Available online at www.sciencedirect.com
SCIENCE DIRECT®
Pattern Recognition Letters 25 (2004) 1085–1095
Pattern Recognition Letters
www.elsevier.com/locate/patrec

Almost autonomous training of mixtures of principal component analyzers

Mohamed E.M. Musa [a,*], Dick de Ridder [b], Robert P.W. Duin [b], Volkan Atalay [c]

[a] Department of Computer Engineering, Cankaya University, Öğretmenler Caddesi No. 14, Balgat, Ankara, Turkey
[b] Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands
[c] Department of Computer Engineering, Middle East Technical University, TR-06531 Ankara, Turkey

Received 18 March 2003; received in revised form 21 January 2004
Available online 23 April 2004

Abstract

In recent years, a number of mixtures of local PCA models have been proposed. Most of these models require the user to set the number of submodels (local models) in the mixture and the dimensionality of the submodels (i.e., number of PC's) as well. To make the model free of these parameters, we propose a greedy expectation-maximization algorithm to find a suboptimal number of submodels. For a given retained variance ratio, the proposed algorithm estimates for each submodel the dimensionality that retains this given variability ratio. We test the proposed method on two different classification problems: handwritten digit recognition and 2-class ionosphere data classification. The results show that the proposed method has a good performance.

Keywords: PCA mixture model; EM algorithm; Regularization

1. Introduction

When the training data have different structures in different parts of the input space, fitting one global linear model can poorly represent the whole data. On the other hand, current global nonlinear models can be slow and inaccurate especially for high dimensional data (Kambhatla, 1995). A combination of local linear models (submodels) can quickly learn the structure of the data in local regions which consequently, offer faster and more accurate model fitting. Partitioning the training data set into smaller subsets may lead to the curse of dimensionality problem, as a training sample subset may not be sufficiently large for estimating the required set of parameters for a submodel. On the other hand, increasing the size of the training data set is not possible in many situations. Interestingly, in high dimensional spaces the data is often highly correlated. Therefore, by decorrelation methods we can reduce the data dimension and hence the number of parameters.

* Corresponding author.
E-mail address: memmusa@cankaya.edu.tr (M.E.M. Musa).

0167-8655/$ - see front matter
doi:10.1016/j.patrec.2004.03.019

Among these new local model methods that entail dimensionality reduction is the mixture of principal component analyzers (MPCA). A major enhancement in the history of this model was the introduction of the expectation-maximization (EM) training (Ghahramani et al., 1996; Hinton et al., 1997). All algorithms proposed before this enhancement have two separate processes; partitioning the data space by *hard clustering* methods, followed by fitting a PCA for each cluster. Another major enhancement proposed by Moghaddam (1997) and by Tipping and Bishop (1999) is the *probabilistic principal component analysis* (PPCA). By giving a probabilistic definition for PCA, the usage of the mixture model and *soft clustering* to define the mixture of PPCA (MPPCA) is straight forward. However, the MPPCA model still suffers from the following problems:

(1) There is no standard method to specify the optimal number of submodels.
(2) There is no standard method for the EM algorithm initialization, nor a standard method to help the algorithm to escape local maxima.
(3) There is no standard method to specify the optimal dimensionality for each submodel.

Previously, we have proposed and tested two algorithms for the first two problems mentioned above (Musa et al., 2001). In order to make training more autonomous, we propose here a method for solving the third problem and test the model performance on two classification tasks. The paper is organized as follows. Section 2 explains the probabilistic PCA. Section 3 describes our training method. Section 4 presents and discusses experimental results. We end the paper with some concluding remarks in Section 5.

2. Probabilistic principal component analyzer

Tipping and Bishop (1999) described a probabilistic formulation of PCA by considering it as a latent variable problem, in which a d-dimensional observed data vector $\boldsymbol{y}$ can be described in terms of an m-dimensional latent vector $\boldsymbol{x}$:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\mu} + \boldsymbol{w} \tag{1}$$

where $\boldsymbol{A}$ is a $d \times m$ matrix, $\boldsymbol{\mu} \in \mathbb{R}^d$ is the data mean and $\boldsymbol{w} \in \mathbb{R}^d$ is independent Gaussian noise with a diagonal covariance matrix $\sigma^2\boldsymbol{I}$. The probability of an observed data vector $\boldsymbol{y}$ is:

$$p(\boldsymbol{y}) = (2\pi)^{-d/2}|\boldsymbol{C}|^{-1/2} \times \exp\left(-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{C}^{-1}(\boldsymbol{y}-\boldsymbol{\mu})\right) \tag{2}$$

where $\boldsymbol{C}$ is the model covariance matrix given by:

$$\boldsymbol{C} = \boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} + \sigma^2\boldsymbol{I} \tag{3}$$

Moreover, Tipping and Bishop (1999) have shown that the maximum likelihood estimate for the noise variance σ^2 is given by:

$$\sigma^2 = \frac{1}{d-m}\sum_{k=m+1}^{d}\lambda_k, \tag{4}$$

where $\{\lambda_k\}_{m+1}^{d}$ are the $d-m$ smallest eigenvalues.

Using a Gaussian prior (zero mean, unit standard deviation) over the latent variables $\boldsymbol{x}$, an EM algorithm can be developed to find the parameters $\boldsymbol{\mu}$, σ^2 and $\boldsymbol{A}$. Furthermore, the algorithm can be quite naturally extended to find mixtures of such models, by estimating the set of parameters ($\boldsymbol{\mu}^i, \boldsymbol{A}^i$ and σ^i) for each submodel i and by re-estimating $p(\boldsymbol{y}|i)$ and the prior probabilities for each submodel, $p(i)$, at each step of the EM algorithm.

2.1. Submodel dimensionality versus variability

In conventional PCA techniques, there are two approaches for specifying principal submodel dimensionality: (1) to be specified by the user beforehand and (2) the user can specify a retained variance value α, and the system calculates the dimensionality that retains this variability ratio by finding the first m eigenvectors that satisfy the following inequality:

$$\frac{\sum_{i=1}^{m}\lambda_i}{\sum_{i=1}^{d}\lambda_i} > \alpha \tag{5}$$

where $\{\lambda_i\}_1^d$ are the eigenvalues sorted in descending order.

Almost all the early PCA mixture models described in the literature use the first approach. The

early proposals were geometrically based models and this could be the main reason behind the choice of the first approach. As Moghaddam (1997) and Tipping and Bishop (1999) extensions have shifted the model to be a probabilistically based model, we think that the choice in this issue should also be shifted to the second approach. Therefore, in our training scheme, we choose a value for α and adjust all submodel dimensionalities to retain this given variability ratio. Moreover, as one of our primary goals is to make training autonomous, the adoption of this retained variance approach will be of much help in this direction. Choosing one constant that allows different local dimensionalities with similar local variabilities is more advantageous over choosing same local dimensionality in all submodels that produces different local variabilities (Meinicke and Ritter, 2001). The simplest way to find α autonomously is by validation methods. This may be computationally intensive. However, exploitation of other speeding up methods, such as using some prior information to limit the choices, is possible.

3. Algorithms

As discussed in Section 1, in training an MPPCA model the user faces a number of choices: the number of submodels to look for, the number of dimensions per submodel to use and the way in which the EM algorithm is initialized, to avoid local minima. In this section, we describe some algorithms which alleviate these choices. First, in Section 3.1 a greedy algorithm is discussed, which can be used to determine a suboptimal number of variable dimensionality submodels. Next, Section 3.2 describes an EM initialization method which helps in preventing local minima. These algorithms lead to an almost parameter-free VD-MPPCA (VD for variable dimensionality) training method. The only parameter left is the amount of variance to retain.

3.1. Greedy EM for MPPCA training

Finding the optimal number of components in a mixture is a very difficult problem, which is not completely solved yet (McLachlan and Peel, 2000). However, recent theoretical developments (Cadez and Smyth, 2000; Li and Barron, 1999) have shown that the log-likelihood of finite mixture models is approximately concave as a function of k (number of submodels). Accordingly, we developed and tested a greedy EM for MPPCA training controlled by likelihood changes (Musa et al., 2001). In addition to the encouraging result of our previous experiments (Musa et al., 2001). Verbeek et al. (2003) have also investigated the problem of learning mixtures of Gaussians using a greedy EM algorithm. Their experimental results showed that their greedy EM outperforms standard EM. In our method, we adopt a simple yet effective and appealing strategy for finding a suboptimal number of submodels. We set a goal of building a mixture model that adequately represents almost all the training data points. To fulfil this goal, our greedy scheme increases the number of submodels in the mixture iteratively. To allow more data points to be well represented by the model, new submodels are generated from fractions of the training data points, which have the lowest probability in the current mixture. We use the membership (i.e., the posterior probability) to the current submodels as a criterion for this representation.

Algorithm 1 outlines our new greedy VD-MPPCA training algorithm. In step 2, the algorithm calculates a global principal submodel from the given training data set D by direct eigenvector method. The input value α is used to select the first m eigenvectors that retain a ratio α of the global variability. Step 6 collects the points that have high membership to the current submodels, $\cup N_i$, $i = 1, \ldots, k-1$. For each submodel i, N_i contains the $|D|/(k-1)$ points which have the highest posterior probability for this submodel. Step 7 calculates F as the complement of the set $\cup N_i$ in the training data set D, i.e., F contains the points that have the lowest membership to the current $k-1$ submodels. Using the given F data points, Algorithm 2 finds a new PPCA submodel and calculates its parameters. In step 9, the old submodels and the new one are fitted softly by the EM. Note that starting the greedy process by one submodel is not a requirement. For instance, if we are sure the number of submodels should be

greater than some constant $k_{\min}$, we can stop step 9 in the first $k_{\min}$ iterations to generate $k_{\min}$ hard clusters without fitting them by the EM algorithm.

Step 10 is responsible for deciding whether there is still significant change in likelihood or not. One way to do this is by specifying a threshold level. If the relative change in likelihood falls below this threshold, the loop terminates. Our investigation showed that this threshold value is not critical and even a high value such as 0.2 can produce acceptable classification results. In our experiments, we set this value to 0.1. An important caveat to be mentioned here is that the modeler should avoid too small threshold values, as for these the loop sometimes fails to terminate. However, the modeler can use a small value and add a control for the maximum number of submodels to terminate the loop when the threshold control fails.

Submodels change their shapes during EM iterations, i.e., their local variabilities. For this reason, after locating all submodels and terminate the greedy iteration, we re-calculate submodels' dimensionalities (step 11) and re-run the EM algorithm (step 12) to ensure soft fitting. With the addition of new submodels, normally old submodels lose some of their points and hence become more local. Therefore, the process of re-estimating submodels' dimensionalities in step 11 normally reduces them.

This algorithm has also a divide and conquer flavor. The algorithm contains two main parts. The first part is the greedy loop part (steps 4–10). The main goal of the greedy loop is finding the optimal number of submodels. The second part (steps 11 and 12) concerns finding the right local dimensionality for each submodel. It looks more logical to move the second part inside the greedy loop, i.e., to re-fit the dimensionality in each greedy step, as this will allow the dimensionality refitting to play its role in likelihood evaluation. However, bearing in mind that we are looking for approximate values for the parameters k (number of submodels) and m (local dimensionality), the given algorithm is simple, fast and efficient in finding these approximate values. In addition, our investigation of the training data shows no significant gain from re-fitting the dimensionality inside the greedy loop.

Algorithm 1 (*Greedy Training Algorithm*).

1: input D (training data set) and α (ratio of variance to retain)
2: estimate one VD-PPCA submodel's parameters ($\boldsymbol{\mu}^1, \boldsymbol{A}^1$ and σ^1), with a dimensionality that retains α of the variability in D
3: $k \leftarrow 1$
4: **repeat**
5: $k \leftarrow k + 1$
6: $\forall i, N_i = \{x : x$ among the $|D|/(k-1)$ points that have highest probabilities for submodel $i\}$
7: $F = D \setminus (\cup N_i)$
8: send the parameters $F, \{\boldsymbol{\mu}^i\}_1^{k-1}$, α to Algorithm 2 to estimate a new submodel's parameters ($\boldsymbol{\mu}^k, \boldsymbol{A}^k$ and σ^k)
9: fit the k submodels using the EM algorithm
10: **until** likelihood $(k) \cong$ likelihood$(k-1)$
11: re-fit all submodel dimensionalities to retain α of their variability
12: fit the model using the EM algorithm

3.2. Greedy hard clustering

An EM algorithm (soft clustering) initialized by a hard clustering algorithm is the state of the art for mixture model training. However, which hard clustering method to use with which EM version is still an open question. Intuitively, a suitable hard clustering method for a greedy EM might also be a greedy one. Dasgupta (1999) designed an algorithm that works greedily to find Gaussian Mixture means. Dasgupta's algorithm maps the data to a random subspace of size $\mathcal{O}(\log k)$ dimensions, where k is the number of Gaussians. In the reduced subspace the Gaussians become more spherical and the number of training data points with respect to the reduced subspace dimensionality is relatively high. Moreover, there is no danger of collapsing Gaussians together, as long as we project to a subspace of size $\mathcal{O}(\log k)$ (see Dasgupta, 1999 for a proof). Dasgupta's algorithm allocates the Gaussian centers one after the other. A point with the smallest radius r_x to enclose a specific number of points, p, in the reduced subspace, is considered a clue for a Gaussian center. To find the other Gaussian centers using

the same criterion, the high-density points of the recently found Gaussian are removed.

Our greedy hard clustering algorithm (Algorithm 2) consists of two main parts. The first part, steps 2–5, is based on Dasgupta's algorithm. The role of this part is to find the point with the smallest radius r_x in the reduced data subspace. To find a complete MPPCA submodel parameters (i.e., $\boldsymbol{\mu}^i, \boldsymbol{A}^i$ and σ^i), we run the nearest mean algorithm in the original data space to find S', the data points nearest to the new mean, $\boldsymbol{\mu}^*$. The means used by the nearest mean algorithm are the means of the existing submodels, $\{\boldsymbol{\mu}^i\}_1^k$, together with the newly found mean $\boldsymbol{\mu}^*$. From S' we calculate the initial estimate for the new MPPCA submodel parameters.

Algorithm 2 (*Initialization Algorithm: Greedy Hard Clustering Algorithm*).

1: input F (training data set), $\{\boldsymbol{\mu}^i\}_1^k$ (existing submodel means) and α
2: project the whole data set F into a random subspace of $\mathcal{O}(\log k)$ dimensions, let $S =$ projected data
3: set $p \leftarrow |S|/k$
4: $\forall \boldsymbol{x} \in S$, let $r_{\boldsymbol{x}}$ be the smallest radius such that there are $\geqslant p$ points within distance $r_{\boldsymbol{x}}$ from $\boldsymbol{x}$
5: let $\boldsymbol{\mu}^*$ be the point $\boldsymbol{x}$ with the lowest radius $r_{\boldsymbol{x}}$
6: run the nearest mean algorithm for $\boldsymbol{\mu}^*$ and the existing k means $\{\boldsymbol{\mu}^i\}_1^k$ in the original data space to find the closest points to $\boldsymbol{\mu}^*(S')$
7: from S' estimate one MPPCA submodel parameters, with a submodel dimensionality that retains α of the variability in S'

Note that the parameter k is determined by the current stage of Algorithm 1. Therefore the dimensionality of the projection subspace increases with k, and the number of points within radius r_x decreases with k.

4. Experiments

This section presents the results of our experiments on two classification problems.

We have designed four experiments:

(I) Training an MPPCA model with a fixed number of submodels of fixed dimensionality, using randomly initialized standard EM.
(II) Training an MPPCA model with a fixed number of submodels of fixed dimensionality, using standard EM initialized by Algorithm 2.
(III) Training an MPPCA model with a variable number of submodels of fixed dimensionality, using greedy EM.
(IV) Training a VD-MPPCA model with a variable number of submodels of variable dimensionality, using greedy EM.

Experiments II–IV are initialized by Algorithm 2. For the purpose of comparison, in experiment IV we set α to the average retained variance ratio found by the models generated by experiment III. All the experiments are repeated five times with differently drawn train and test sets. Experiment I is repeated three times for each pair of sets. During the training phase, only patterns of one class are presented to the model generator program, i.e., each class model is built separately. In Algorithm 2 we set the reduced subspace to the k-dimensional principal subspace.

4.1. Handwritten digit recognition

The data set used in this set of experiments was extracted from the well-known NIST handwritten digit database (Wilson, 1992). The original data set consisted of 128×128 pixel binary images. In preprocessing, these images were normalised for position, size, slant and stroke width, resulting in 16×16 pixel grey-value images (de Ridder et al., 1996). Furthermore, for the experiments described in this paper PCA was used on the entire data set to reduce the number of dimensions from 256 to 64. The resulting data set was used to construct training and testing sets. The data set has been classified before using a number of methods (de Ridder, 2001; Wilson, 1992). Table 1 gives an overview of the results obtained thus far on a training set of 1000 samples per class. 1000 patterns (images) per class (digit) have been used for training and 1000 patterns (images) per class

Table 1
Results for various classifiers on the NIST data set

Type	Classifier	Error (%)
Bayes plug-in	Nearest mean	15.88
	Linear	9.84
	Quadratic	4.70
Neural network	LeNotre	4.87
	LeNet	3.43
	LeCun	2.32
	1 Hidden layer at 256 units	2.44
	1 Hidden layer at 512 units	1.99
Support vector classifier	Polynomial, fifth degree	1.29
	Radial basis, $\sigma = 10$	1.38

(digit) have been used for testing. At first, in some of the experiments the EM algorithm did not converge. Therefore, the covariance matrix $\boldsymbol{C}$ was regularized by adding a small constant value, 0.01, to the parameter σ^2 in each iteration (see Eqs. (3) and (4)).

4.1.1. Results and discussion

Table 2 summarizes the first testing results. The error in this table is the average percentage of misclassified patterns in the test set. The results of this first set of experiments show that the performance of all methods of initialization and model fitting are nearly equal. This is curious. After inspecting the performance for each class separately, we noted that digit "1" has the worst performance (see Table 3). This is also curious. To investigate what caused this, we inspected the models found by each experiment. It became obvious that for some models, problems in estimating σ^2, the noise level, caused poor performance (see σ^2 for digit "1" and "9" in Table 3). The regularization value introduced to help the EM algorithm to converge, 0.01, seems to be too small.

The technique, adopted by PPCA, of approximating the average of the minor eigenvalues corresponding to the variance not explained by PCA, as a noise parameter σ^2 gives insight into this problem (see Eq. (4)). Eq. (3) shows that the diagonal components of the model covariance matrix $\boldsymbol{C}$ are dependent on σ^2 in the minor eigen-

Table 2
Test results, as average error percentage and standard deviation, for the four experiments on NIST handwritten digit data set

Exp.	Initialization	#Sub.	#Dim.	Error	#Sub.	#Dim.
I	Random	Fixed	Fixed	2.66 ± 0.21	10	10
II	Algorithm 2	Fixed	Fixed	2.67 ± 0.13	10	10
III	Algorithm 2	Variable	Fixed	2.60 ± 0.18	5.2	10
IV	Algorithm 2	Variable	Variable	2.60 ± 0.14	7.1	10

#Sub. and #Dim. are the average number of submodels and dimensionality per class. In all models, the estimated covariance matrices $\boldsymbol{C}$ are regularized by adding 0.01 times the identity matrix.

Table 3
The average σ^2 and test error for each of the 10 digit classes in the first set of experiments

Class	0	1	2	3	4	5	6	7	8	9
Avg. σ^2	0.18	0.06	0.25	0.22	0.20	0.24	0.16	0.14	0.22	0.13
Error %	1.99	4.99	0.79	2.79	1.48	1.75	1.92	3.00	2.36	4.10

In these experiments, the estimated covariance matrices $\boldsymbol{C}$ are regularized by adding 0.01 times the identity matrix.

vector directions. This structure makes the covariance matrix very sensitive to the actual value of σ^2. For very small values, $\boldsymbol{C}$ will become singular and the whole model becomes undefined. However, even when the matrix is non-singular and σ^2 is very small, the model will become prone to overfitting. This can be seen by realizing that for small σ^2, some elements of $\boldsymbol{C}^{-1}$ will become very large. Now, normally the image elements which are multiplied by the large values in $\boldsymbol{C}^{-1}$ (see Eq. (2)) will be very small, as their variance is very low. However, if in the data set an image occurs which has some noise present in pixel positions which normally have low variance, this noise will be blown up. It will have a large effect on the estimate of the probability of the image (Eq. (2)) and both training (specifically, the E-step in the EM algorithm) and recognition will suffer. This is the main reason for the bad performance of digit "1". Table 3 shows the average σ^2 (over all submodels) and the recognition error for each class for one of the experiment I. The table shows that digit "1" has the lowest value of σ^2. Fig. 1 shows some of digit "1" images which have been misclassified and one can easily note that most of the images have a generally reasonable shape, with some noise present. This gives the insight that σ^2 can be used as a clue for deciding on the optimal number of PCs for the PPCA model in general and most importantly that regularizing σ^2, e.g. by adding a larger regularization constant, could improve performance.

Fig. 1. Examples for the misclassified digit "1" images in the first set of experiments. In these experiments the estimated covariance matrices $\boldsymbol{C}$ are regularized by adding 0.01 times the identity matrix.

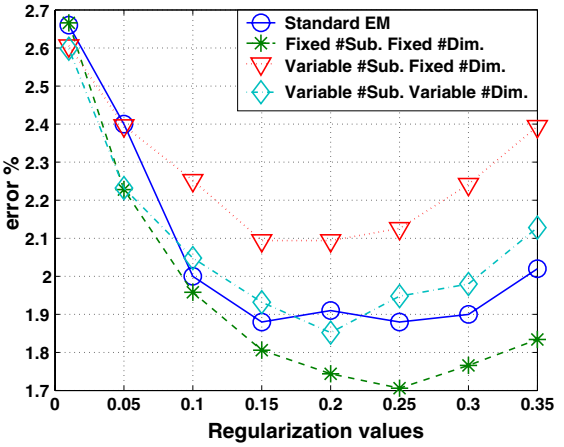

Fig. 2. Error (% of the test set classified incorrectly) as a function of the regularization value, for experiments I–IV.

To verify this, we re-ran all the experiments seven times using different regularization values. In the first run, each iteration of the EM algorithm, σ^2 was calculated by averaging the minor eigenvalues, as in normal PPCA, and adding a fixed regularization constant of 0.05. We then iteratively increased the regularization constant by 0.05 and repeated all experiments. The results of these experiments, shown in Fig. 2, show that an optimal regularization value is expected to be around 0.2. Detailed results for the experiments run with a regularization value of 0.2 are shown in Table 4. The experiments also show that, using Algorithm 2 as an initialization of the EM algorithm improves results. However, the most interesting result is the fact that experiment IV models perform as good as experiments I and II models, with a

Table 4
Test results for the four experiments using a regularization value of 0.20

Exp.	Initialization	#Sub.	#Dim.	Error	#Sub.	#Dim.
I	Random	Fixed	Fixed	1.91 ± 0.24	10	10
II	Algorithm 2	Fixed	Fixed	1.74 ± 0.18	10	10
III	Algorithm 2	Variable	Fixed	2.09 ± 0.35	4.5	10
IV	Algorithm 2	Variable	Variable	1.85 ± 0.12	7.2	10

smaller number of submodels. Experiment IV found 7.2 submodels for each class on average, where the models with fixed number of submodels (i.e., experiments I and II) used 10 for each class. It also worth noting that experiment III found even less submodels, 4.5, for each class on average, at only a small increase in test error.

In the previous set of experiments, for experiment IV we set α to the average retained variance ratio found by experiment III. This setting force experiment IV to generate submodels with average dimensionality 10. To have insight into the relation between the retained variance ratio (α), submodel dimensionality and classification performance, we re-ran experiment IV for 15 different α values. Fig. 3 shows the errors and average dimensionality for all classes for different α values. Fig. 3 reflects the important aspect that there exists a suboptimal α value (α_{sub}), at approximately 0.77. For α values less than α_{sub}, the classification performance is proportional to the submodel dimensionality. What is of real interest, is that the performance is approximately the same for α values greater than α_{sub} for both training and testing data. In fact there is a very slight improvement that can hardly justify the increase in dimensionality reflected in Fig. 3(b). Table 5 shows the detailed results for α_{sub}. Classes 1, 4 and 7 have the least number of submodels: 5, 8 and 8 respectively. On the other hand, classes 0, 6 and 8 have the maximum number of submodels, 11 submodels for each. This is striking, as the latter have more curvature in their shapes and their input space is more nonlinear structure, while the former are semi-straight lines, i.e., less nonlinearity. It shows that the algorithm is not ad hoc in model selection.

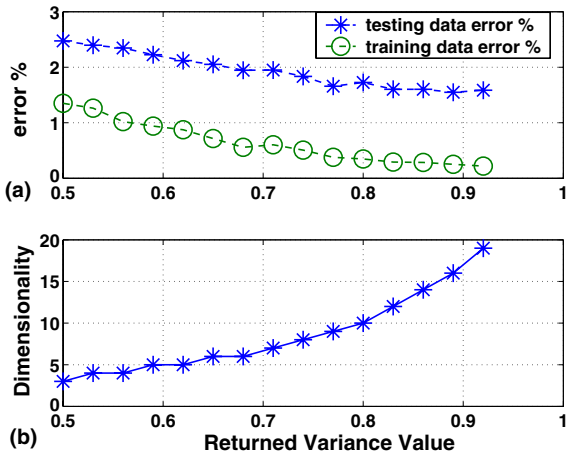

Fig. 3. (a) Error (% of test and training set classified incorrectly in the 15 experiments) as a function of the retained variance ratio i.e., α. (b) Average submodel dimensionality as a function of α.

4.2. Ionosphere data set

This data set was obtained from the UCI repository (donated by V. Sigillito from the applied Physics Laboratory in Johns Hopkins University, Blake and Merz (1998)). The data set consists of two classes of signals, "good" and "bad". Each signal instance has 34 attributes. In accordance with previous experiments, of which results are reported in Table 6, we have used 200 instances for training, 100 for each class. The testing set consists of 123 "good" and 24 "bad" instances. The previous results show that "good" signal recognition is much better than the "bad" one.

In this set of experiments we eliminated experiments I and II. We fixed the dimensionality to 5 and the number of submodels to 5 in experiments III. As before, for experiment IV, We set α to the variance retained by experiment III. we repeated the experiments 8 times for the same regularization

Table 5
This table summarize the suboptimal α_{sub} value result, 0.77

Class	0	1	2	3	4	5	6	7	8	9	Avg.
#Sub.	11	5	10	8	8	9	11	8	11	10	9
#Dim.	9	4	10	12	9	11	8	7	10	8	9
Error %	1.0	1.4	1.4	2.5	1.8	1.9	0.9	1.5	2.3	2.4	1.68 ± 0.17

The first row shows the average number of submodels for each class. The second row shows the average submodel dimensionality for each class. The third row shows the error for each class. The last column shows the average for all classes.

Table 6
Results for various classifiers on the ionosphere data set

Classifier	Error %
Linear perceptron	9.3
Nonlinear perceptron	8.0
Backpropagation ANN	4
Nearest neighbor	7.9
Quinlan's C4	6
IB3 (Aha and Kibler IJCAI-1989)	3.3

values set used in previous test. We ran the same second experiment to see the performance of the model for different α values.

4.2.1. Results and discussion

Figs. 4 and 5 summarize the results for the given set of experiments graphically. Since the model has its best average performance when the regularization value is 0.1, Table 7 gives a detailed result at this regularization value. The figures show that the performance in the 2 experiments is nearly the same except for the small regularization values, the performance of experiment IV is better. In experiment IV the model chooses to model class "good" with an average of 3 submodels of 3 dimensions each. As these values are optimal for modelling the "good" class it helps the model to be less vulnerable to noise than in experiment III where the number of submodels and dimensionality are higher.

The "bad" class is not as well structured as the "good" class. The model reflects this fact by choosing 6 submodels of 8 dimensions each on average. Estimating 6 submodels with average dimensionality 8 using only 100 patterns in the training data set may produce a model that suffers from the curse of dimensionality problem.

Fig. 5 shows the performance is almost the same, when α has a value which is greater than or equal to 0.65. In fact this is needed solely for the "bad" class, as a lower value suffices for class "good".

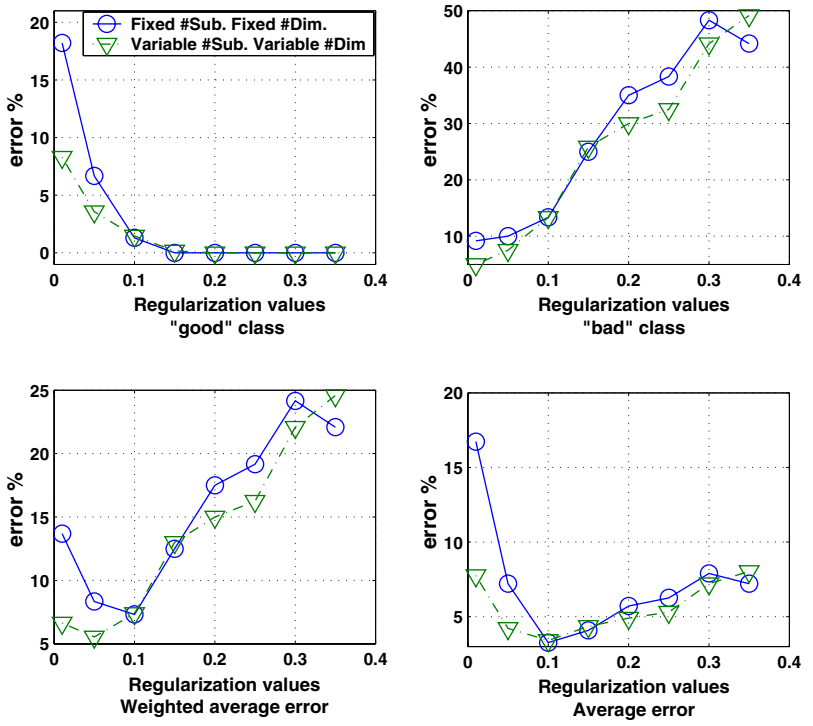

Fig. 4. Error (% of the test set classified incorrectly) as a function of the regularization value.

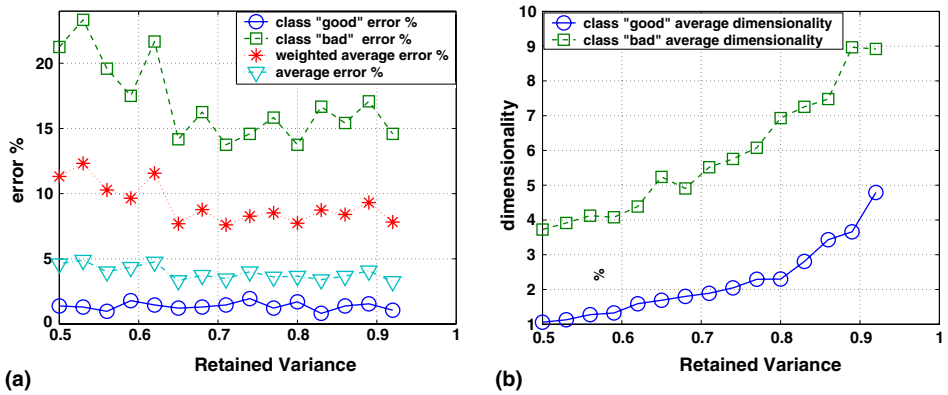

Fig. 5. (a) Error (% of test set classified incorrectly) as a function of the retained variance ratio i.e., α, (b) average submodel dimensionality as a function of α.

Table 7
Test results, as average error in % and standard deviation, for the two experiments on the ionosphere data set

Exp.	#Sub.	#Dim.	Error %	#Sub.		#Dim.	
				"good"	"bad"	"good"	"bad"
III	Fixed	Fixed	3.3 ± 1.1	5	5	5	5
IV	Variable	Variable	3.4 ± 1.4	3	6	3	8

#Sub. and #Dim. are the average number of submodels and dimensionality per class. For all models, the estimated covariance matrices $\boldsymbol{C}$ are regularized by a value of 0.1.

5. Conclusion

We designed a greedy EM algorithm for training a mixture of probabilistic principal component analyzers. According to a given retained variance ratio, α, the algorithm determines for each submodel the dimensionality that retains a ratio α of the local variability.

We applied the model to classification problems. The results show that the retained variance is a suitable guidance for the process of searching for the optimal submodel dimensionality. While increasing the number of submodels iteratively is effective in searching for the optimal number of submodels, it is also helpful in avoiding the inherent problem of overfitting.

As the discussion reflects, there is an optimal or suboptimal α that should be estimated or hypothesized first. This suboptimal value could be reached by validation methods. However, finding a direct method will speed up the model. Moreover, some researchers suggested solving this problem (i.e., dimensionality) by Bayesian methods (Minka, 2000). In our future work, we plan to compare our method for dimensionality selection with the Bayesian based method.

In the second problem, which is a two class classification problem, one class has much higher error rates. It seems that exploiting the natural rejection capability of the density models may leverage the performance in such cases. This is also a subject of future work. Using maximum likelihood as a sole control for the number of submodel complexity may not be enough. Especially for classification, this complexity control mechanism may generate excess submodels. Therefore, as future work, we plan to support our model by using a penalized maximum likelihood or an information theoretic measurement for the optimal number of submodels selection (McLachlan and Peel, 2000).

References

Blake, C., Merz, C., 1998. UCI repository of machine learning databases. Department of Information and Computer Science, University of California.

Cadez, I., Smyth, P., 2000. On model selection and concavity for finite mixture models. In: Proc. Int. Symp. on Information Theory (ISIT).

Dasgupta, S., 1999. Learning mixtures of Gaussians. In: Proc. IEEE Symp. on Foundation of Computer Science.

de Ridder, D., Hoekstra, A., Duin, R., 1996. Feature extraction in shared weights neural networks. In: Proc. 2nd Annual Conference of the Advanced School for Computing and Image Processing, Delft, Netherlands, pp. 289–294.

de Ridder, D., 2001. Adaptive methods of image processing. Doc. dissertation, Faculty of Applied Science, Delft University of Technology.

Ghahramani, Z., Hinton, G., 1996. The EM Algorithm for mixtures of factor analyzers. Technical Report, CRG-TR-96-1. University of Toronto.

Hinton, G., Dayan, P., Revow, M., 1997. Modeling the manifolds of images of handwritten digits. IEEE Trans. Neural Networks 10 (3), 65–74.

Kambhatla, N., 1995. Local models and Gaussian mixture models for statistical data processing. Doc. dissertation, Oregon Graduate Inst. of Science & Tech.

Li, J., Barron, A., 1999. Mixture density estimation. NIPS 1999.

McLachlan, G., Peel, D., 2000. Finite Mixtures Models. John Wiley & Sons Inc.

Minka T., 2000. Automatic choice of dimensionality for PCA. NIPS 2000.

Meinicke, P., Ritter, H., 2001. Resolution-based complexity control for Gaussian mixture models. Neural Comp. 13 (2), 453–475.

Moghaddam, B., Pentland, A., 1997. Probabilistic visual learning for object representation. PAMI 19 (7), 696–710.

Musa, M., Duin, R., de Ridder, D., 2001. An enhanced EM algorithm for mixture of probabilistic principal component analysis. ICANN 2001 Workshop on Kernel & Subspace Methods for Computer Vision.

Tipping, M., Bishop, C., 1999. Mixtures of principal component analyzers. Neural Comp. 11 (2), 443–482.

Verbeek, J., Vlassis, N., Krose, B., 2003. Efficient greedy learning of Gaussian mixture models. Neural Comp. 15 (2), 469–485.

Wilson, C., Garris, M., 1992. Handprinted character database. National Institute of Standards and Technology; Advanced Systems Division.