is to capture the most relevant directions for classification. Aladjem [1] and Calà [9] have proposed methods to model class-conditional probability densities with Gaussian mixtures by projection pursuit. A related approach to supervised

classification is given by Saito *et al.* [9], who search the most discriminant subspaces by estimating the difference among the probability distributions of the projected data belonging to each class. In this latter case, the probability densities are not assumed to be Gaussian. These methods are not so closely related with SOMs, which are unsupervised systems in most cases. Nevertheless, unsupervised projection pursuit methods are also available for feature selection [20]. In these approaches, it is desired that the probability density of the projected data is as non-Gaussian as possible.

III. The PPCASOM Model

A. Mixture Model

Each neuron of the map stores a PPCA model [53] to perform a dimensionality reduction from the observed (input) space dimension d to the latent (reduced) subspace dimension q, with $q < d$. The observed data $\mathbf{t}$ depend linearly on the latent variables in $\mathbf{x}$, with a mean vector $\boldsymbol{\mu}$ and a noise model ξ

$$\mathbf{t} = \mathbf{W}\mathbf{x} + \boldsymbol{\mu} + \xi. \tag{1}$$

The latent variables are defined to be independent and Gaussian with unit variance and zero mean, i.e., $\mathbf{x} \sim (N(0, \mathbf{I})$. The noise model is also Gaussian such that $\xi \sim N(0, \sigma^2\mathbf{I})$, and the $d \times q$ parameter matrix $\mathbf{W}$ contains the factor loadings. This formulation implies that the observation vectors are also normally distributed, $\mathbf{t} \sim N(\boldsymbol{\mu}, \mathbf{C})$, with a covariance matrix $\mathbf{C} = \sigma^2\mathbf{I} + \mathbf{W}\mathbf{W}^T$.

The PPCASOM model is defined as a mixture of H PPCA components, with prior probabilities or mixing proportions π_i

$$p(\mathbf{t}_n) = \sum_{i=1}^{H} \pi_i p(\mathbf{t}_n|i) \tag{2}$$

where $p(\mathbf{t}_n|i)$ is the PPCA probability density associated with mixture component i

$$p(\mathbf{t}_n|i) = (2\pi)^{-d/2}|\mathbf{C}_i|^{-1/2}\exp\left(-E_{ni}^2/2\right) \tag{3}$$

$$E_{ni}^2 = (\mathbf{t}_n - \boldsymbol{\mu}_i)\mathbf{C}_i^{-1}(\mathbf{t}_n - \boldsymbol{\mu}_i)^T. \tag{4}$$

Now we need a procedure to compute $p(\mathbf{t}_n|i)$ in $O(d)$. From PPCA, we know that the parameter matrix $\mathbf{W}_i$ can be decomposed as

$$\mathbf{W}_i = \mathbf{U}_{qi}\left(\mathbf{K}_{qi} - \sigma_i^2\mathbf{I}\right)^{1/2}\mathbf{R}_i \tag{5}$$

where the columns of the $d \times q$ matrix $\mathbf{U}_{qi}$ are the eigenvectors corresponding to the q principal directions of the subspace of neuron i, $\mathbf{K}_{qi}$ is a $q \times q$ diagonal matrix with the corresponding eigenvalues, and $\mathbf{R}_i$ is a rotation matrix which may be computed as the matrix of eigenvectors of the $q \times q$ matrix $\mathbf{W}_i^T\mathbf{W}_i$. Then, the decomposition of $\mathbf{W}_i$ is completed by normalization of the columns of $\mathbf{W}_i\mathbf{R}_i^{-1}$.

The error term E_{ni} can be expressed in terms of this decomposition (see [53] for details)

$$E_{ni}^2 = \mathbf{z}_{in}^T\mathbf{K}_{qi}^{-1}\mathbf{z}_{in} + E_{ri}^2/\sigma_i^2 \tag{6}$$

where $\mathbf{z}_{in}$ is the projection of $\mathbf{t}_n - \boldsymbol{\mu}_i$ onto the principal subspace of neuron i and E_{ri} is the reconstruction error corresponding to the reconstruction vector $\hat{\mathbf{t}}_n^i$

$$\mathbf{z}_{in} = \mathbf{U}_{qi}^T(\mathbf{t}_n - \boldsymbol{\mu}_i) \tag{7}$$

$$E_{ri}^2 = \left\|\mathbf{t}_n - \hat{\mathbf{t}}_n^i\right\|^2 \tag{8}$$

$$\hat{\mathbf{t}}_n^i = \mathbf{W}_i\left(\mathbf{W}_i^T\mathbf{W}_i\right)^{-1}\mathbf{W}_i^T(\mathbf{t}_n - \boldsymbol{\mu}_i). \tag{9}$$

Finally, the determinant of $\mathbf{C}_i$ can be computed as

$$|\mathbf{C}_i| = \left(\prod_{j=1}^{q} k_j\right)\left(\sigma_i^2\right)^{d-q} \tag{10}$$

where the k_j are the q first eigenvalues of $\mathbf{C}_i$, which are stored in $\mathbf{K}_{qi}$.

B. Self-Organization

At each time step n, the network is presented a data sample $\mathbf{t}_n$. We introduce a discrete hidden variable z_n whose value (from 1 to H, where H is the number of mixture components) indicates which component generated the data sample $\mathbf{t}_n$. In order to achieve self-organization, we only allow distribution models for z_n which have the property that one component is the most probable and the probability decreases with the *topological distance* to that component (see [32] and [59]). A *topology* is defined in the network so that the topological distance between mixture components i and i' is called $\delta(i, i')$. A flat rectangular lattice may be used

$$\delta(i, i') = \left\|(i_x, i_y) - \left(i'_x, i'_y\right)\right\| \tag{11}$$

where (i_x, i_y) is the coordinate vector of mixture component i in the lattice. Other lattice topologies and/or geometries could be also considered, such as hexagonal lattice topologies and toroidal lattice geometries.

Hence, we consider the following set of distributions $Q_n = \{q_{n1}, \ldots, q_{nH}\}$ for z_n:

$$q_{nj}(z_n{=}i) = P_{nj}(z_n{=}i) = P(z_n{=}i|q_{nj}) \propto \exp\left(-\frac{\delta(i,j)}{\Delta(n)}\right) \tag{12}$$

where $\sum_{i=1}^{H} q_{nj}(z_n = i) = 1$. Note that δ is the topological distance function and $\Delta(n)$ is the neighborhood width, which is a positive decreasing function of n such that $\Delta(n) \to 0$ as $n \to N$, where N is the total number of time steps. A standard choice is the linear decay

$$\Delta(n) = \Delta(0)\left(1 - \frac{n}{N}\right). \tag{13}$$

In the experiments, we have used an initial value $\Delta(0) = (\text{rows}+\text{cols})/2$ for $\text{rows} \times \text{cols}$ topologies. This ensures that the neighborhoods are large at the beginning, when the map needs to be plastic.

For brevity, we note

$$q_{nji} = q_{nj}(z_n = i) = P_{nj}(z_n = i) = P(z_n = i|q_{nj}). \tag{14}$$

For each time step n and distribution q_{nj}, we have

$$p(\mathbf{t}_n|q_{nj}) = \sum_{i=1}^{H} q_{nji} p(\mathbf{t}_n|i). \tag{15}$$

Then, a maximum-likelihood approach is used to decide which distribution from Q_n has the most probability $p(q_{nj}|\mathbf{t}_n)$ of having generated the sample $\mathbf{t}_n$. This distribution is called the *winning distribution*. The prior probabilities $P(q_{nj})$ are assumed equal, i.e., $P(q_{nj}) = 1/H \forall j$, and so we have

$$\begin{aligned} \text{Winner}(n) &= \arg\max_j \{p(q_{nj}|\mathbf{t}_n)\} \\ &= \arg\max_j \left\{ \frac{P(q_{nj}) p(\mathbf{t}_n|q_{nj})}{\sum_{k=1}^{H} P(q_{nk}) p(\mathbf{t}_n|q_{nk})} \right\} \\ &= \arg\max_j \{p(\mathbf{t}_n|q_{nj})\}. \end{aligned} \tag{16}$$

This selection of a distribution from the set of distributions Q_n enforces the self-organization of the network. When the winning distribution q_{nh} has been selected, we may compute the posterior responsibility R_{ni} of mixture component i for generating the observed sample $\mathbf{t}_n$, given the set of possible distributions Q_n

$$R_{ni} = P(i|\mathbf{t}_n, Q_n) = P(i|q_{nh}) = q_{nhi}, \quad \text{where } h = \text{Winner}(n). \tag{17}$$

After this competition, the mixture components are updated according to R_{ni} and $\mathbf{t}_n$ in order to build a SOM.

C. Mixture Component Update

When a new sample $\mathbf{t}_n$ has been presented to the network, 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 version of the original EM method of the PPCA model is required. This possibility has been examined by Sato and Ishii [45] for general PPCA mixtures. Our online EM generates the updated values $\sigma_i(n)$ and $\mathbf{W}_i(n)$ from the old values $i(n-1), Wi(n-1)$ and the new sample $\mathbf{t}_n$. The application of Robbins–Monro stochastic approximation algorithm yields the following update equations (see the Appendix for details):

$$\pi_i(n) = (1 - \varepsilon(n))\, \pi_i(n-1) + \varepsilon(n) R_{ni} \tag{18}$$

$$\mathbf{m}_i(n) = (1 - \varepsilon(n))\, \mathbf{m}_i(n-1) + \varepsilon(n) R_{ni} \mathbf{t}_n \tag{19}$$

$$\boldsymbol{\mu}_i(n) = \frac{\mathbf{m}_i(n)}{\pi_i(n)} \tag{20}$$

$$\mathbf{M}_i = \sigma_i^2 \mathbf{I} + \mathbf{W}_i^T \mathbf{W}_i \tag{21}$$

$$E\left[\mathbf{x}_j^T|i\right] = (\mathbf{t}_j - \boldsymbol{\mu}_i)^T \mathbf{W}_i \mathbf{M}_i^{-1} \tag{22}$$

$$E\left[\mathbf{x}_j \mathbf{x}_j^T|i\right] = \sigma_i^2 \mathbf{M}_i^{-1} + \left(E\left[\mathbf{x}_j^T|i\right]\right)^T E\left[\mathbf{x}_j^T|i\right] \tag{23}$$

$$\boldsymbol{\Omega}_i(n) = (1-\varepsilon(n))\, \boldsymbol{\Omega}_i(n-1) + \varepsilon(n) R_{ni} (\mathbf{t}_n - \boldsymbol{\mu}_i) E\left[\mathbf{x}_n^T|i\right] \tag{24}$$

$$\boldsymbol{\Xi}_i(n) = (1-\varepsilon(n))\, \boldsymbol{\Xi}_i(n-1) + \varepsilon(n) R_{ni} E\left[\mathbf{x}_n \mathbf{x}_n^T|i\right] \tag{25}$$

$$\mathbf{W}_i(n) = \boldsymbol{\Omega}_i(n) \left(\boldsymbol{\Xi}_i(n)\right)^{-1} \tag{26}$$

$$\begin{aligned} \sigma_i^2(n) = {} & (1 - \varepsilon(n))\, \sigma_i^2(n-1) + \varepsilon(n) R_{ni} \frac{1}{d} \\ & \times \left(\|\mathbf{t}_n - \boldsymbol{\mu}_i\|^2 - 2E\left[\mathbf{x}_n^T|i\right] \mathbf{W}_i(n)^T (\mathbf{t}_n - \boldsymbol{\mu}_i) \right. \\ & \left. + \operatorname{tr}\left(E\left[\mathbf{x}_n \mathbf{x}_n^T|i\right] \mathbf{W}_i(n)^T \mathbf{W}_i(n)\right) \right) \end{aligned} \tag{27}$$

where $\pi_i(n)$ are the mixing proportions, $\mathbf{m}_i(n)$ are not normalized versions of the mean vectors, $\boldsymbol{\mu}_i(n)$ are the mean vectors, $\boldsymbol{\Omega}_i(n)$ and $\boldsymbol{\Xi}_i(n)$ are matrices which are used to build the PPCA matrices $\mathbf{W}_i(n)$, and $\sigma_i^2(n)$ are the noise variances. Note that $\mathbf{W}_i(n)$ cannot be estimated directly by stochastic approximation because it is not obtained in PPCA as an accumulated sum. This is the reason for introducing $\boldsymbol{\Omega}_i(n)$ and $\boldsymbol{\Xi}_i(n)$. Finally, $\varepsilon(n)$ is the step size of the Robbins–Monro algorithm and is typically chosen as

$$\varepsilon(n) = \frac{1}{an + b}, \qquad 0 < a < 1. \tag{28}$$

In our experiments, we have found that selecting $b = 1/a$ often yields good results because the network remains plastic long enough. So, we have used in practice

$$\varepsilon(n) = \frac{1}{\varepsilon_0 n + \dfrac{1}{\varepsilon_0}}, \qquad 0 < \varepsilon_0 < 1 \tag{29}$$

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

Please note that (21)–(27) are coupled because they implement the EM iteration. We see in (18)–(27) that the rate at which the new information is incorporated to the model is controlled by the step size $\varepsilon(n)$ and the posterior responsibility R_{ni}.

D. Network Initialization

The initialization of the network follows the standard PPCA initialization outlined in [52]. For each neuron i, we select K samples $\mathbf{t}_n$, and we compute their mean as $\boldsymbol{\mu}_i(0)$

$$\boldsymbol{\mu}_i(0) = \frac{1}{K} \sum_{n=1}^{K} \mathbf{t}_n. \tag{30}$$

The value of K is not crucial, provided that it is higher than q in order to avoid degenerate vector bases. We have used $K = 10q$ in the experiments. Then, we choose q of these samples and compute their differences with $\boldsymbol{\mu}_i(0)$ to yield the columns of a matrix $\mathbf{V}_i$ of size $d \times q$. After that we enter an iterative procedure where each column $\mathbf{v}_i$ of $\mathbf{V}_i$ is multiplied by $\mathbf{S}_i(0)$, which does not need to be computed explicitly

$$\mathbf{S}_i(0)\mathbf{v}_i = \sum_{n=1}^{K} (\mathbf{t}_n - \boldsymbol{\mu}_i(0)) \left[(\mathbf{t}_n - \boldsymbol{\mu}_i(0))^T \mathbf{v}_i \right]. \tag{31}$$

Please note that the above equation can be evaluated in $O(Kd)$ steps by following the indicated order of evaluation of the matrix products. After the multiplication, the obtained vectors are orthonormalized to yield the new value of $\mathbf{V}_i$. The orthonormalization can be accomplished by several methods; in our implementation, we have done it by singular value decomposition (SVD). After a few steps, this procedure converges.

The final eigenvectors are stored into the $d \times q$ matrix $\mathbf{U}_i$ and the corresponding eigenvalues go to the $q \times q$ diagonal matrix $\mathbf{\Lambda}_i$. Then, we use the following formula for $\sigma_i(0)$:

$$\sigma_{\mathrm{i}}(0) = \mathrm{trace}\left(\mathbf{S}_{\mathrm{i}}(0)\right) - \sum_{j=1}^{q} \lambda_j \tag{32}$$

where the trace of $\mathbf{S}_i(0)$ can be computed in $O(Kd)$ steps. The starting value of $\mathbf{W}_i$ is obtained as

$$\mathbf{W}_i(0) = \mathbf{U}_i(0)\left(\mathbf{\Lambda}_i(0) - (\sigma_i(0))^2\mathbf{I}\right)^{1/2}. \tag{33}$$

Now we have initial estimations for the parameters of the original PPCA scheme. Finally, the remaining parameters can be computed following (A.10) and (A.19)-(A.21)

$$\pi_i(0) = \frac{1}{H} \tag{34}$$

$$\mathbf{m}_i(0) = \frac{1}{HK}\sum_{n=1}^{K}\mathbf{t}_n = \frac{1}{H}\boldsymbol{\mu}_i(0) \tag{35}$$

$$\mathbf{\Omega}_i(0) = \sum_{j=1}^{K}\left(\mathbf{t}_j - \boldsymbol{\mu}_i(0)\right)E\left[\mathbf{x}_j^T|i\right]_i \tag{36}$$

$$\mathbf{\Xi}_i(0) = \sum_{j=1}^{K}E\left[\mathbf{x}_j\mathbf{x}_j^T|i\right]. \tag{37}$$

E. Summary

The PPCASOM model can be summarized as follows:

1) Set the initial values $\boldsymbol{\mu}_i(0)$, $\sigma_i(0)$, $\boldsymbol{W}_i(0)$, $\pi_i(0)$, $\mathbf{m}_i(0)$, $\mathbf{\Omega}_i(0)$, and $\mathbf{\Xi}_i(0)$ for all mixture components i with (30) and (32)–(37), respectively.
2) Choose an input sample $\mathbf{t}_n$ and compute the winning distribution by (16). Use (17) to compute the posterior responsibilities R_{ni}.
3) For every component i, estimate its parameters $\pi_i(n)$, $\mathbf{m}_i(n)$, and $\boldsymbol{\mu}_i(n)$ by (18)–(20).
4) For every component i, run the EM iteration by repeated application of (21)–(27) until the EM method converges, i.e., until no significant changes are made. The values obtained at convergence are the updated values $\sigma_i(n)$, $\mathbf{W}_i(n)$, $\mathbf{\Omega}_i(n)$, and $\mathbf{\Xi}_i(n)$.
5) If the map has converged or the maximum time step N has been reached, stop. Otherwise, go to step 2.

IV. Complexity Analysis

Here we analyze the computational complexity of the PPCASOM model. The initialization requires $O(Kd)$ steps per reduced dimension and neuron, as stated in Section III-D, where K is a constant independent of the problem size. Hence, if the network has H neurons, the initialization of the SOM needs $O(qdH)$ steps.

Each presentation of an input sample to the network leads to a competition, which is implemented with $O(q^2dH)$ steps. Then, an EM iteration is started in each mixture component, where (21) and (22) are the most computationally expensive, with $O(q^2d)$ steps. Hence, if we regard the number of EM iterations as a constant, each input sample is processed by the map in $O(q^2dH)$ steps.

We can conclude that the PPCASOM has linear complexity in the number of neurons H. It is also linear in the size of the input space dimension d when the latent space dimension q does not grow with d. Anyway the complexity in d is lower than cubic (note that $q < d$), particularly in typical dimensionality reduction applications with $q \ll d$. This allows to process high-dimensional data sets, as we will see in the computational experiments.

V. Discussion

There are some self-organizing models in the literature with the capability of learning local linear subspaces. Now we compare them to the PPCASOM model.

1) The ASSOM model [23] is regarded as a classic example of this kind of SOMs. It has been studied and applied extensively, as seen in the Introduction. Nevertheless, it has serious disadvantages: first, it does not consider a mean vector; and second, the update equations need stabilization against "spurious" components in the basis vectors. Furthermore, it does not define any local probability distribution, so the subspaces are "crisp," with no reference to input noise. On the other hand, the PPCASOM learns the mean vectors, it does not need to artificially remove some components of the basis vectors, and it defines local PPCA probability models. In spite of the lack of these features, the ASSOM model does not offer any advantage in computational complexity, since it processes each training sample in $O(q^2dH)$ steps. It is most useful in applications which need to map directions instead of data points.
2) The generative topographic mapping (GTM) by Bishop *et al.* [8] is a constrained mixture of Gaussians where the model parameters are determined by an EM algorithm. The main differences with the PPCASOM are that GTM works in batch mode and that GTM uses a single latent space where all the local models lie, while PPCASOM builds a local latent space per neuron. Furthermore, PPCASOM offers the possibility of having a topology with different dimensionality than the latent space dimensionality q, and even closed topologies (ring, toroidal, etc.), while the GTM is unable to do so. All this allows the PPCASOM to adapt to the input distribution with more flexibility. As before, these disadvantages of GTM are not compensated by a lower complexity, since GTM learns its single global latent space with complexity $O(q^2d)$, and PPCASOM learns H local latent spaces in $O(q^2dH)$. GTM is expected to be preferred if the task at hand needs a unique latent space, while retaining the unsupervised learning capability of SOMs.
3) The SOMM by Verbeek *et al.* [59] uses a modified version of the EM algorithm to achieve self-organization of Gaussian models with isotropic covariance matrices. The drawbacks of the SOMM are that it only works in batch mode, it is only developed for isotropic covariance matrices, and it has a heavy computational load because it is quadratic in the number of neurons: $O(dH^2)$ for SOMM versus PPCASOM's $O(dH)$. There is a $O(dH)$ speedup of

SOMM, but then it nearly reduces to a classical batch SOM with an isotropic covariance matrix. In this setup, SOMM can be seen as an improvement over SOM which features a parsimonious probabilistic model, since it does not store principal directions.

4) The kernel topographic maps by Van Hulle [56]–[58] define probability distributions, but they are rather constrained because only diagonal covariance matrices are considered. This is a problem if the local subspaces are not aligned with the input space coordinates, and the situation is worse as the input space dimension grows. On the other hand, PPCASOM does not have those constraints, so its capability to represent complex input distributions is higher. Furthermore, the version with different covariances for each axis [58] has a complexity of $\mathrm{O}(d^3)$ per step, which prevents its use with high-dimensional data. Hence, the best suited version for high-dimensional data is that with isotropic Gaussians, which we compare with PPCASOM in the experiments.
5) The PCA SOM [30] and the SOMN model [64] both learn the mean vector and the full covariance matrix at each neuron. Hence, they are more flexible for the representation of complex input data than some of the previous models, which either do not learn the mean vector (ASSOM) or only consider diagonal covariance matrices (SOMM, topographic maps). Nevertheless, their use of the full covariance matrix makes them run in $\mathrm{O}(d^3)$ complexity. Therefore, they are not as suitable as the PPCASOM for high-dimensional data.

Finally, it should be noticed that we could have presented the PPCASOM without the use of stochastic approximation, as considered in [31]. Despite the fact that the computational results are similar, the approach considered here has the advantage that the learning rate and the topology of the network are introduced within the stochastic approximation framework, which allows the convergence proof considered in the Appendix.

VI. Computational Results

A. Data Visualization Experiments

We have chosen 12 data sets in order to test the self-organization and data visualization capabilities of our proposal. We have selected high-dimensional image and video data. These kinds of data are commonly processed by SOMs in application domains such as image clustering and retrieval [18], [27], [28], [51] and video indexing [5]. Furthermore, these experiments show the self-organization capabilities of the PPCASOM model. That is, they show how the units adjust their parameters so that a computational map emerges.

The "Faces" database [50] is composed of 64 $\times$ 64 grayscale images (256 gray levels) which are versions of a computer-generated human face with different poses and lighting directions. This database has been used as a benchmark for high-dimensional data visualization [49], [63]. The "Zeros" to "Nines" databases are composed of 28 $\times$ 28 grayscale images (256 gray levels) of handwritten digits, and come from the MNIST Handwritten Digit Database [29]. These databases have been used as benchmarks for SOMs aimed to high-dimensional data processing as ours [38], and they have been also considered for the experimental evaluation of the visualization models mentioned above [49], [63]. The "Video" database [10] is composed of 64 $\times$ 52 grayscale images (256 gray levels), which have been obtained by reducing original video frames with 352 $\times$ 288 with 24 b/pixel (RGB color space). In all cases, the components of the input vectors are real numbers in the interval $[0, 1]$. We have used no other preprocessing on the original databases available from the Internet. The details of each database and the parameter selections for the PPCASOM are shown in Table I.

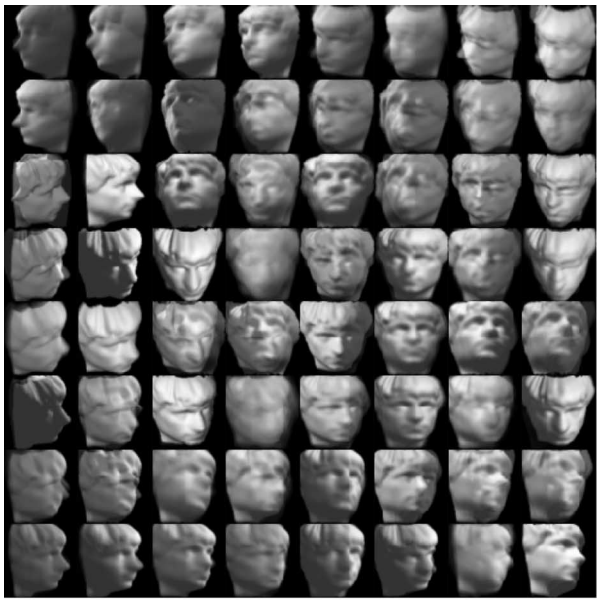
TABLE I
Data Sets and Parameter Selections

Data sets	# of input samples	d	q	N	ε_0
Faces	698	4096	1	100,000	0.01
Zeros...Nines	5421 ... 6742	784	3	100,000	0.01
Video	1262	3328	2	100,000	0.01

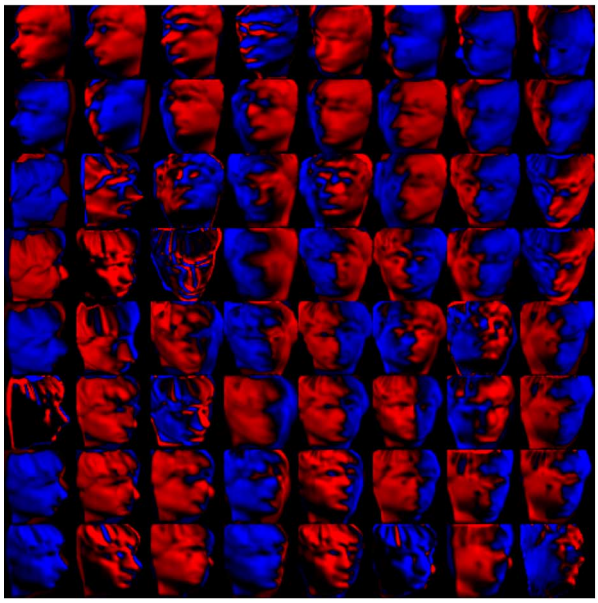
The choice of the dimension q of the latent subspaces is driven by the dimensionality reduction needs of the application at hand. Lower values of q yield more compact representations of the input data, but they will be less faithful (lower data likelihood). On the other hand, higher values of q correspond to more accurate accounts of the variability of the data (higher data likelihood), but they could compromise the ease of visualization. The amount of data to be visualized is also important, as there is no point in selecting a high q if there is not enough variability in the data. Hence, we have used values of q, which follow the number of samples of each database.

The final network states are pictured in Figs. 1–3 for data sets "Faces," "Twos," and "Video," respectively. We have plotted the mean vectors and the q first eigenvectors in image format. We can see that the network self-organizes, and that the mean vectors and eigenvectors capture relevant features of the input data sets. Note that each pixel corresponds to a different dimension of the input data set. In order to understand the pictures of the eigenvectors, it is important to remember that the entries of an eigenvector measure the amount of variability (dispersion) in each dimension with respect to the mean vector. The positive elements of the eigenvectors have been drawn in red, and the negative elements in blue (more color saturation indicates a larger value; color version of the figures are available in an online version of this paper). Only the changes of sign are relevant, and not the sign itself, because we can negate an eigenvector to obtain other eigenvector pointing in the same direction. The null elements of the eigenvectors are drawn in black, and they correspond to the dimensions with no variability with respect to the mean. Note that the pictures of the mean vectors are in gray levels because the input values lie in the interval $[0, 1]$, and hence, the mean vectors are nonnegative.

In order to compare the performance of the PPCASOM model with similar proposals, we have selected the SOMM by Verbeek *et al.* [59] and the joint entropy maximization kernel-based topographic maps (KBTM) by Van Hulle [57]. We have considered the homoskedastic version of Van Hulle's maps because the heteroskedastic version is $O(d^3)$ per step, which limits its use with the considered databases.



(a)



(b)

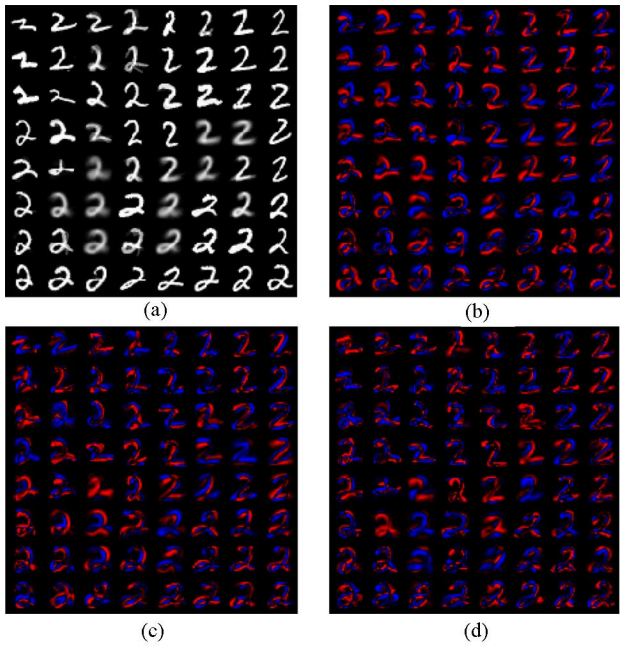
Fig. 1. Results for the "Faces" database: (a) mean vectors and (b) first eigenvectors.

The optimized version of SOMM has been used for the tests. We have simulated the KBTM for 2 000 000 steps, with parameters $\eta_w = 0.01$, $\eta_\sigma = 10_{-4}\,\eta_w$.

Since the three self-organizing models we are comparing define probability distributions, we have considered the average negative log-likelihood (ANLL) as the performance measure

$$\mathrm{ANLL} = -\frac{1}{K}\sum_{n=1}^{K}\log p(\mathbf{t}_n). \tag{38}$$

We have used the T-test to check the statistical significance of the advantage of PPCASOM results over SOMM and KBTM. The difference has been found to be statistically significant for all databases and topologies (less than 0.015 probability that the difference between the means is caused by chance), except for "Faces" with SOMM and 64×1 topology. In this case, the probability that the difference between the means is caused by chance is 0.1007.



(a) (b) (c) (d)

Fig. 2. Results for the "Twos" database: (a) mean vectors, (b) first eigenvectors, (c) second eigenvectors, and (d) third eigenvectors.

Hence, our proposal achieves a better self-organized representation of the considered input distributions with a small computational complexity.

The ANLL values are expected to be lower (which is better) as the number of units H grows. Hence, we have fixed $H = 64$ units in all the experiments. The results of the tenfold cross validation are shown in Tables II and III, with the standard deviations in parentheses. We have considered two different topologies: 2-D rectangular maps with 8×8 units (Table II) and 1-D maps with 64×1 units (Table III). We can see that the PPCASOM clearly outperforms SOMM and KBTM in all the tests.

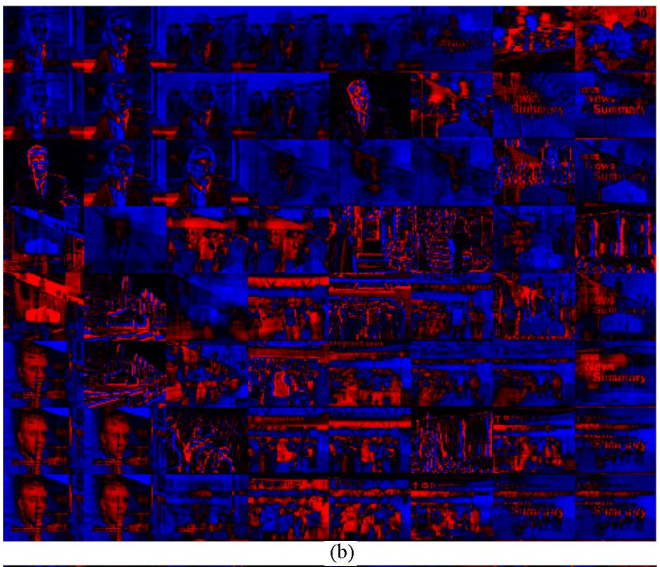
B. Image Compression Experiments

In this section, we explore the ability of the PPCASOM to build parsimonious representations of image data, when compared with other SOMs which define probability distributions such as the KBTM and the SOMM. We also include conventional PCA results for comparison with a classical linear technique. Image compression by SOMs is an active area of research, both theoretical and applied. Some recent proposals are [2], [3], [6], [40], and [48]. As the PPCASOM is a linear subspace model, it is natural to represent the input data by its projection on the orthonormal vector basis of each unit. The unit j which yields the least reconstruction error of an input sample $\mathbf{t}_n$ is given by

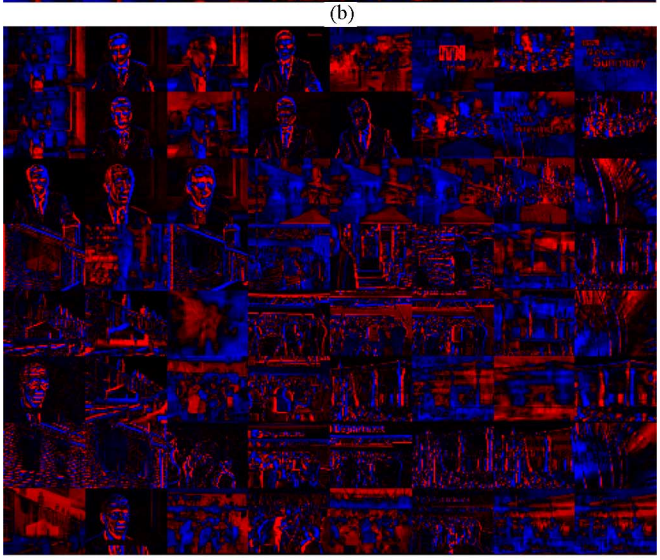
$$j = \arg\min_i \left\|\mathbf{t}_n - \hat{\mathbf{t}}_n^i\right\|^2 \tag{39}$$

where the reconstruction vector $\hat{\mathbf{t}}_n^i$ is obtained with (9). Note that we have dropped the subindex n in j for clarity. This unit

(a)



(b)



(c)

Fig. 3. Results for the "Video" database: (a) mean vectors, (b) first eigenvectors, and (c) second eigenvectors.

is used to obtain the reduced version of $\mathbf{t}_n$

$$\mathbf{z}_n = \mathbf{U}_{qj}^T(\mathbf{t}_n - \boldsymbol{\mu}_j) \tag{40}$$

TABLE II
ANLL RESULTS WITH 8 × 8 RECTANGULAR TOPOLOGY
(STANDARD DEVIATIONS IN PARENTHESES)

Data set	PPCASOM	SOMM	KBTM
Zeros	–287.41 (5.28)	–214.50 (3.37)	9.47 (3.42)
Ones	–927.85 (11.85)	–734.64 (9.92)	–323.48 (10.30)
Twos	–178.28 (4.89)	–130.41 (3.12)	20.26 (2.76)
Threes	–235.48 (5.55)	–184.01 (4.65)	–29.84 (2.33)
Fours	–300.89 (5.18)	–239.84 (7.00)	–69.60 (2.21)
Fives	–240.67 (5.78)	–190.49 (4.24)	–9.23 (3.77)
Sixes	–328.44 (12.69)	–253.62 (6.68)	–51.13 (3.30)
Sevens	–401.25 (8.43)	–315.10 (4.98)	–106.16 (2.71)
Eights	–211.52 (7.32)	–165.06 (4.48)	–22.61 (5.03)
Nines	–366.91 (8.82)	–291.92 (5.13)	–96.41 (3.51)
Faces	–3050.3 (228.38)	–2846.4 (139.52)	–452.28 (65.23)
Video	–6045.1 (334.23)	–2732.0 (414.70)	–705.06 (59.57)

TABLE III
ANLL RESULTS WITH 64 × 1 RECTANGULAR TOPOLOGY
(STANDARD DEVIATIONS IN PARENTHESES)

Data set	PPCASOM	SOMM	KBTM
Zeros	–282.47 (5.24)	–217.00 (4.61)	9.24 (2.93)
Ones	–927.56 (8.44)	–740.31 (13.69)	–326.66 (4.92)
Twos	–172.33 (4.70)	–131.39 (5.22)	20.62 (2.07)
Threes	–230.56 (5.24)	–185.46 (4.11)	–30.03 (2.90)
Fours	–293.37 (11.30)	–240.80 (6.11)	–69.30 (4.27)
Fives	–234.45 (6.37)	–192.16 (4.51)	–8.25 (2.27)
Sixes	–323.62 (5.50)	–256.75 (7.31)	–51.41 (2.74)
Sevens	–398.73 (6.96)	–317.75 (7.78)	–106.41 (3.99)
Eights	–206.71 (7.87)	–164.77 (2.47)	–21.89 (3.85)
Nines	–361.79 (7.69)	–292.65 (8.12)	–94.74 (4.16)
Faces	–3074.7 (325.7)	–2926.5 (129.0)	–434.3 (49.8)
Video	–6177.9 (511.5)	–3337.4 (292.9)	–736.4 (85.9)

where the reduced (projected) vector $\mathbf{z}_n$ has dimensions $q \times 1$. The optimal reconstructed vector is

$$\hat{\mathbf{t}}_n = \hat{\mathbf{t}}_n^j \tag{41}$$

with j given by (39). For conventional PCA, a similar approach is followed, but with only one global vector base $\mathbf{U}_q$. For the KBTM and SOMM models, each input sample $\mathbf{t}_n$ is represented by the mean vector of the unit which is closest (in the MSE sense) to $\mathbf{t}_n$.

In the image compression context, it is convenient to split the original image into equally sized windows, so that each window is an input vector. Here we have used windows of 8×8 pixels, which is a common choice in many cases such as JPEG [60]. This corresponds to an input space dimension of $d = 64$. Smaller window sizes result in poor compression ratios, while larger windows compromise image quality. On the other hand, each component of the input vectors is an integer value in the range $[0, 255]$, as we consider grayscale images with 8 b of precision. Since each unit of the three considered self-organizing models stores an approximation of the mean vector, for these

(a)

(b)

Fig. 4. Original images: (a) "Lena" and (b) "monarch."

cases, we need to store the following data in the compressed file.

1) The mean vectors for each unit. First they are rounded to 8 b, and then they are Huffman encoded to remove the redundancy [17].
2) The index of the best matching unit for each window of the original image. These integers are represented with the least possible number of bits, and then Huffman encoded.

For conventional PCA, we only need to store the global mean. Note that the encoding has been exactly the same in the four cases in order to make a fair comparison. Only in the PPCASOM and PCA cases we need to store additional data.

1) The orthogonal vector bases $\mathbf{U}_q$ for each unit are represented with 32-b floating point numbers. We make use of the fact that these real numbers are in the range $[-1, 1]$, since they are part of unit norm vectors.
2) The components of the projected vectors $\mathbf{z}_n$ are quantized with a variable number of bits (from 1 to 10 b) and run length encoding of the sequences of zero values. The quantization procedure involves the division of each component of $\mathbf{z}_n$ by an integer constant, the rounding of the obtained quotient, and the codification of the resulting integer with the desired number of bits. Finally, the resulting string of bits is Huffman encoded. We have tuned the quality of the compression by choosing smaller integer constants to obtain more quality, and *vice versa*.

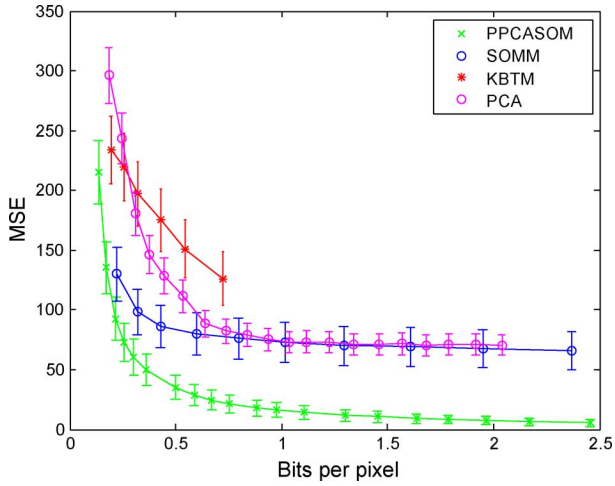


Fig. 5. MSE versus BPP for the "Lena" image. Note that nearer to the coordinate origin is better.

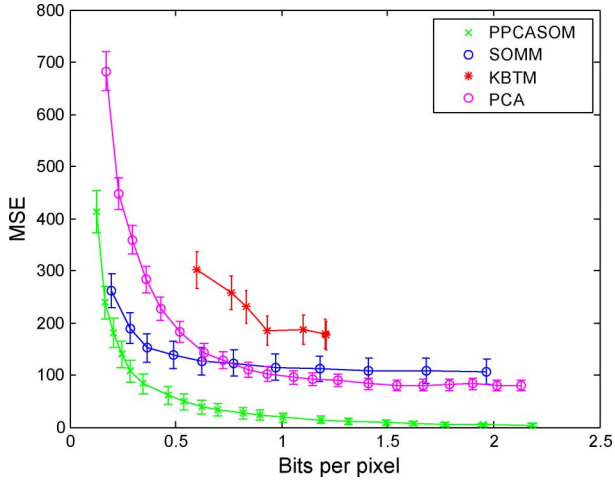


Fig. 6. MSE versus BPP for the "monarch" image. Note that nearer to the coordinate origin is better.

We have divided the input data randomly into two disjoint subsets: the training set, with 90% of the windows, and the test set, with the remaining 10%. The data compression performance has been evaluated by two measures. First, the MSE per pixel has been computed for the test set

$$\mathrm{MSE} = \frac{1}{64 \cdot \mathrm{NTest}} \sum_{n=1}^{\mathrm{NTest}} ||\mathbf{t}_n - \hat{\mathbf{t}}_n||^2 \tag{42}$$

where NTest is the number of windows of the test set, and $\hat{\mathbf{t}}_n$ is the optimal reconstructed vector. For the KBTM and the SOMM, this is equal to the best matching mean vector, in the MSE sense. For the PPCASOM and the PCA, it is computed with (41); for conventional PCA, we do not need to choose among units because there is only one vector basis. On the other hand, the size of the compressed file has been considered, expressed as number of bits per pixel (BPP)

$$\mathrm{BPP} = \frac{\mathrm{FileSize}}{\mathrm{NPix}} \tag{43}$$

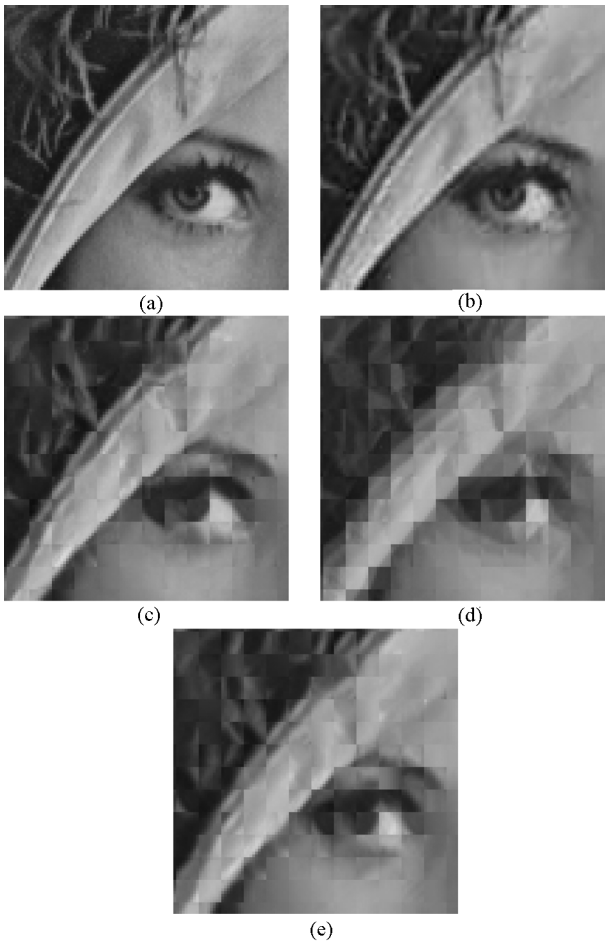


Fig. 7. Detail of "Lena": (a) original (8 b/pixel), (b) PPCASOM compressed (0.6696 b/pixel), (c) SOMM compressed (0.7953 b/pixel), (d) KBTM compressed (0.7212 b/pixel), and (e) PCA compressed (0.8132 b/pixel).

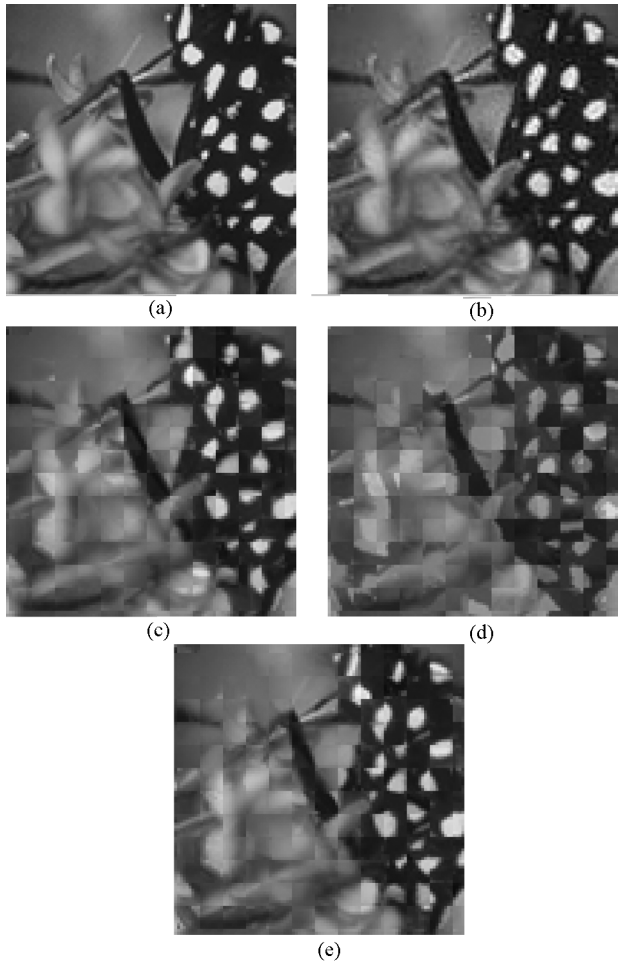


Fig. 8. Detail of "monarch": (a) original (8 b/pixel), (b) PPCASOM compressed (1.0082 b/pixel), (c) SOMM compressed (1.1780 b/pixel), (d) KBTM compressed (1.2047 b/pixel), and (e) PCA compressed (1.1698 b/pixel).

where $\mathrm{FileSize}$ is the size of the compressed file in bits, and NPix is the number of pixels of the image.

The parameter selection for the three models has been the same as in Section VI-A, except for the size of the maps, which has been varied in order to tune the quality of the compression. As the maps are larger, the quality of the reconstructed image is better, but there are more storage requirements. For KBTM and SOMM, we have experimented with maps with sizes in the range from 5×5 to 50×50 units. For the PPCASOM, since each unit has more compression capability, we have used smaller maps: from 2×2 to 5×5 units, with sizes of the vector bases in the range from $q = 1$ to 60. In the three cases, only the results for the best performing model sizes have been shown in the plots. For conventional PCA, we have tested basis vector sizes in the range from $q = 1$ to 60.

We can see in Fig. 4 two benchmark images taken from the University of Waterloo Repertoire [54]. The "Lena" image has 512×512 pixels, and "monarch" has 768×512 pixels. The plots of MSE versus BPP are in Figs. 5 and 6. Please note that every model yields its own BPP values, so they cannot be forced to produce the exact same BPP. The results show that our proposal achieves lower error at all bit rates in both of the tested images. It is worth mentioning that the KBTM model is not able to learn correctly when the map size is too large for the variability of the input data set. This causes the KBTM data series to have different lengths in the plots. On the other hand, Figs. 7 and 8 show that PPCASOM yields better visual image quality with less bit rate. We can conclude that PPCASOM is able to take advantage of its linear subspace self-organization ability to achieve efficient dimensionality reduction.

C. Video Compression Experiments

The third set of experiments studies the capability of our proposal to compress video data. SOMs have been proposed recently as vector quantizers for video compression [14], [41]. Here we evaluate the potential of PPCASOM, KBTM, and SOMM for this purpose. Additionally, we include conventional PCA results for comparison with a linear technique, as in Section VI-B.

Lossy compression of video data is commonly performed by dividing the frame (picture) sequence into small groups of pictures (GOPs). Each GOP contains three types of frames [46].

1) Intraframes (I-Frames) are complete images. They are compressed completely independent of other frames. Hence, the coding of I-Frames is a standard lossy image

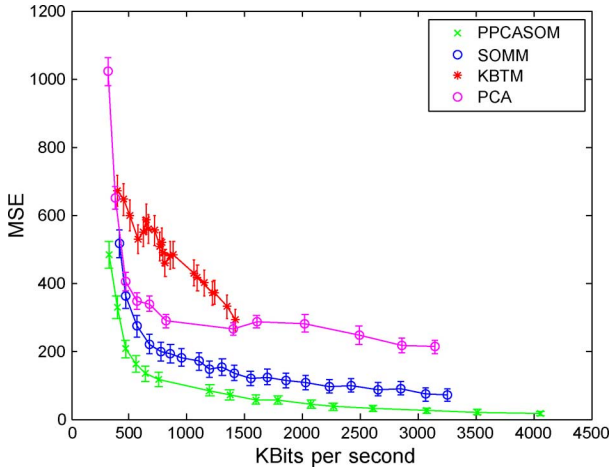


Fig. 9. MSE versus kilobits per second (kb/s) for the "container" video. Note that nearer to the coordinate origin is better.

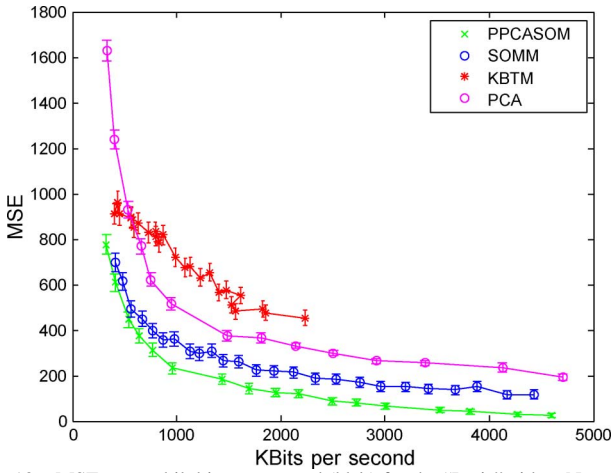


Fig. 10. MSE versus kilobits per second (kb/s) for the "Paris" video. Note that nearer to the coordinate origin is better.

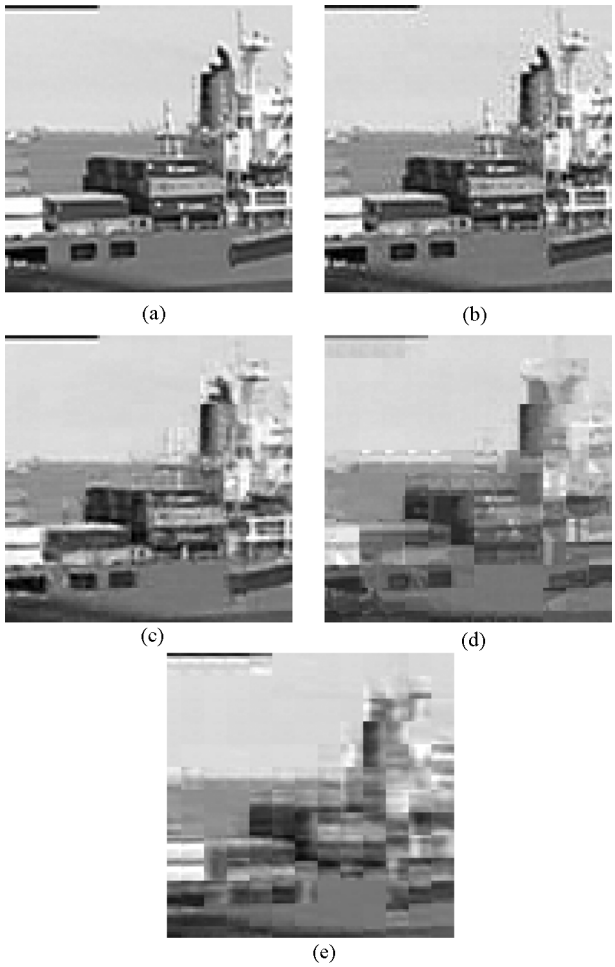


Fig. 11. Detail of the 51st frame of "container": (a) original (20.2752 Mbps), (b) PPCASOM compressed (3.0714 Mb/s), (c) SOMM compressed (3.2522 Mb/s), (d) KBTM compressed (1.4188 Mb/s), and (e) PCA compressed (3.1436 Mb/s).

compression problem. Typically, the first frame of a GOP is an I-Frame.

2) Predicted frames (P-Frames) are images within the GOP that are created using information from preceding images. Typically, the difference between the P-Frame and the first frame of the GOP (which is an I-Frame) is lossy compressed.
3) Bidirectional frames (B-Frames) are images within the GOP that are created using information from preceding images and images that follow. The usage of B-Frames needs the evaluation of which other images will do the best to approximate the B-Frame. Since this is a complex issue in its own, we are not using B-Frames in these experiments in order to avoid side effects which could make the comparison unfair.

Additionally, each image is divided into small blocks of pixels in order to perform the lossy compression. In our experiments, we have considered a GOP length of 12 frames, i.e., we have the first I-Frame and 11 following P-Frames which are coded as the differences with respect to the I-Frame. In each frame, we have used blocks of 8×8 pixels, and we have compressed only the Y component (luminance), which is given as an integer in the range $[0, 255]$. Both choices (GOP length and block size) are typical in MPEG, for example [46]. This means that we have input vectors of dimension $d = 64$ and two SOMs to be trained for each GOP: one for the I-Frame and the other for the 11 difference images of the P-Frames. We have divided the input data for each SOM randomly into two disjoint subsets: the training set, with 90% of the blocks, and the test set, with the remaining 10%. The encoding of the SOMs has been the same as in Section VI-B

The evaluation of the results has been similar to the image compression experiments. First, the MSE per pixel has been computed for the test sets, with (42). On the other hand, the size of the compressed file has been considered, in this case expressed as kilobits per second (kb/s) of video

$$\mathrm{kb/s} = \frac{\mathrm{FileSize}}{\mathrm{NSec}} \tag{44}$$

where FileSize is the size of the compressed file in kilobits, and NSec is the duration of the video sequence in seconds.

We have selected two freely available benchmark video sequences from the Xiph.org Test Media [62]. They are commonly used for video compression evaluation [4], [7], [16], [21] [34].

Fig. 12. Detail of the 855th frame of "Paris": (a) original (20.2752 Mb/s), (b) PPCASOM compressed (4.2695 Mb/s), (c) SOMM compressed (4.4292 Mb/s), (d) KBTM compressed (2.2345 Mb/s), and (e) PCA compressed (4.7050 Mb/s).

The original sequences are in the uncompressed common interchange format (CIF), with frames of 352 × 288 pixels. The parameter selection for the three models has been the same as in Section VI-B. For the KBTM and the SOMM, we have experimented with map sizes in the range from 2 × 2 to 30 × 30 units. For the PPCASOM, we have used smaller maps because each unit has more compression capability: from 2 × 2 to 5 × 5 units, with sizes of the vector bases in the range from $q = 1$ to 60. In the three cases, only the results for the best performing model sizes have been shown in the plots. The same range of vector base sizes (from $q = 1$ to 60) has been used for conventional PCA.

The plots of MSE versus kilobits per second are in Figs. 9 and 10. As every model yields its own kilobits-per-second values, they cannot be forced to produce the exact same kilobits per second. It can be seen that our proposal achieves lower error at all bit rates in both of the tested videos. The KBTM data series end sooner for the same reason as in the image compression experiments, i.e., it does not learn correctly with larger map sizes. On the other hand, Figs. 11 and 12 show that PPCASOM yields better visual quality with less bit rate. As in the image compression experiments, these results indicate that PPCASOM is suitable for dimensionality reduction, due to its linear subspace self-organization ability.

VII. Conclusion

We have presented a probabilistic self-organizing neural model, which features online learning of the local principal subspaces of the input data. It is based on a mixture of Gaussians where only a certain number q of relevant principal directions is considered. It is particularly suited for high-dimensional data because it has a low computational complexity. Experimental results have been presented that show the self-organization capabilities of the model and its performance in image and video compression applications. In particular, it outperforms two SOMs based on mixtures of homoskedastic Gaussians. Hence, our model achieves both scalability and a correct representation of the input distribution.

Appendix
Stochastic Approximation

First, we consider the original (batch) M-step equations for $\mathbf{W}$ and σ at time step n (see [53])

$$\widetilde{\pi}_i = \frac{1}{n}\sum_{j=1}^{n} R_{ji} \tag{A.1}$$

$$\widetilde{\boldsymbol{\mu}}_i = \frac{\sum_{j=1}^{n} R_{ji}\mathbf{t}_j}{\sum_{j=1}^{n} R_{ji}} \tag{A.2}$$

$$\widetilde{\boldsymbol{W}}_i = \left[\sum_{j=1}^{n} R_{ji}(\mathbf{t}_j - \boldsymbol{\mu}_i) E\left[\mathbf{x}_j^T|i\right]\right]\left[\sum_{j=1}^{n} R_{ji} E\left[\mathbf{x}_j\mathbf{x}_j^T|i\right]\right]^{-1} \tag{A.3}$$

$$\widetilde{\sigma}_i^2 = \frac{1}{nd}\sum_{j=1}^{n} R_{ji}\left\{\|\mathbf{t}_j - \boldsymbol{\mu}_i\|^2 - 2E\left[\mathbf{x}_j^T|i\right]\widetilde{\boldsymbol{W}}_i^T(\mathbf{t}_j - \boldsymbol{\mu}_i) + \operatorname{tr}\left(E\left[\mathbf{x}_j\mathbf{x}_j^T|i\right]\widetilde{\boldsymbol{W}}_i^T\widetilde{\boldsymbol{W}}_i\right)\right\} \tag{A.4}$$

where the corresponding E-step equations are

$$E\left[\mathbf{x}_j^T|i\right] = (\mathbf{t}_j - \boldsymbol{\mu}_i)^T\mathbf{W}_i\mathbf{M}_i^{-1} \tag{A.5}$$

$$E\left[\mathbf{x}_j\mathbf{x}_j^T|i\right] = \sigma_i^2\mathbf{M}_i^{-1} + \left(E\left[\mathbf{x}_j^T|i\right]\right)^T E\left[\mathbf{x}_j^T|i\right] \tag{A.6}$$

$$\mathbf{M}_i = \sigma_i^2\mathbf{I} + \mathbf{W}_i^T\mathbf{W}_i. \tag{A.7}$$

Several weighted means appear implicitly in (A.1)–(A.4), where the weights are the responsibilities R_{ji}. Let $\boldsymbol{\Theta}_i = (\boldsymbol{\mu}_i, \mathbf{W}_i, \sigma_i)$ be a vector comprising the parameters for mixture component i, and let $\varphi(\boldsymbol{\Theta}_i, \mathbf{t})$ be an arbitrary function of $\boldsymbol{\Theta}_i$ and the input sample $\mathbf{t}$. Then, we define the weighted mean of $\varphi(\boldsymbol{\Theta}_i, \mathbf{t})$ as

$$\langle\varphi\rangle_i = E\left[P(i|\mathbf{t})\varphi(\boldsymbol{\Theta}_i, \mathbf{t})\right]. \tag{A.8}$$

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

$$\langle\varphi\rangle_i = \frac{1}{n}\sum_{j=1}^{n} P(i|\mathbf{t}_j)\varphi(\boldsymbol{\Theta}_i, \mathbf{t}_j) = \frac{1}{n}\sum_{j=1}^{n} R_{ji}\varphi(\boldsymbol{\Theta}_i, \mathbf{t}_j). \tag{A.9}$$

As $n \to \infty$, the approximation of (A.9) converges to the exact value given by (A.8). In this case, we can rewrite equations

(A.1)–(A.4) in terms of the weighted means $\langle\varphi\rangle_i$

$$\widetilde{\pi}_i = \langle 1\rangle_i \tag{A.10}$$

$$\widetilde{\boldsymbol{\mu}}_i = \frac{\langle \mathbf{t}\rangle_i}{\langle 1\rangle_i} \tag{A.11}$$

$$\widetilde{\boldsymbol{W}}_i = \left(\left\langle (\mathbf{t}-\boldsymbol{\mu}_i) E[\mathbf{x}^T|i]\right\rangle_i\right)\left(\left\langle E[\mathbf{x}\mathbf{x}^T|i]\right\rangle_i\right)^{-1} \tag{A.12}$$

$$\widetilde{\sigma}_i^2 = \left\langle \frac{1}{d}\|\mathbf{t}-\boldsymbol{\mu}_i\|^2 - 2E[\mathbf{x}^T|i]\widetilde{\boldsymbol{W}}_i^T(\mathbf{t}-\boldsymbol{\mu}_i) + \operatorname{tr}\left(E[\mathbf{x}\mathbf{x}^T|i]\widetilde{\boldsymbol{W}}_i^T\widetilde{\boldsymbol{W}}_i\right)\right\rangle_i . \tag{A.13}$$

On the other hand, we can apply the Robbins–Monro stochastic approximation algorithm (see [26]) to estimate iteratively the value of the weighted means $\langle\varphi\rangle_i$

$$\langle\varphi\rangle_i(0) = P(i|\mathbf{t}_0)\varphi(\boldsymbol{\Theta}_i, \mathbf{t}_0) \tag{A.14}$$

$$\langle\varphi\rangle_i(n) = \langle\varphi\rangle_i(n-1) + \varepsilon(n)\left(P(i|\mathbf{t}_n)\varphi(\boldsymbol{\Theta}_i, \mathbf{t}_n) - \langle\varphi\rangle_i(n-1)\right) \tag{A.15}$$

where $\varepsilon(n)$ is the step size, which must satisfy the following conditions in order to guarantee convergence of the Robbins–Monro method:

$$\varepsilon(n) > 0 \quad \lim_{n\to\infty}\varepsilon(n) = 0 \quad \sum_{n=1}^{\infty}\varepsilon(n) = \infty \quad \sum_{n=1}^{\infty}\varepsilon^2(n) < \infty. \tag{A.16}$$

In order to fulfill these requirements, $\varepsilon(n)$ is typically selected as

$$\varepsilon(n) = \frac{1}{an+b}, \qquad 0 < a < 1. \tag{A.17}$$

On the other hand, (A.15) is more conveniently written as

$$\langle\varphi\rangle_i(n) = (1-\varepsilon(n))\langle\varphi\rangle_i(n-1) + \varepsilon(n)R_{ni}\varphi(\boldsymbol{\Theta}_i, \mathbf{t}_n). \tag{A.18}$$

Now we derive an online EM algorithm by applying (A.18) to (A.10)–(A.13). First, we define three auxiliary variables

$$\boldsymbol{\Omega}_i = \left\langle (\mathbf{t}-\boldsymbol{\mu}_i)E[\mathbf{x}^T|i]\right\rangle_i \tag{A.19}$$

$$\boldsymbol{\Xi}_i = \left\langle E[\mathbf{x}\mathbf{x}^T|i]\right\rangle_i \tag{A.20}$$

$$\mathbf{m}_i = \langle \mathbf{t}\rangle_i. \tag{A.21}$$

The corresponding update equations are

$$\boldsymbol{\Omega}_i(n) = (1-\varepsilon(n))\boldsymbol{\Omega}_i(n-1) + \varepsilon(n)R_{ni}(\mathbf{t}_n - \boldsymbol{\mu}_i)E\left[\mathbf{x}_n^T|i\right] \tag{A.22}$$

$$\boldsymbol{\Xi}_i(n) = (1-\varepsilon(n))\boldsymbol{\Xi}_i(n-1) + \varepsilon(n)R_{ni}E\left[\mathbf{x}_n\mathbf{x}_n^T|i\right] \tag{A.23}$$

$$\mathbf{m}_i(n) = (1-\varepsilon(n))\mathbf{m}_i(n-1) + \varepsilon(n)R_{ni}\mathbf{t}_n. \tag{A.24}$$

Then, we are ready to rewrite (A.10)–(A.13)

$$\pi_i(n) = (1-\varepsilon(n))\pi_i(n-1) + \varepsilon(n)R_{ni} \tag{A.25}$$

$$\boldsymbol{\mu}_i(n) = \frac{\mathbf{m}_i(n)}{\pi_i(n)} \tag{A.26}$$

$$\mathbf{W}_i(n) = \boldsymbol{\Omega}_i(n)\left(\boldsymbol{\Xi}_i(n)\right)^{-1} \tag{A.27}$$

$$\sigma_i^2(n) = (1-\varepsilon(n))\sigma_i^2(n-1) + \varepsilon(n)R_{ni}\frac{1}{d} \times \left(\|\mathbf{t}_n - \boldsymbol{\mu}_i\|^2 - 2E\left[\mathbf{x}_n^T|i\right]\mathbf{W}_i(n)^T(\mathbf{t}_n - \boldsymbol{\mu}_i) + \operatorname{tr}\left(E\left[\mathbf{x}_n\mathbf{x}_n^T|i\right]\mathbf{W}_i(n)^T\mathbf{W}_i(n)\right)\right). \tag{A.28}$$

Proposition: If (A.16) holds, then the stochastic approximation algorithm of (A.25)–(A.28) converges to a maximum of the likelihood.

Proof: The general form of the Robbins–Monro stochastic algorithm is

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \varepsilon(n)Y(n) \tag{A.29}$$

where, in our case, we take

$$\boldsymbol{\theta}(n) = \left(\langle\varphi\rangle_i(n)\right)_{\varphi,i}. \tag{A.30}$$

That is, we include all the weighted means in the current estimation vector $\theta(n)$. We also take

$$Y(n) = \xi(n) - \boldsymbol{\theta}(n) \tag{A.31}$$

where the new data to be incorporated into the estimation is

$$\xi(n) = \left(P\left(i|\mathbf{t}_n, \boldsymbol{\Theta}(n), Q_n\right)\varphi\left(\boldsymbol{\Theta}_i(n), \mathbf{t}_n\right)\right)_{\varphi,i}. \tag{A.32}$$

In the above equation, $\bar{\boldsymbol{\Theta}} = (\boldsymbol{\mu}_i, \boldsymbol{W}_i, \sigma_i)_i$ is the complete parameter vector of the SOM, and Q_n is the distribution set which is used to enforce self-organization at time step n. The goal of the stochastic algorithm is to find a root of the equation

$$\overline{g}\left(\boldsymbol{\theta}(n)\right) = \mathbf{0} \tag{A.33}$$

where

$$\overline{g}\left(\boldsymbol{\theta}(n)\right) = \left(E\left[P\left(i|\mathbf{t}, \boldsymbol{\Theta}(n), Q_\infty\right)\varphi\left(\boldsymbol{\Theta}_i(n), \mathbf{t}\right)\right] - \langle\varphi\rangle_i(n)\right)_{\varphi,i} \tag{A.34}$$

and Q_∞ is the limit distribution set

$$q_{\infty ii} = 1 \quad q_{\infty ij} = 0 \qquad \forall i \neq j \tag{A.35}$$

which is such that the Q_n tend to Q_∞ as $n \to \infty$

$$\lim_{n\to\infty} q_{nij} = q_{\infty ij} \qquad \forall i \forall j. \tag{A.36}$$

In order to prove the convergence of the algorithm, we are going to prove that the "noise" in the observation Y_n is a *martingale difference* (see [26]). That is, there is a function $g_n(\cdot)$ of θ such that

$$E\left[Y_n|\boldsymbol{\theta}(0), Y_j, j < n\right] = g_n\left(\boldsymbol{\theta}(n)\right). \tag{A.37}$$

This is readily verified

$$g_n\left(\boldsymbol{\theta}(n)\right) = \left(E\left[P\left(i|\mathbf{t}, \boldsymbol{\Theta}(n), Q_n\right)\varphi\left(\boldsymbol{\Theta}_i(n), \mathbf{t}\right)\right]\right)_{\varphi,i} - \boldsymbol{\theta}(n). \tag{A.38}$$

Note that $g_n(\theta(n))$ only depends on n and $\theta(n)$, because Q_n is obtained from n and $\mathbf{\Theta}(n)$ can be computed from $\theta(n)$. So we have that

$$Y_n = g_n\left(\boldsymbol{\theta}(n)\right) + \delta M_n \tag{A.39}$$

where δM_n is a martingale difference. The associated "bias" process is defined as

$$\beta_n = E\left[Y_n|\boldsymbol{\theta}(0), Y_j, j < n\right] - \overline{g}\left(\boldsymbol{\theta}(n)\right). \tag{A.40}$$

We can guarantee convergence by proving the following three conditions:

$$\sup_n E\left[||Y_n||^2\right] < \infty \tag{A.41}$$

$$\lim_{n\to\infty} \beta_n = \mathbf{0} \tag{A.42}$$

$$\forall m > 0 \forall \boldsymbol{\theta}, \quad \lim_{n\to\infty} \left\| \sum_{j=n}^{n+m} \varepsilon_j \left(g_j(\boldsymbol{\theta}) - \overline{g}(\boldsymbol{\theta})\right) \right\| = 0. \tag{A.43}$$

Next we examine the first condition

$$\begin{aligned} &E\left[||Y_n||^2\right] \\ &\quad = E\left[\left|\left|\xi(n) - \boldsymbol{\theta}(n)\right|\right|^2\right] \\ &\quad = \sum_{\varphi,i} E\left[\left|\left|P\left(i|\mathbf{t}, \mathbf{\Theta}(n), Q_n\right)\varphi\left(\mathbf{\Theta}_i(n), \mathbf{t}\right) - \langle\varphi\rangle_i(n)\right|\right|^2\right]. \end{aligned} \tag{A.44}$$

The above value is finite if we assume that the input distribution has a compact support. This assumption should hold in practice, since real data always appear in a finite domain. Alternatively, we can relax this assumption by considering that the input density decreases exponentially as $||\mathbf{t}|| \to \infty$.

Now we study the second condition

$$\begin{aligned} \lim_{n\to\infty} \beta_n &= \lim_{n\to\infty} \left(E\left[Y_n|\boldsymbol{\theta}(0), Y_j, j < n\right] - \overline{g}\left(\boldsymbol{\theta}(n)\right)\right) \\ &= \lim_{n\to\infty} \left(E\left[P\left(i|\mathbf{t}, \mathbf{\Theta}(n), Q_n\right)\varphi\left(\mathbf{\Theta}_i(n), \mathbf{t}\right)\right]\right. \\ &\qquad \left. - E\left[P\left(i|\mathbf{t}, \mathbf{\Theta}(n), Q_\infty\right)\varphi\left(\mathbf{\Theta}_i(n), \mathbf{t}\right)\right]\right)_{\varphi,i}. \end{aligned} \tag{A.45}$$

This limit is zero because (A.36) implies that

$$\lim_{n\to\infty} P\left(i|\mathbf{t}, \mathbf{\Theta}(n), Q_n\right) = P\left(i|\mathbf{t}, \mathbf{\Theta}(n), Q_\infty\right). \tag{A.46}$$

Finally, for the third condition, we have

$$\begin{aligned} \forall \boldsymbol{\theta}, \quad \lim_{j\to\infty} \left(g_j(\boldsymbol{\theta}) - \overline{g}(\boldsymbol{\theta})\right) &= \lim_{j\to\infty} \left(E\left[P(i|\mathbf{t}, \mathbf{\Theta}, Q_j)\varphi(\mathbf{\Theta}_i, \mathbf{t})\right]\right. \\ &\qquad \left. - E\left[P(i|\mathbf{t}, \mathbf{\Theta}, Q_\infty)\varphi(\mathbf{\Theta}_i, \mathbf{t})\right]\right)_{\varphi,i} \end{aligned} \tag{A.47}$$

which is also zero because of (A.36). In turn, this means that the limit

$$\lim_{n\to\infty} \left\| \sum_{j=n}^{n+m} \varepsilon_j \left(g_j(\boldsymbol{\theta}) - \overline{g}(\boldsymbol{\theta})\right) \right\| \tag{A.48}$$

is zero $\forall m > 0 \forall \boldsymbol{\theta}$ since it is the norm of a finite sum of two factors where both tend to zero.

Hence, we have proved that the algorithm converges to a root of (A.33). At convergence, we have

$$\langle\varphi\rangle_i(n) = E\left[P\left(i|\mathbf{t}, \mathbf{\Theta}(n), Q_\infty\right)\varphi\left(\mathbf{\Theta}_i(n), \mathbf{t}\right)\right] \qquad \forall\varphi\forall i \tag{A.49}$$

which is equivalent to the maximum-likelihood condition

$$\frac{\partial L}{\partial \mathbf{\Theta}} = \mathbf{0} \tag{A.50}$$

where

$$L(\mathbf{\Theta}) = E\left[\log p(\mathbf{t}|\mathbf{\Theta}, Q_\infty)\right] \tag{A.51}$$

because (A.49) is a fixed point of (A.10)–(A.13).

ACKNOWLEDGMENT

The authors would like to thank the associate editor and the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] M. Aladjem, "Projection pursuit mixture density estimation," *IEEE Trans. Signal Process.*, vol. 53, no. 11, pp. 4376–4383, Nov. 2005.

[2] C. Amerijckx, M. Verleysen, P. Thissen, and J.-D. Legat, "Image compression by self-organized Kohonen map," *IEEE Trans. Neural Netw.*, vol. 9, no. 3, pp. 503–507, May 1998.

[3] C. Amerijckx, J.-D. Legat, and M. Verleysen, "Image compression using self-organizing maps," *Syst. Anal. Model. Simul.*, vol. 43, no. 11, pp. 1529–1543, 2003.

[4] S. H. Baek, Y. H. Moon, and J. H. Kim, "An improved H.264/AVC video encoding based on a new syntax element," *J. Vis. Commun. Image Represent.*, vol. 17, no. 2, pp. 345–357, 2006.

[5] T. Barecke, E. Kijak, A. Nurnberger, and M. Detyniecki, "Summarizing video information using self-organizing maps," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Vancouver, BC, Canada, 2006, pp. 540–546.

[6] S. Battiato, F. Rundo, and F. Stanco, "Self organizing motor maps for color-mapped image re-indexing," *IEEE Trans. Image Process.*, vol. 16, no. 12, pp. 2905–2915, Dec. 2007.

[7] U. Bayazit, "Classification-based macroblock layer rate control for low delay transmission of H.263 video," *J. Electron. Imag.*, vol. 12, no. 3, pp. 499–510, 2003.

[8] C. M. Bishop, M. Svensen, and C. K. I. Williams, "GTM: The generative topographic mapping," *Neural Comput.*, vol. 10, no. 1, pp. 215–234, 1998.

[9] D. G. Calò, "Gaussian mixture model classification: A projection pursuit approach," *Comput. Statist. Data Anal.*, vol. 52, pp. 471–482, 2007.

[10] G. Daniel and M. Chen, Video Visualization Benchmark Resources, 2006 [Online]. Available: http://www.swan.ac.uk/compsci/research/graphics/vg/video/

[11] S. Dharanipragada and K. Visweswariah, "Gaussian mixture models with covariances or precisions in shared multiple subspaces," *IEEE Trans. Audio Speech Lang. Process.*, vol. 14, no. 4, pp. 1255–1266, Jul. 2006.

[12] M. Dittenbach, A. Rauber, and D. Merkl, "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.

[13] R. D. Dony and S. Haykin, "Optimally adaptive transform coding," *IEEE Trans. Image Process.*, vol. 4, no. 10, pp. 1358–1370, Oct. 1995.

[14] K. L. Ferguson and N. M. Allinson, "Efficient video compression codebooks using SOM-based vector quantisation," *IEE Proc.—Vis. Image Signal Process.*, vol. 151, no. 2, pp. 102–108, 2004.

[15] K. Fukushima, "Self-organization of shift-invariant receptive fields," *Neural Netw.*, vol. 12, no. 6, pp. 791–802, 1999.

[16] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 5, pp. 645–658, May 2005.

[17] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. Inst. Radio Eng.*, vol. 40, no. 9, pp. 1098–1101, 1952.

[18] M. Hussain and J. P. Eakins, "Component-based visual clustering using the self-organizing map," *Neural Netw.*, vol. 20, no. 2, pp. 260–273, 2007.

[19] A. Hyvarinen, P. O. Hoyer, and M. Inki, "Topographic independent component analysis," *Neural Comput.*, vol. 13, no. 7, pp. 1527–1558, 2001.

[20] L. O. Jiménez-Rodríguez, E. Arzuaga-Cruz, and M. Vélez-Reyes, "Unsupervised linear feature-extraction methods and their effects in the classification of high-dimensional data," *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 2, pp. 469–483, Feb. 2007.

[21] E. Kaminsky, D. Grois, and O. Hadar, "Dynamic computational complexity and bit allocation for optimizing H.264/AVC video compression," *J. Vis. Commun. Image Represent.*, vol. 19, no. 1, pp. 56–74, 2008.

[22] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990.

[23] T. Kohonen, "The Adaptive-Subspace SOM (ASSOM) and its use for the implementation of invariant feature detection," in *Proc. Int. Conf. Artif. Neural Netw.*, F. Fogelman-Soulié and P. Galniari, Eds., 1995, vol. 1, pp. 3–10.

[24] T. Kohonen, "Emergence of invariant-feature detectors in the adaptive-subspace SOM," *Biol. Cybern.*, vol. 75, pp. 281–291, 1996.

[25] T. Kohonen, S. Kaski, and H. Lappalainen, "Self-organized formation of various invariant-feature filters in the adaptive-subspace SOM," *Neural Comput.*, vol. 9, no. 6, pp. 1321–1344, 1997.

[26] H. J. Kushner and G. G. Yin, *Stochastic approximation and Recursive Algorithms and Applications*, 2nd ed. New York: Springer-Verlag, 2003.

[27] J. Laaksonen, M. Koskela, and E. Oja, "PicSOM-self-organizing image retrieval with MPEG-7 content descriptors," *IEEE Trans. Neural Netw.*, vol. 13, no. 4, pp. 841–853, Jul. 2002.

[28] J. Laaksonen, M. Koskela, and E. Oja, "Class distributions on SOM surfaces for feature extraction and object retrieval," *Neural Netw.*, vol. 17, no. 8–9, pp. 1121–1133, 2004.

[29] T. LeCun and C. Cortes, The MNIST Database of Handwritten Digits, 2006 [Online]. Available: http://yann.lecun.com/exdb/mnist/

[30] E. López-Rubio, J. Muñoz-Pérez, and J. A. Gómez-Ruiz, "A principal components analysis self-organizing map," *Neural Netw.*, vol. 17, no. 2, pp. 261–270, 2004.

[31] E. López-Rubio, J. M. Ortiz-de-Lazcano-Lobato, D. López-Rodríguez, and M. C. Vargas-González, "Self-organization of probabilistic PCA models," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2007, vol. 4507, pp. 211–218.

[32] S. P. Luttrell, "Derivation of a class of training algorithms," *IEEE Trans. Neural Netw.*, vol. 1, no. 2, pp. 229–232, Jun. 1990.

[33] E. Oja, "Neural networks, principal components, and subspaces," *Int. J. Neural Syst.*, vol. 1, pp. 61–68, 1989.

[34] B. G. Mobasseri and D. Cinalli, "Lossless watermarking of compressed media using reversibly decodable packets," *Signal Process.*, vol. 86, no. 5, pp. 951–961, 2006.

[35] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 696–709, Jul. 1997.

[36] K. Okajima, "Two-dimensional Gabor-type receptive field as derived by mutual information maximization," *Neural Netw.*, vol. 11, no. 3, pp. 441–447, 1998.

[37] K. Okajima, "The Gabor function extracts the maximum information from input local signals," *Neural Netw.*, vol. 11, no. 3, pp. 435–439, 1998.

[38] J. Ontrup and H. Ritter, "Large-scale data exploration with the hierarchically growing hyperbolic SOM," *Neural Netw.*, vol. 19, no. 6–7, pp. 751–761, 2006.

[39] S. A. Orlitsky, "Semi-parametric exponential family PCA," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2004, vol. 17.

[40] S.-C. Pei, Y.-T. Chuang, and W.-H. Chuang, "Effective palette indexing for image compression using self-organization of Kohonen feature map," *IEEE Trans. Image Process.*, vol. 15, no. 9, pp. 2493–2498, Sep. 2006.

[41] A. Ramirez-Agundis, R. Gadea-Girones, R. Colom-Palero, and J. Diaz-Carmona, "A wavelet-VQ system for real-time video compression," *J. Real-Time Image Process.*, vol. 2, no. 4, pp. 271–280, 2007.

[42] J. Ruiz del Solar and M. Köppen, "A texture segmentation architecture based on automatically generated oriented filters," *J. Microelectron. Syst. Integr.*, vol. 5, no. 1, pp. 43–52, 1997.

[43] J. Ruiz del Solar, "TEXSOM: Texture segmentation using self-organizing maps," *Neurocomputing*, vol. 21, no. 1–3, pp. 7–18, 1998.

[44] N. Saitoa, R. R. Coifman, F. B. Geshwind, and F. Warner, "Discriminant feature extraction using empirical probability density estimation and a local basis library," *Pattern Recognit.*, vol. 35, pp. 2841–2852, 2002.

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

[46] W.-C. Siu, Y.-L. Chan, and K.-T. Fung, "On transcoding a b-frame to a p-frame in the compressed domain," *IEEE Trans. Multimedia*, vol. 9, no. 6, pp. 1093–1102, Oct. 2007.

[47] N. V. Subba Reddy and P. Nagabhushan, "A three-dimensional neural network model for unconstrained handwritten numeral recognition: A new approach," *Pattern Recognit.*, vol. 31, pp. 511–516, 1998.

[48] N. Sudha, "An ASIC implementation of Kohonen's map based colour image compression," *Real-Time Imag.*, vol. 10, no. 1, pp. 31–39, 2004.

[49] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2322, 2000.

[50] J. B. Tenenbaum, V. de Silva, and J. C. Langford, Data Sets for Nonlinear Dimensionality Reduction, 2006 [Online]. Available: http://isomap.stanford.edu/datasets.html

[51] C. Theoharatos, N. Laskaris, G. Economou, and S. Fotopoulos, "Combining self-organizing neural nets with multivariate statistics for efficient color image retrieval," *Comput. Vis. Image Understand.*, vol. 102, no. 3, pp. 250–258, 2006.

[52] M. E. Tipping and C. M. Bishop, "A hierarchical latent variable model for data visualization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 281–293, Mar. 1998.

[53] M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal components analyzers," *Neural Comput.*, vol. 11, pp. 443–482, 1999.

[54] University of Waterloo Repertoire of Images, Waterloo, ON, Canada, 2008 [Online]. Available: http://links.uwaterloo.ca/colorset.base.html

[55] M. M. Van Hulle, "Faithful representations with topographic maps," *Neural Netw.*, vol. 12, no. 6, pp. 803–823, 1999.

[56] M. M. Van Hulle, "Kernel-based topographic map formation by local density modeling," *Neural Comput.*, vol. 14, no. 7, pp. 1561–1573, 2002.

[57] M. M. Van Hulle, "Joint entropy maximization in kernel-based topographic maps," *Neural Comput.*, vol. 14, no. 8, pp. 1887–1906, 2002.

[58] M. M. Van Hulle, "Maximum likelihood topographic map formation," *Neural Comput.*, vol. 17, no. 3, pp. 503–513, 2005.

[59] J. J. Verbeek, N. Vlassis, and B. J. A. Krose, "Self-organizing mixture models," *Neurocomputing*, vol. 63, pp. 99–123, 2005.

[60] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consumer Electron.*, vol. 38, no. 1, pp. 18–34, Feb. 1992.

[61] C. Wang and W. Wang, "Links between PPCA and subspace methods for complete Gaussian density estimation," *IEEE Trans. Neural Netw.*, vol. 17, no. 3, pp. 789–792, May 2006.

[62] Xiph.org Test Media, 2008 [Online]. Available: http://media.xiph.org/video/derf/

[63] L. Yang, "Distance-preserving projection of high-dimensional data for nonlinear dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 9, pp. 1243–1246, Sep. 2004.

[64] H. Yin and N. M. Allinson, "Self-organizing mixture networks for probability density estimation," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 405–411, Mar. 2001.

[65] B. Zhang, M. Fu, H. Yan, and M. A. Jabri, "Handwritten digit recognition by adaptive-subspace self-organizing map (ASSOM)," *IEEE Trans. Neural Netw.*, vol. 10, no. 4, pp. 939–945, Jul. 1999.

[66] J. Zhao and Q. Jiang, "Probabilistic PCA for t distributions," *Neurocomputing*, vol. 69, no. 16–18, pp. 2217–2226, 2006.

Ezequiel López-Rubio was born in 1976. He received the M.Sc. and Ph.D. (honors) degrees 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 technical interests are in unsupervised learning, pattern recognition, and image processing.

Juan Miguel Ortiz-de-Lazcano-Lobato was born in 1978. He received the M.Tech. degree in computer science and the Ph.D. degree in software engineering and artificial intelligence from the University of Málaga, Málaga, Spain, in 2003 and 2007, respectively.

Since 2008, he has been an Associate Professor at the University of Málaga. His research interests include fields such as multidimensional data analysis, pattern recognition and neural networks, especially competitive and self-organizing neural networks.

Domingo López-Rodríguez was born in 1978. He received the Ph.D. degree in applied mathematics from the University of Málaga, Málaga, Spain, in 2007.

Since 2003, he has been a Researcher with the Soft Computing and Image Analysis Unit, University of Málaga. His research interests include optimization, neural networks, and combinatorial optimization problems. He is particularly interested in the application of multivalued recurrent networks to solve optimization problems on graphs, and in the development of a general framework to study competitive learning. Web-based education and distance learning are also fields of his study.