Self-organization of Probabilistic PCA Models

Ezequiel López-Rubio, Juan Miguel Ortiz-de-Lazcano-Lobato,
Domingo López-Rodríguez, and María del Carmen Vargas-González

School of Computer Engineering
University of Málaga
Campus de Teatinos, s/n. 29071 Málaga.
Spain
Phone: (+34) 95 213 71 55
Fax: (+34) 95 213 13 97
{ezeqlr,jmortiz}@lcc.uma.es, dlopez@ctima.uma.es

Abstract. We present a new neural model, which extends Kohonen's self-organizing map (SOM) by performing a Probabilistic Principal Components Analysis (PPCA) at each neuron. Several self-organizing maps have been proposed in the literature to capture the local principal subspaces, but our approach offers a probabilistic model at each neuron while it has linear complexity on the dimensionality of the input space. This allows to process very high dimensional data to obtain reliable estimations of the local 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.

Keywords: Probabilistic Principal Components Analysis (PPCA), competitive learning, unsupervised learning, dimensionality reduction, face recognition, handwritten digit recognition.

1 Introduction

The concept of self-organization seems to explain several neural structures of the brain that perform invariant feature detection. These structures inspired the proposal of computational maps designed to explore multidimensional data. The original self-organizing map (SOM) was proposed by Kohonen [2], where each neuron had a weight vector to represent a point of the input space.

The Self-Organizing Mixture Model (SOMM) by Verbeek et al. [8] 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 self-organizing maps include kernel-based topographic maps [7], where Gaussian kernels are defined around a centroid.

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 framework. Furthermore, our proposal has linear complexity both in the size of the map and in the input space dimension, so that it is suited for high dimensional data.

The outline of the paper is as follows. In Section 2 we present the new model, called *Probabilistic Principal Components Analysis Self-Organizing Map (PPCASOM)*. Section 3 is devoted to computational experiments. Finally, conclusions are presented in Section 4.

2 The PPCASOM Model

2.1 Mixture Model

Each neuron of the map stores a PPCA model [6] 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 ε:

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

The latent variables are defined to be independent and gaussian with unit variance and zero mean. The noise model is also gaussian such that $\varepsilon \sim N(0, \sigma\mathbf{I})$, 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 $\mathbf{C} = \sigma\mathbf{I} + \mathbf{W}\mathbf{W}^T$.

The PPCASOM model is defined as a mixture of H PPCA units (or neurons), where the prior probabilities (or mixing proportions π_i) are constrained to be equal: $\forall i,\ \pi_i = 1/H$. This restriction is aimed to achieve self-organization at the network (mixture) level by replacing the standard likelihood-guided mixture learning with a topology-guided learning, as we will see in subsection 2.4. That is, we seek topologically ordered states of the network which may not be states of maximum likelihood. At the neuron level the network topology is not relevant, so the maximum likelihood approach is used (subsection 2.3).

2.2 Competitive Process

At each time step n the network is presented a data sample $\mathbf{t}_n$. Then, a Bayesian model selection is carried out to decide which neuron has the most probability $p(i \mid \mathbf{t}_n)$ of having generated the sample. This neuron is called the *winning neuron*. Since the prior probabilities π_i are equal, this is equivalent to find the maximum value of $p(\mathbf{t}_n \mid i)$:

$$Winner(n) = \arg\max_i \{p(i \mid \mathbf{t}_n)\} = \arg\max_i \{p(\mathbf{t}_n \mid i)\} \tag{2}$$

where

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

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

Now we need a procedure to compute $p(\mathbf{t}_n \mid i)$ 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:

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

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 $\hat{\mathbf{t}}_n^i$.

2.3 Neuron Update

When a new sample $\mathbf{t}_n$ has been presented to the network, its information should be used to update the neurons. If we want to update neuron i with the information from sample $\mathbf{t}_n$, an online version of the original expectation-maximization (EM) method of the PPCA model is required. The online EM will generate the updated values $\sigma_i(n)$ and $\mathbf{W}_i(n)$ from the old values $\sigma_i(n-1)$, $\mathbf{W}_i(n-1)$ and the new sample $\mathbf{t}_n$.

First we consider the original M-step equations for $\mathbf{W}$ and σ:

$$\tilde{\mathbf{W}}_i = \left[\sum_{j=1}^{n}\left(\mathbf{t}_j - \boldsymbol{\mu}_i\right)\left\langle \mathbf{x}_j^{\;T}\right\rangle_i\right]\left[\sum_{j=1}^{n}\left\langle \mathbf{x}_j\mathbf{x}_j^{\;T}\right\rangle_i\right]^{-1} \tag{7}$$

$$\tilde{\sigma}_i^2 = \frac{1}{nd}\sum_{j=1}^{n}\left\{\left\|\mathbf{t}_j - \boldsymbol{\mu}_i\right\|^2 - 2\left\langle \mathbf{x}_j^{\;T}\right\rangle_i \tilde{\mathbf{W}}_i^T\left(\mathbf{t}_j - \boldsymbol{\mu}_i\right) + tr\left(\left\langle \mathbf{x}_j\mathbf{x}_j^{\;T}\right\rangle_i \tilde{\mathbf{W}}_i^T\tilde{\mathbf{W}}_i\right)\right\} \tag{8}$$

where the corresponding E-step equations are:

$$\left\langle \mathbf{x}_j^{\;T}\right\rangle_i = \mathbf{M}_i^{-1}\mathbf{W}_i^T\left(\mathbf{t}_j - \boldsymbol{\mu}_i\right) \tag{9}$$

$$\left\langle \mathbf{x}_j\mathbf{x}_j^{\;T}\right\rangle_i = \sigma_i^2\mathbf{M}_i^{-1} + \left\langle \mathbf{x}_j^{\;T}\right\rangle_i\left(\left\langle \mathbf{x}_j^{\;T}\right\rangle_i\right)^T \tag{10}$$

$$\mathbf{M}_i = \sigma_i^2\mathbf{I} + \mathbf{W}_i^T\mathbf{W}_i \tag{11}$$

The summations on j of equations (7) and (8) represent the contributions of all the input samples $\mathbf{t}_j$ which have been incorporated so far into the PPCA model of neuron i. So, we can split these contributions in two parts: the past samples (j=1,...,n–1) and the current sample (j=n). In the case of matrix $\mathbf{W}_i$, this leads to rewrite equation (7), with a 1/n factor added for notational convenience:

$$\tilde{\mathbf{W}}_i = \left[\frac{1}{n} (\mathbf{t}_n - \boldsymbol{\mu}_i) \left\langle \mathbf{x}_n^T \right\rangle_i + \frac{1}{n} \sum_{j=1}^{n-1} (\mathbf{t}_j - \boldsymbol{\mu}_i) \left\langle \mathbf{x}_j^T \right\rangle_i \right] \left[\frac{1}{n} \left\langle \mathbf{x}_n \mathbf{x}_n^T \right\rangle_i + \frac{1}{n} \sum_{j=1}^{n-1} \left\langle \mathbf{x}_j \mathbf{x}_j^T \right\rangle_i \right]^{-1} \tag{12}$$

If we note

$$\boldsymbol{\Omega}_i = \frac{1}{n-1} \sum_{j=1}^{n-1} (\mathbf{t}_j - \boldsymbol{\mu}_i) \left\langle \mathbf{x}_j^T \right\rangle_i \tag{13}$$

$$\boldsymbol{\Xi}_i = \frac{1}{n-1} \sum_{j=1}^{n-1} \left\langle \mathbf{x}_j \mathbf{x}_j^T \right\rangle_i \tag{14}$$

$$\tilde{\boldsymbol{\Omega}}_i = \alpha_i(n) (\mathbf{t}_n - \boldsymbol{\mu}_i) \left\langle \mathbf{x}_n^T \right\rangle_i + (1 - \alpha_i(n)) \boldsymbol{\Omega}_i \tag{15}$$

$$\tilde{\boldsymbol{\Xi}}_i = \alpha_i(n) \left\langle \mathbf{x}_n \mathbf{x}_n^T \right\rangle_i + (1 - \alpha_i(n)) \boldsymbol{\Xi}_i \tag{16}$$

where $\alpha_i(n)$= 1/n, then equation (12) can be rewritten as

$$\tilde{\mathbf{W}}_i = \tilde{\boldsymbol{\Omega}}_i \left(\tilde{\boldsymbol{\Xi}}_i \right)^{-1} \tag{17}$$

Now the EM method uses equations (15), (16) and (17) at each iteration. It must be noted that the value of $\mathbf{W}_i$ is updated in every EM iteration by equation (17), while the values of $\boldsymbol{\Omega}_i$ and Ξ_i are not updated until the EM algorithm has converged. This means that in the right side of equations (15) and (16) we always use the old values $\boldsymbol{\Omega}_i(n\text{–}1)$ and $\Xi_i(n\text{–}1)$ obtained with the previous sample $\mathbf{t}_{n\text{–}1}$. The reason is that $\boldsymbol{\Omega}_i(n\text{–}1)$ and $\Xi_i(n\text{–}1)$ store valuable information from past input samples. Hence we restrict the optimization so that $\boldsymbol{\Omega}_i(n\text{–}1)$ and $\Xi_i(n\text{–}1)$ are fixed. Furthermore, this avoids the reprocessing of the past input samples with the updated values of $\mathbf{W}_i$. The use of equations (15) and (16) at each EM iteration is needed only to account for the effect of the change in the estimations of $\mathbf{W}_i$ and $\mathbf{M}_i$ on the present input sample $\mathbf{t}_n$ during the EM optimization.

If we vary $\alpha_i(n)$ in the range $0 \le \alpha_i(n) \le 1$, then equations (15) and (16) become weighted sums of the terms given by (9) and (10). High values of $\alpha_i(n)$ mean that the present input sample is given more importance in the weighted sum with respect to past samples, and vice versa. That is, $\alpha_i(n)$ controls the amount of learning at time step n. A similar approach can be used with σ, and the final result is:

$$\tilde{\sigma}_i^2 = \alpha_i(n) \frac{1}{d} \left\{ \left\| \mathbf{t}_n - \boldsymbol{\mu}_i \right\|^2 - 2 \left\langle \mathbf{x}_n^T \right\rangle_i \tilde{\mathbf{W}}_i^T (\mathbf{t}_n - \boldsymbol{\mu}_i) + tr\left(\left\langle \mathbf{x}_n \mathbf{x}_n^T \right\rangle_i \tilde{\mathbf{W}}_i^T \tilde{\mathbf{W}}_i \right) \right\} + (1 - \alpha_i(n)) \sigma_i^2 \tag{18}$$

where $\alpha_i(n)=1/n$. Here σ_i^2 is taken from the previous time step n–1, in order to take advantage of the accumulated sum. Equation (18) is used at each iteration of the EM optimization. Finally, the mean vector is updated without the need of an optimization:

$$\boldsymbol{\mu}(n)=\frac{1}{n}\sum_{j=1}^{n}\mathbf{t}_j=\frac{1}{n}\mathbf{t}_n+\frac{n-1}{n}\sum_{j=1}^{n-1}\mathbf{t}_j=\alpha_i(n)\mathbf{t}_n+(1-\alpha_i(n))\boldsymbol{\mu}(n-1) \tag{19}$$

where again $\alpha_i(n)=1/n$. Equation (19) should be used before the EM optimization, in order to use the best approximation possible to the mean vector.

2.4 Self-organizing Map Formation

The emergence of a self-organizing pattern is obtained by tuning the learning control parameter $\alpha_i(n)$. First of all, a *topology* is defined in the network so that the topological distance between neurons p and q is called $\delta(p,q)$. A rectangular lattice may be used. Other topologies could be also considered, like hexagonal or toroidal.

The learning control parameter is selected for each neuron i so that the neurons which are nearer to the winner learn more:

$$\alpha_i(n)=\eta(n)\exp\left(-\frac{\delta(i,Winner(n))}{\Delta(n)}\right) \tag{20}$$

where both the *learning rate* $\eta(n)$ and the *neighbourhood radius* $\Delta(n)$ are decreasing functions of n such that $0\le\eta(n)\le 1$ and $\Delta(n)>0$. A standard choice is the linear decay. Note that if we used $\alpha_i(n)=\eta(n)$ we would get a competitive learning neural network with no self-organization.

2.5 Summary

The PPCASOM model can be summarized as follows:

1. Set the initial values $\boldsymbol{\mu}_i(0)$, $\sigma_i(0)$, $\mathbf{W}_i(0)$, $\boldsymbol{\Omega}_i(0)$ and $\Xi_i(0)$ for all neurons i following the standard PPCA initialization presented in [5].
2. Choose an input sample $\mathbf{t}_n$ and compute the winning neuron by (2).
3. For every neuron i, compute its learning parameter $\alpha_i(n)$ with (20).
4. For every neuron i, estimate its mean vector $\boldsymbol{\mu}_i(n)$ by equation (19).
5. For every neuron i, run the EM iteration by using equations (15), (16), (17) and (18) until the EM method converges. The values obtained at convergence are the updated values $\sigma_i(n)$, $\mathbf{W}_i(n)$, $\boldsymbol{\Omega}_i(n)$ and $\Xi_i(n)$.
6. If the map has converged or the maximum time step has been reached, stop. Otherwise, go to step 2.

The above PPCASOM algorithm has linear complexity in the size of the input space dimension d. It is also linear in the number of neurons H. This allows to process high dimensional data sets, as we will see in the computational experiments.

3 Computational Results

We have chosen four high-dimensional data sets in order to test the self-organization ability of our proposal. The 'Faces' database [4] is composed of 64×64 grayscale images (256 gray levels) which are versions of a computer-generated human face with different poses and lighting directions. The 'Sixes' and 'Twos' databases are composed of 28×28 grayscale images (256 gray levels) of handwritten sixes and twos, and come from the MNIST Handwritten Digit Database [3]. The 'Video' database [1] is composed of 64×52 grayscale images (256 gray levels), which have been obtained by reducing original video frames with 352×288 with 24 bits per pixel (RGB color space). The details are shown in Table 1.

Table 1. Data sets and parameter selections for PPCASOM. *T* is the number of epochs, *d* is the dimension of the input space and *q* is the dimension of the latent subspace.

Data set	*# of input samples*	*T*	*d*	*q*	*Topology*
Faces	698	30000	4096	2	7×7, flat
Sixes	5918	30000	784	5	8×8, flat
Twos	5958	30000	784	3	8×8, flat
Video	1262	30000	3328	2	8×8, toroidal

The final network states are pictured in Figures 1 and 2 for datasets 'Faces' and 'Twos'. 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 eigenvectors capture relevant features of the input data sets.

In order to compare the performance of the PPCASOM model with similar proposals, we have selected the Self-Organizing Mixture Model (SOMM) by Verbeek et al. (2005) and the joint entropy maximization kernel based topographic maps (KBTM) by Van Hulle (2002b). 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.

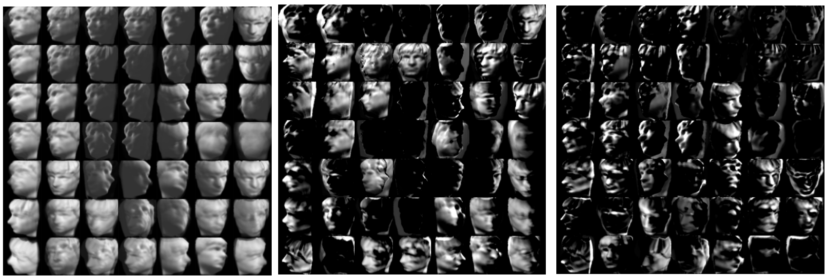

Fig. 1. Results for the 'Faces' database: mean vectors (left), first eigenvectors (middle) and second eigenvectors (right)

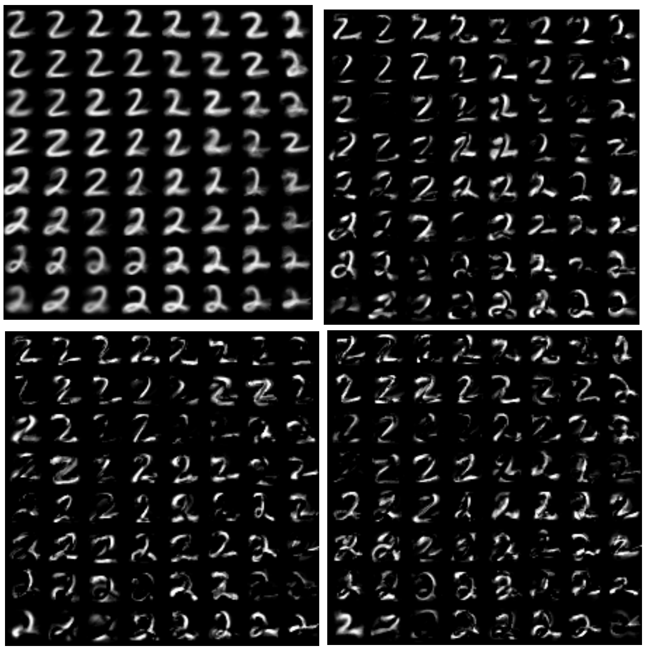

Fig. 2. Results for the 'Twos' database. From up to down and from left to right: mean vectors, first eigenvectors, second eigenvectors and third eigenvectors.

The size of the map has been the same for PPCASOM, SOMM and KBTM, and can be seen on the 'Topology' column of Table 1. The optimized version of SOMM has been used for the tests. We have simulated the KBTM for 2,000,000 steps, with parameters η_w=0.01, η_σ=$10^{-4}\eta_w$.

Since the three self-organizing models we are comparing define probability distributions, we measure their performance by the average negative log-likelihood:

$$ANLL = -\frac{1}{P}\sum_{n=1}^{P}\log p(\mathbf{t}_n) \tag{21}$$

The results of the 10-fold cross-validation are shown in Table 2. We can see that the PPCASOM clearly outperforms SOMM and KBTM in all the tests. Hence our proposal achieves a better representation of the input with a small complexity.

Table 2. ANLL mean values for the considered data sets (standard deviations in parentheses)

Data set	*PPCASOM*	*SOMM*	*KBTM*
Faces	–5472.35 (195.70)	–2846.4 (139.52)	–452.28 (65.23)
Sixes	–338.46 (10.4)	–252.72 (7.56)	–52.35 (3.31)
Twos	–175.01 (4.85)	–130.36 (2.94)	–19.98 (2.55)
Video	–7254.48 (291.57)	–2732.0 (414.70)	–705.06 (59.57)

4 Conclusions

We have presented a new 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 linear computational complexity. Experimental results have been presented that show the self-organization capabilities of the model. In particular, it outperforms two self-organizing maps based on mixtures of homoskedastic Gaussians. Hence, our model achieves both scalability and a correct representation of the input distribution.

Acknowledgements

This work was partially supported by the Ministry of Education and Science of Spain under Projects TIN2005-02984 and TIN2006-07362.

References

1. Daniel, G., Chen, M.: Video Visualization Benchmark Resources. (2006) In: Internet (November 2006) http://www.swan.ac.uk/compsci/research/graphics/vg/video/
2. Kohonen, T.: The Self-Organizing Map. In: Proc. IEEE, 78, 1464-1480 (1990)
3. LeCun, Y., Cortes, C.: The MNIST Database of Handwritten Digits. (2006) In: Internet (November 2006) http://yann.lecun.com/exdb/mnist/
4. Tenenbaum, J. B., de Silva, V., Langford, J. C.: Data sets for nonlinear dimensionality reduction. (2006) In: Internet (November 2006) http://isomap.stanford.edu/datasets.html
5. Tipping, M.E., Bishop, C.M.: A Hierarchical Latent Variable Model for Data Visualization. IEEE Trans. on Pattern Analysis and Machine Intelligence 20(3), 281–293 (1998)
6. Tipping, M.E., Bishop, C.M.: Mixtures of Probabilistic Principal Components nalyzers, Neural Computation, 11, 443-482 (1999)
7. Van Hulle, M.M.: Joint Entropy Maximization in Kernel-Based Topographic Maps. Neural Computation 14(8), 1887–1906 (2002)
8. Verbeek, J.J., Vlassis, N., Krose, B.J.A.: Self-organizing mixture models. Neurocomputing 63, 99–123 (2005)