Automatic Model Selection for Probabilistic PCA

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. The Mixture of Probabilistic Principal Components Analyzers (MPPCA) is a multivariate analysis technique which defines a Gaussian probabilistic model at each unit. The number of units and principal directions in each unit is not learned in the original approach. Variational Bayesian approaches have been proposed for this purpose, which rely on assumptions on the input distribution and/or approximations of certain statistics. Here we present a different way to solve this problem, where cross-validation is used to guide the search for an optimal model selection. This allows to learn the model architecture without the need of any assumptions other than those of the basic PPCA framework. Experimental results are presented, which show the probability density estimation capabilities of the proposal with high dimensional data.

Keywords: Probabilistic Principal Components Analysis (PPCA), dimensionality reduction, cross-validation, handwritten digit recognition.

1 Introduction

The original Mixtures of Probabilistic PCA (MPPCA) models [9] do not address the problem of selecting the optimal number of units (neurons) M nor the number of basis vectors q_i for each unit i. This problem has been studied in the context of global PCA [2]. It has been also considered in the context of Bayesian PCA ([5], [6], [8]).

The basic MPPCA framework obtains the optimal model parameters θ that maximize the data likelihood p($\mathbf{t} \mid \theta$). This maximum likelihood (ML) strategy fails to take into account the problem of model complexity, since more complex models are not penalized, and this produces overfitting. The Variational Bayesian PPCA treats the model paramaters θ as random variables, and averages over the range of models they define. So, the data evidence p($\mathbf{t}$) is used instead of data likelihood p($\mathbf{t} \mid \theta$). However, these averages produce integrals that are analytically intractable, and we end up with various more or less exact approximations. Moreover, since the model paramaters θ are now random variables, the above mentioned process of averaging requires that a probability model for θ is assumed. Hence we get an approximation of a probability model of the parameters of a probability model of the real data, which may be very far from the real input probability distribution. We propose to avoid these idealizations by using a sound data-driven strategy such as the cross-validation. Furthermore, we propose a method to integrate the model parameter selection in the learning process.

The outline of the paper is as follows. Section 2 is devoted to the original MPPCA model. In Section 3 we present the new model, called *Dynamic Mixtures of Probabilistic Principal Components Analyzers (DMPPCA).* Section 4 is devoted to computational experiments. Finally, conclusions are presented in Section 5.

2 The MPPCA Model

2.1 Mixture Model

Each unit i of the network stores a PPCA model [9] to perform a dimensionality reduction from the observed (input) space dimension d to the latent (reduced) subspace dimension q_i, with $q_i<d$. 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:

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

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

The MPPCA model is defined as a mixture of M PPCA units, where the prior probabilities or mixing proportions are called π_i. The global probability distribution is defined as

$$p(\mathbf{t}) = \sum_{i=1}^{M} \pi_i p(\mathbf{t} \mid i) \tag{2}$$

2.2 Parameter Estimation

Given M and q_i, the maximum likelihood (ML) values of π_i, $\mathbf{W}_i$ and σ_i can be computed by a expectation-maximization (EM) iterative algorithm [9]. At the convergence of the EM algorithm, the parameter matrix $\mathbf{W}_i$ can be decomposed as

$$\mathbf{W}_i = \mathbf{U}_i\left(\boldsymbol{\Lambda}_i - \sigma_i^2\mathbf{I}\right)^{1/2}\mathbf{R}_i \tag{3}$$

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

3 The DMPPCA Model

The DMPPCA model is an extension of the MPPCA where two procedures act in sequence: a vector basis resizing procedure and a procedure to change the number of units (mixture components) of the model. At each iteration, a variation of +1 or –1 in the number of basis vectors q_i and in the number of units M is evaluated. In order to implement cross-validation, the training set used to compute the MPPCA parameters π_i, $\mathbf{W}_i$ and σ_i is disjoint with the validation set used to decide the changes in M and q_i.

3.1 Vector Basis Reduction

If we want to decrement the number of basis vectors of unit i by one, $\tilde{q}_i = q_i - 1$, we build the new eigenvalue matrix $\tilde{\mathbf{\Lambda}}_i$ by selecting the q_i–1 largest elements of the diagonal of $\mathbf{\Lambda}_i$. The corresponding q_i–1 eigenvectors form $\tilde{\mathbf{U}}_i$. Then, the new estimation of the noise variance takes into account the discarded principal direction:

$$\tilde{\sigma}_i^2 = \frac{1}{d - \tilde{q}_i}\left((d - q_i)\sigma_i^2 + \lambda_{q_i}\right) \tag{4}$$

Finally, equation (3) is used to obtain $\tilde{\mathbf{W}}_i$, where we take $\tilde{\mathbf{R}}_i = \mathbf{I}$.

3.2 Vector Basis Growth

When we need to increment the number of basis vectors of unit i by one, $\tilde{q}_i = q_i + 1$, we face with the problem of computing the $\tilde{q}_i$-th eigenvector without computing the covariance matrix $\mathbf{S}_i$ explicitly. This is required in order to maintain the linear computational complexity of the method in the size of the input space d. We propose to use the Symmetric Power method [3] to find the eigenvector corresponding to the largest eigenvalue of the linear subspace of dimension d–q_i formed by the d–q_i least important principal directions. We take into account that the difference vector $\mathbf{t}_n$–$\boldsymbol{\mu}_i$ of a training sample $\mathbf{t}_n$ can be decomposed by projecting it onto the d principal directions of the local subspace:

$$\mathbf{t}_n - \boldsymbol{\mu}_i = \sum_{j=1}^{d}\left((\mathbf{t}_n - \boldsymbol{\mu}_i)^T \mathbf{u}_j^i\right)\mathbf{u}_j^i \tag{5}$$

where $\mathbf{u}_j^i$ is the j-th eigenvector of the basis of unit i. On the other hand, the covariance matrix $\mathbf{S}_i$ can also be decomposed in its eigenvalues and eigenvectors:

$$\mathbf{S}_i = \sum_{j=1}^{d}\lambda_j \mathbf{u}_j^i \left(\mathbf{u}_j^i\right)^T = \frac{1}{\sum_n R_{ni}}\sum_n R_{ni}(\mathbf{t}_n - \boldsymbol{\mu}_i)(\mathbf{t}_n - \boldsymbol{\mu}_i)^T \tag{6}$$

where $R_{ni} = p(\mathbf{t}_n \mid i)\pi_i / p(\mathbf{t}_n)$ is the resposibility of mixture component i for the generation of training sample $\mathbf{t}_n$. By substituting (5) in (6) we obtain

$$\mathbf{S}_i = \frac{1}{\sum_n R_{ni}} \sum_n \sum_{j=1}^{d} R_{ni} \left((\mathbf{t}_n - \boldsymbol{\mu}_i)^T \mathbf{u}_j^i \right)^2 \mathbf{u}_j^i \left(\mathbf{u}_j^i\right)^T = \frac{1}{\sum_n R_{ni}} \sum_{j=1}^{d} \sum_n R_{ni} \left((\mathbf{t}_n - \boldsymbol{\mu}_i)^T \mathbf{u}_j^i \right)^2 \mathbf{u}_j^i \left(\mathbf{u}_j^i\right)^T \tag{7}$$

So we can consider a covariance matrix which takes into account only the d–q_i lesser principal directions:

$$\hat{\mathbf{S}}_i = \sum_{j=q_i+1}^{d} \lambda_j \mathbf{u}_j^i \left(\mathbf{u}_j^i\right)^T \tag{8}$$

Now we need to compute the largest eigenvalue of $\hat{\mathbf{S}}_i$, and its corresponding eigenvector, by the Symmetric Power method. This method needs to evaluate the product of $\hat{\mathbf{S}}_i$ by an arbitrary vector $\mathbf{v}$, which can be done in $O(d)$ by considering the projection error vector instead of the difference vector $\mathbf{t}_n$–$\boldsymbol{\mu}_i$:

$$\hat{\mathbf{S}}_i \mathbf{v} = \frac{1}{\sum_n R_{ni}} \sum_n R_{ni} \left(\mathbf{t}_n - \hat{\mathbf{t}}_n^i\right) \left(\left(\mathbf{t}_n - \hat{\mathbf{t}}_n^i\right)^T \mathbf{v} \right) \tag{9}$$

where the reconstruction vector of training sample $\mathbf{t}_n$ for the unit i is found as follows:

$$\hat{\mathbf{t}}_n^i = \boldsymbol{\mu}_i + \mathbf{W}_i \left(\mathbf{W}_i^T \mathbf{W}_i\right)^{-1} \mathbf{W}_i^T (\mathbf{t}_n - \boldsymbol{\mu}_i) \tag{10}$$

The eigenvalue/eigenvector pair computed by the Symmetric Power method is used as the $\tilde{q}_i$-th columns of $\tilde{\boldsymbol{\Lambda}}_i$ and $\tilde{\mathbf{U}}_i$, respectively. Then we apply (3) to get $\tilde{\mathbf{W}}_i$, as before. Finally,

$$\tilde{\sigma}_i^2 = \frac{1}{d - \tilde{q}_i} \left((d - q_i)\sigma_i^2 - \lambda_{\tilde{q}_i} \right) \tag{11}$$

3.3 Basis Size Selection

In order to decide if increment, decrement, or leave as it is the basis vector size of unit i, we use the criterion of the maximum log-likelihood L measured over the validation set,

$$L = \sum_n \ln p(\mathbf{t}_n) = \sum_n \ln \left\{ \sum_i \pi_i p(\mathbf{t}_n \mid i) \right\} \tag{12}$$

So, we compute for a unit i, which of the three options gives the maximum L, and apply the changes (if any). Then we evaluate a new unit, and the process continues until there are no more units in the model. The units are evaluated in random order. When the evaluation process is finished, we run the EM algorithm with the training set to obtain the maximum likelihood parameters π_i, $\mathbf{W}_i$ and σ_i for the new values of q_i, if any change has been applied.

3.4 Unit Removal and Creation

In order to remove the unit j from the model, we renormalize the a priori probabilities of the remaining units:

$$\tilde{\pi}_i = \frac{\pi_i}{\sum_{k \neq j} \pi_k} \tag{13}$$

On the other hand, if we need to create a new unit, we select an existing unit i to split it into two. Each of the two 'daughter units' receives a half of the a priori probability π_i of their 'mother'. The parameters $\mathbf{W}_i$ and σ_i are inherited without changes, and the mean vectors are slightly perturbated in order to avoid two exactly identical units. Hence one of the daughters has a mean vector $\boldsymbol{\mu}_i - \varepsilon \mathbf{u}_1^i$ and the other $\boldsymbol{\mu}_i - \varepsilon \mathbf{u}_1^i$, where $\mathbf{u}_1^i$ is the eigenvector of the first principal direction of unit i and ε>0 is arbitrarily small.

In order to decide if remove, duplicate, or leave the unit i as it is, we use the criterion of the maximum log-likelihood L measured over the validation set (12). So, we compute for a unit i, which of the three options gives the maximum L, and apply the changes (if any). Then we evaluate a new unit, and the process continues until there are no more units in the model (the newly created units are not evaluated). Like before, the units are evaluated in random order. When the evaluation process is finished, we run the EM algorithm with the training set to obtain the maximum likelihood parameters π_i, $\mathbf{W}_i$ and σ_i for the modified model, if any deletion or creation has been applied.

3.5 Summary

The DMPPCA model can be summarized as follows:

1. Set the initial values of M and q_i, and apply the EM algorithm to obtain maximum likelihood parameters π_i, $\mathbf{W}_i$ and σ_i.
2. Evaluate possible changes in the size of vector bases q_i, as explained in subsections 3.1, 3.2 and 3.3. Apply the selected changes (if any).
3. If step 2 resulted in any change, apply the EM algorithm to obtain maximum likelihood parameters π_i, $\mathbf{W}_i$ and σ_i.
4. Evaluate possible removal and/or creation of units, as considered in subsection 3.4. Apply the selected changes (if any).
5. If step 4 resulted in any change, apply the EM algorithm to obtain maximum likelihood parameters π_i, $\mathbf{W}_i$ and σ_i.
6. If steps 2 or 4 produced any change, go to step 2. Otherwise, stop.

4 Computational Results

We have selected some databases to test the probability density estimation performance of our proposal. 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 [7]. The 'VizieR' database comes from the VizieR service [10], which is an information system for astronomical data. In particular, we have selected the Table 6 of the Complete near-infrared and optical photometric CDFS Catalog from Las Campanas Infrared Survey. We have extracted 22 numerical features from 10,000 stars. The 'Faces' database is the Yale Face Database B [4].

Table 1. ANLL (test set). The standard deviations for the 10 runs are shown in parentheses.

Database	DPCCA	VBMFA
Faces	–112.2755 (1.7426)	–24.6031 (0.8789)
Sixes	–90.0612 (12.2074)	–32.7079 (11.7853)
Twos	–65.5737 (6.1066)	–30.0245 (5.3913)
VizieR	5.5600 (0.6480)	19.7465 (8.6321)

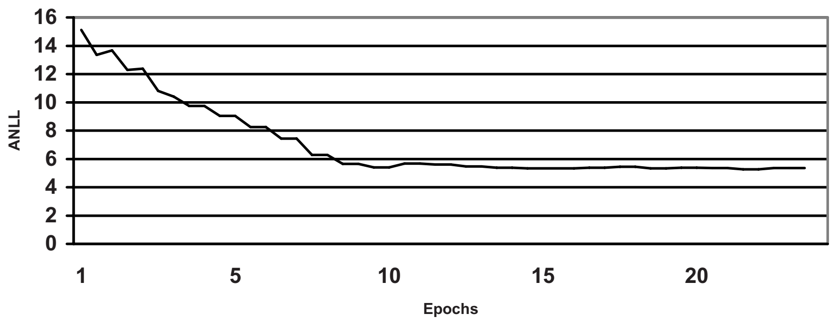

Fig. 1. Evolution of ANLL (validation set) during DPPCA training on 'VizieR' database

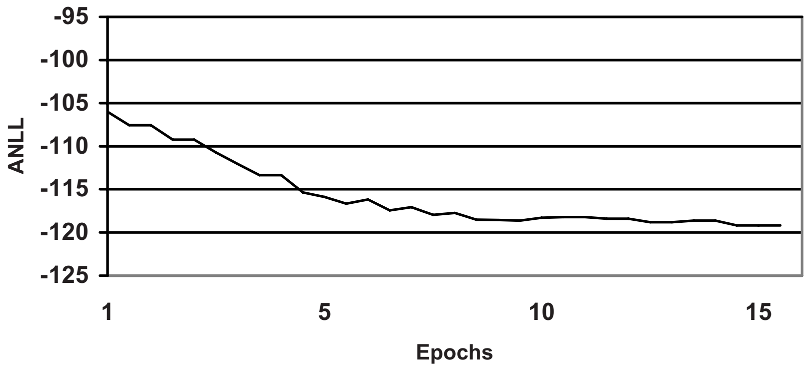

Fig. 2. Evolution of ANLL (validation set) during DPCCA training on 'Faces' database

We have selected the *Variational Bayesian Mixtures of Factor Analyzers* (VBMFA) by Ghahramani & Beal [5] to compare our model. The VBMFA has the same goal as ours, i.e., automatic selection of the number of mixing components and principal directions for probability distribution modeling. We have used the MATLAB implementation of the VBMFA model by Beal [1], with his selection of the model parameters.

For our DPPCA model we have started the simulations with M=10 and q_i=2 for all units, and a limit of 30 iterations of the loop of subsection 3.5 (epochs) has been set.

We have run a 10-fold cross-validation, with disjoint training, validation and test sets. The Average Negative Log-Likelihood (ANLL) computed on the test set has been used as the performance measure:

$$ANLL = -\frac{1}{P}\sum_{n=1}^{P}\log p(\mathbf{t}_n) \tag{14}$$

The results for the considered databases are presented in Table 1. We can see that the DMPCCA clearly outperforms VBMFA in all tests. Furthermore, the standard deviation data demonstrate that our proposal is also stable.

We can see in Figs. 1 and 2 the evolution of the ANLL computed on the validation set for a single simulation run with the 'VizieR' and 'Faces' database, respectively. Similar results were obtained with 'Sixes' and 'Twos' databases and therefore, omitted. We can see that the DMPCCA model shows a stable behaviour, with a first phase of fast decrease of ANLL, followed by a more stabilized phase. Please note that the simulations end before epoch 30, because there are no more changes to be made, that is, convergence is reached in less than 30 epochs.

5 Conclusions

We have proposed a a new Probabilistic PCA model which learns the maximum likelihood values of the number of mixing components and number of principal directions for each component. It features a cross-validation method to control the growth of the model and to avoid overfitting, without the need of additional assumptions on the characteristics of the input distribution. We have presented experimental results that show the performance of our proposal when compared with a well-known Variational Bayesian PCA approach.

Acknowledgements

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

References

1. Beal, M.J.: Software in Matlab.[online] Available at: http://www.cse.buffalo.edu/faculty/mbeal/software.html
2. Besse, P.: PCA stability and choice of dimensionality. Statistics and Probability Letters 13(5), 405–410 (1992)

3. Burden, R.L., Faires, D.: Numerical Analysis. Brooks/Cole Publishing, Pacific Grove, CA (2004)
4. Georghiades, A.S., Belhumeur, P.N., Kriegman, D.J.: From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose. IEEE Trans. Pattern Anal. Mach. Intelligence 23(6), 643–660 (2001)
5. Ghahramani, Z., Beal, M.J.: Variational Inference for Bayesian Mixtures of Factor Analysers. Advances in Neural Information Processing Systems 12, 449–455 (1999)
6. Kwon, O.-W., Chan, K., Lee, T.-W.: Speech Feature Analysis Using Variational Bayesian PCA. IEEE Signal Processing Letters 10(5), 137–140 (2003)
7. LeCun, Y., Cortes, C.: The MNIST Database of Handwritten Digits. In: Internet (November 2006) http://yann.lecun.com/exdb/mnist/
8. Oba, S., Sato, M., Ishii, S.: Prior Hyperparameters in Bayesian PCA. LNCS, vol. 2714, pp. 271–279. Springer, Heidelberg (2003)
9. Tipping, M.E., Bishop, C.M.: Mixtures of Probabilistic Principal Components Analyzers. Neural Computation 11, 443–482 (1999)
10. VizieR service [online]. (March 29, 2004) Available at: http://vizier.cfa.harvard.edu/viz-bin/VizieR