Incremental Mixtures of Factor Analysers

Albert Ali Salah, Ethem Alpaydın
Department of Computer Engineering
Boğaziçi University, 34342 Bebek, Istanbul, Turkey
{salah, alpaydin}@boun.edu.tr

Abstract

A mixture of factor analyzer is a semiparametric density estimator that performs clustering and dimensionality reduction in each cluster (component) simultaneously. It performs nonlinear dimensionality reduction by modeling the density as a mixture of local linear models. The approach can be used for classification by modeling each class-conditional density using a mixture model and the complete data is then a mixture of mixtures. We propose an incremental mixture of factor analysis algorithm where the number of components (local models) in the mixture and the number of factors in each component (local dimensionality) are determined adaptively. Our results on different pattern classification tasks prove the utility of our approach and indicate that our algorithms find a good trade-off between model complexity and accuracy.[1]

1 Introduction

A *mixture model* is written as

$$p(\boldsymbol{x}) = \sum_{j=1}^{J} p(\boldsymbol{x}|\mathcal{G}_j)P(\mathcal{G}_j) \quad (1)$$

where $\mathcal{G}_j$ stand for the components, $P(\mathcal{G}_j)$ is the prior probability, and $p(\boldsymbol{x}|\mathcal{G}_j)$ is the probability that the data point is generated by component j. In classification, each class-conditional density is written as a separate mixture model and the input is then a *mixture of mixtures* (MoM):

$$\begin{aligned} p(\boldsymbol{x}|\mathcal{C}_i) &= \sum_{j=1}^{J_i} p(\boldsymbol{x}|\mathcal{G}_{ij})P(\mathcal{G}_{ij}) && (2) \\ p(\boldsymbol{x}) &= \sum_{i=1}^{N} p(\boldsymbol{x}|\mathcal{C}_i)P(\mathcal{C}_i) && (3) \end{aligned}$$

[1]This work has been supported by the Turkish Academy of Sciences in the framework of the Young Scientist Award Program (EA-TÜBA-GEBIP/2001-1-1) and Boğaziçi University Scientific Research Projects 02A104D and 03K120250.

During testing, all $p(x|\mathcal{C}_i)$ are calculated and the class with the highest posterior is chosen using Bayes' rule: $P(\mathcal{C}_i|x) = P(\mathcal{C}_i)p(\boldsymbol{x}|\mathcal{C}_i)/p(\boldsymbol{x})$.

In a *mixture of Gaussians* (MoG), each component of Eq. 1 is a Gaussian: $p(x|\mathcal{G}_j) \sim \mathcal{N}(\mu_j, \boldsymbol{\Sigma}_j)$. Similarly, a *mixture of mixtures of Gaussians* (MoMoG) assumes that each class is modeled with a number of Gaussian components (Eq. 2). With complete covariance matrices $\boldsymbol{\Sigma}_j$, the number of parameters can be very large and this may cause overfitting. With tied (shared) covariances, the number of parameters decreases but this assumes the same input distribution in different components and may be a source of bias. Another approach is to constrain the covariances to be diagonal or spherical, but these discard valuable correlation information.

Decreasing the number of parameters while still modeling the covariances is possible with a *factor analysis* (FA) model, where $\boldsymbol{\Sigma}_j = \boldsymbol{\Lambda}_j\boldsymbol{\Lambda}_j^T + \Psi$, and using either Eq. 1 or Eq. 2 we have a *mixture of factor analyzers* (MoFA) or a *mixture of mixtures of factor analyzers* (MoMoFA), respectively. This corresponds to assuming that a small number of low-dimensional latent variables (factors) $\boldsymbol{z}$ cause the correlation in component j: $\boldsymbol{x} - \boldsymbol{\mu}_j = \boldsymbol{\Lambda}_j\boldsymbol{z} + \boldsymbol{\epsilon}_j$

$\boldsymbol{\Lambda}_j$ is called the *factor loading matrix* (for component j) and shows the dependence of data points to each factor. $\boldsymbol{\epsilon}_j \sim \mathcal{N}(0, \Psi)$ is the Gaussian noise, where Ψ is a diagonal matrix, interpreted as sensor noise common to all components. When $\boldsymbol{x}$ is d-dimensional, $\boldsymbol{\Sigma}_j$ is $d \times d$, whereas with $p < d$ factors, $\boldsymbol{\Lambda}_j$ is $d \times p$.

Given a training set, the maximum likelihood estimates can be calculated using the Expectation-Maximization (EM) algorithm, which simultaneously places the components in the input space ($\boldsymbol{\mu}_j$) and also finds the factors in each component, performing dimensionality reduction in each component, ($\boldsymbol{\Lambda}_j$) [3, 5, 8]. However, this requires that the number of components and the factors in each component be specified in advance. In this paper, we propose an *incremental algorithm* where components and factors are added iteratively as needed, without requiring them to be specified in advance, thereby better matching the complexity of the mixture model to that of the data. The algorithm is discussed in section 2, simulation results are given in section 3, and we conclude in section 4.

2 Incremental Mixture of Factor Analysers

We introduce an incremental MoFA algorithm that starts with a one-factor, one-component mixture and proceeds by adding new factors or new components until some stopping condition is satisfied. Our *Incremental Mixture of Factor Analysers* (IMoFA) algorithm relies on fast heuristic metrics to single out one component for splitting and another for factor addition. The pseudocode of the algorithm is given in Fig. 1. To check complexity and alleviate overfitting, the split and factor addition are tested on a validation set, separate from the training set over which the parameters are calculated, and the action that causes the greatest increase in the validation likelihood is chosen. The algorithm terminates when there is no improvement on the validation likelihood.

```
algorithm IMoFA(train, validation)
    [Λ, μ, Ψ] ← train a 1-component, 1-factor model
    oldLikelihood ← -Infinity
    /*Likelihoods are calculated on validation set*/
    newLikelihood ← likelihood(Λ, μ, Ψ)
    while newLikelihood > oldLikelihood
    /*Perform a single split*/
        x ← Select a component for splitting
        [Λ1, μ1, Ψ1, π1] ← EM(split x).
        actionL(1) ← likelihood(Λ1, μ1, Ψ1, π1)
    /*Perform a single factor addition*/
        y ← Select a component to add a factor
        [Λ2, μ2, Ψ2, π2] ← EM(add factor to y).
        actionL(2) ← likelihood(Λ2, μ2, Ψ2, π2)
    /*Select the best action*/
        z ← max(action(L1),action(L2))
    /*Update the parameters*/
        [Λ, μ, Ψ, π] ← [Λz, μz, Ψz, πz]
        oldLikelihood ← newLikelihood
        newLikelihood ← likelihood(Λ, μ, Ψ, π)
    end
    return [Λ, μ, Ψ, π]
end
```

Figure 1. IMoFA Algorithm

IMoFA algorithm given in Fig. 1 is unsupervised. In the case of a classification problem, we can use it to fit a MoFA to examples of each class seperately (Eq.2). This is a *likelihood-based* approach (IMoFA-L) where the likelihood of each class is maximized separately. We have also implemented a *discriminative-based* variant of this algorithm (IMoFA-A) that adds factors and components while monitoring classification accuracy on the validation set, instead of the likelihood. All class models are needed to calculate accuracy, thus they can no longer be trained separately. We start with one-factor, one-component models for each class, and at each iteration add a component/factor to the class having the highest Type 2 error (accepted samples of other classes) and check for improvement, except that we use accuracy on the validation set, instead of the likelihood.

For the incremental algorithm to be accurate and efficient, the candidate component and factor should be placed and initialized intelligently.

2.1 Component Addition

Adding a new component by splitting an existing one involves two decisions: which component to split, and how to split it. Our approach is to split the component that looks least likely to a Gaussian, i.e., unimodal. We have tested several likelihood and kurtosis-based methods to test the non-Gaussianity of the data under each component and selected a multivariate kurtosis metric [6]. For a multinormal distribution, the multivariate kurtosis takes the value $\beta_{2,d} = d(d+2)$, and if the underlying population is multivariate normal with mean $\boldsymbol{\mu}$, the sample counterpart of $\beta_{2,d}$, namely $b_{2,d}$, has the following asymptotic distribution as the number of samples $N \to \infty$:

$$\frac{b_{2,d} - d(d+2)}{\left[\frac{8d(d+2)}{N}\right]^{\frac{1}{2}}} \sim \mathcal{N}(0,1) \tag{4}$$

with

$$b_{2,d} = \frac{1}{N}\sum_{t=1}^{N}\left[(\boldsymbol{x}^t - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}^t - \boldsymbol{\mu})\right]^2 \tag{5}$$

We adapt this metric to the mixture model by using the"soft count" $h_j^t \equiv E[\mathcal{G}_j|\boldsymbol{x}^t]$:

$$\gamma_j = \{b_{2,d}^j - d(d+2)\}\left[\frac{8d(d+2)}{\sum_{t=1}^N h_j^t}\right]^{-\frac{1}{2}} \tag{6}$$

$$b_{2,d}^j = \frac{1}{\sum_{l=1}^N h_j^l}\sum_{t=1}^{N} h_j^t\left[(\boldsymbol{x}^t - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}^t - \boldsymbol{\mu}_j)\right]^2 \tag{7}$$

The component with greatest γ_j is the one that looks least like a unimodal multivariate normal, and is selected for splitting. We locate its mean and go one standard deviation in either way along the principal direction of the component. Two points obtained this way are used to initialize EM. Our simulations showed that this method works better than random splits or initializing from potential outliers.

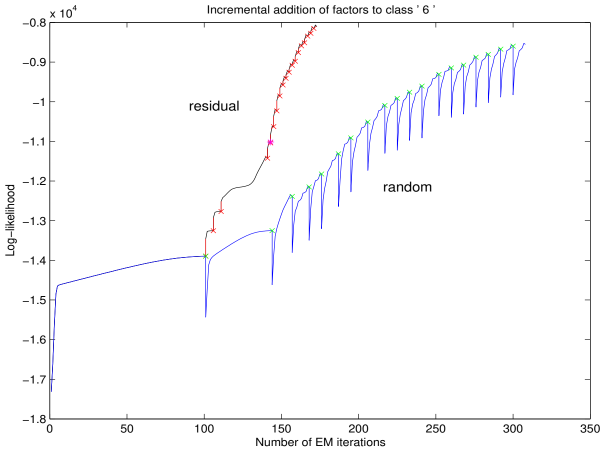

Figure 2. Residual factor addition converges faster than random addition. '$\times$' indicate that a new factor is added.

2.2 Factor Addition

The basic factor analysis uses the factor loading matrix $\boldsymbol{\Lambda}$ to model all the covariances, and part of the variances. Hence a natural way to select the component to add a factor is to look at the difference between the sample covariance and modeled covariance for each component, and consider the one with greatest difference for factor addition. Once the component is selected, the new factor can be randomly initialized, in which case the EM procedure is expected to take it to a local optimum. We propose a method of initializing the new factor that we call *residual factor addition* that works better (see Fig.2).

Given a component $\mathcal{G}_j$ and a data point $\boldsymbol{x}^t$, we can find the expected value of $\boldsymbol{z}^t$. Actually, $E[\boldsymbol{z}^t|\boldsymbol{x}^t, \mathcal{G}_j]$ is the p-dimensional data point if we intend to employ the factor analyser as a dimensionality reduction method. If we wish to restore the original data point from the dimensionality-reduced $\boldsymbol{z}^t$, we will multiply it with the factor loading matrix: $\tilde{\boldsymbol{x}}_j^t = \boldsymbol{\Lambda}_j E[\boldsymbol{z}^t|\boldsymbol{x}^t, \mathcal{G}_j]$

However, unless we use the complete eigenvectors in $\boldsymbol{\Sigma}_j$ (or some equivalent basis), $\tilde{\boldsymbol{x}}_j^t$ will not exactly coincide with $\boldsymbol{x}^t$. The residual error is: $\boldsymbol{e}_j^t = \boldsymbol{x}^t - \tilde{\boldsymbol{x}}_j^t$. When we add the new factor column $\boldsymbol{\Lambda}_{j,p+1}$ to $\boldsymbol{\Lambda}_j$, it will make a contribution to the re-estimated value of each $\boldsymbol{x}^t$ in the direction of $\boldsymbol{\Lambda}_{j,p+1}$ with the magnitude of z_{p+1}^t. The residual factor addition aims at minimizing these residuals by selecting $\boldsymbol{\Lambda}_{j,p+1}$ to be the principal direction (the eigenvector with the largest eigenvalue) of the residual vectors. This new factor is used in bootstrapping the EM procedure.

3 Simulation Results

We tested our algorithm with eight datasets. Pendigits, Optdigits, Waveform and Segment are taken from UCI Repository [1]. We used the ORL face data for male-female classification [7] and a pre-processed 10-class selection of Vistex texture database [9]. We have the Yeast microarray gene expression data [2], and a 20-class phoneme data distributed with LVQ package of the Helsinki University of Technology [4]. For each of those datasets, one third of the training set is used for validation (See Table 1 for details).

Table 1. Datasets

Dataset	Training	Test	Dimensions	Classes
PEN	5,494	3,498	16	10
OPT	2,880	1,797	64	10
ORL	400	cv10	256	2
VIS	2,700	910	169	10
YEAST	208	cv7	79	5
LVQ	1,929	1,929	20	16
WAVE	300	4,700	21	3
SEG	700	1,610	14	7

On these datasets, we tested MoG, MoFA, MoMoG, MoMoFA and IMoFA algorithms with different parameters. EM runs are initialized with k-means clustering. We report the configuration that produces the highest likelihood on the validation set (See Table 2). The average classification accuracy and number of parameters are reported for 10 independent runs on the test sets. We used 10-fold cross-validation in ORL, and 7-fold cross-validation on Yeast.

In MoG simulations, the best solutions were with unrestricted (full) covariances. MoFA and MoMoFA models were tested with $1-10$ factors per component, and again the highest-likelihood solutions were produced by the model with a maximum number of parameters. For a fixed number of components in MoMoFA, the increase in likelihood is approximately linear with the number fo factors. However in MoMoG models, the unrestricted cases are never taken as a result of overlearning; either diagonal or spherical covariances (but with maximum number of components per class) are selected. The preference of unrestricted MoG over other MoG options and the general preference of MoMoFA models to MoMoG models show that the correlation information is too valuable to discard.

Increasing the number of parameters in IMoFA tends to increase the likelihood, but these changes can be due to samples that lie closest to the component means, where small changes have the most effect. This is a wasteful increase for classification purposes. By monitoring classification accuracy, IMoFA-A finds solutions with lower likelihood but with equal accuracy for much fewer parameters.

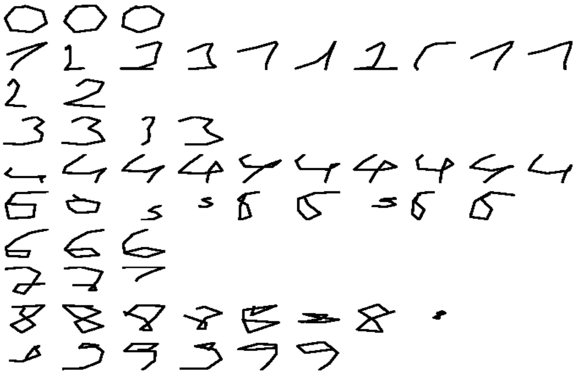

Figure 3. Components in IMoFA-L for Pendigits. Different styles prompt introducing of more components.

4 Conclusions

Our simulation results indicate that the incremental mixture of factor analysers is capable of finding reasonable points in the complexity vs. accuracy curve. Compared to fixed-parameter MoG and MoFA models, IMoFA-L finds high-likelihood solutions with a moderate number of parameters. Similarly, IMoFA-A finds highly accurate solutions with a moderate number of parameters.

Modeling a class with more than one component is useful if samples of the class are generated by different processes, like the allomorphs of digits. The incremental algorithm allows automatic allocation of more components to classes with many alternative structures (See Fig. 3). In a dataset where this is not the case, increasing the number of components quickly leads to overlearning, as the components impose superficial clusters on the data. Our proposed incremental algorithm checks for improvement after each component addition on a separate validation set and does not permit such overfitting.

The number of factors is a parameters of different nature; it can be increased up to the number of dimensions in order to model the sample covariance better and has a smaller overlearning effect. The benefit of constraining it is mostly to keep the time and space complexity low. Our simulations show that when the incremental algorithm assigns a small number of components to a class, it usually assigns a greater number of factors to those components.

Our proposed incremental algorithms tailor the mixture structure, i.e., the components and the factors in each component, automatically to the input density. The mixture model incrementally fits a nonlinear model to the data as a mixture of locally linear models. The component means and factors tell us respectively the positions and important dimensions of correlation in each local model, thereby allowing knowledge extraction and increased interpretability.

Table 2. Simulation Results

Method		Accuracy	Par.		Accuracy	Par.
MoG		98.50 ± 0.0	1529		95.76 ± 0.0	18909
MoMoG	**P**	98.72 ± 0.1	4589	**O**	92.98 ± 0.5	6049
MoFA	**E**	95.20 ± 0.0	1625	**P**	96.92 ± 0.0	6669
MoMoFA	**N**	99.35 ± 0.2	8865	**T**	97.94 ± 0.4	30109
IMoFA-L		97.93 ± 0.4	2699		92.92 ± 0.6	7629
IMoFA-A		97.63 ± 0.7	1106		93.84 ± 0.2	3819
MoG		87.25 ± 4.3	66305		35.60 ± 0.0	145349
MoMoG	**O**	98.25 ± 1.7	5129	**V**	73.82 ± 1.1	8549
MoFA	**R**	99.00 ± 1.3	5889	**I**	62.53 ± 0.0	18768
MoMoFA	**L**	99.25 ± 1.2	28425	**S**	58.29 ± 1.2	93168
IMoFA-L		98.50 ± 1.3	8068		70.40 ± 1.7	29493
IMoFA-A		97.75 ± 0.8	2407		70.71 ± 1.2	12688
MoG	**Y**	63.78 ± 4.3	16199		88.96 ± 0.0	3695
MoMoG	**E**	93.37 ± 6.0	2384	**L**	86.15 ± 1.1	14783
MoFA	**A**	92.35 ± 6.0	2453	**V**	87.56 ± 0.0	3555
MoMoFA	**S**	93.88 ± 7.4	7203	**Q**	89.24 ± 0.6	17699
IMoFA-L	**T**	91.33 ± 6.8	3151		89.44 ± 0.6	1749
IMoFA-A		94.39 ± 4.5	1321		90.31 ± 0.5	1512
MoG		77.89 ± 0.0	758		88.57 ± 0.0	839
MoMoG	**W**	74.57 ± 1.7	3035	**S**	89.65 ± 2.1	3359
MoFA	**A**	75.85 ± 0.0	716	**E**	65.78 ± 0.0	1098
MoMoFA	**V**	73.53 ± 0.8	3500	**G**	88.89 ± 1.3	419
IMoFA-L	**E**	82.55 ± 0.5	195		85.13 ± 2.4	603
IMoFA-A		80.92 ± 1.5	466		90.29 ± 1.4	568

References

[1] Blake, C.L., and C.J. Mertz, *UCI repository of machine learning databases*, http://www.ics.uci.edu/∼mlearn/ MLRepository.html, UCI: Dept. of Info. and Comp. Sci., 1998.

[2] Brown, M.P.S. et al., “Knowledge-based analysis of microarray gene expression data using support vector machines,” *Proc. National Academy of Sciences*, 97:262-267, 2000.

[3] Ghahramani, Z., and G.E. Hinton, “The EM algorithm for mixtures of factor analyzers,” Technical Report CRG-TR-96-1, University of Toronto, (revised), 1997.

[4] Kohonen, T., J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola, *LVQ-PAK*, Helsinki Univ. of Technology, 1995.

[5] McLachlan, G.J., and D. Peel, *Finite Mixture Models,* Wiley-Interscience, 2000.

[6] Mardia K.V., J.T. Kent, and S.M. Bibby, *Multivariate Analysis*, Academic Press, 1979.

[7] The Olivetti Research Laboratory database of faces, http://www.cam-orl.co.uk/facedatabase.html, 1994.

[8] Tipping, M., and C. Bishop, “Mixtures of probabilistic principal component analysers,” *Neural Computation*, vol.11(2), pp.443-482, 1999.

[9] MIT Media Lab Vistex database, http://www.white. media.mit.edu/vismod/imagery/VisionTexture/vistex.html,2002.