A Comparison of Mixture Models for Density Estimation

Perry Moerland
IDIAP, CP 592, Martigny, Switzerland
E-mail: Perry.Moerland@idiap.ch

Abstract. Gaussian mixture models (GMMs) are a popular tool for density estimation. However, these models are limited by the fact that they either impose strong constraints on the covariance matrices of the component densities or no constraints at all. This paper presents an experimental comparison of GMMs and the recently introduced mixtures of linear latent variable models. It is shown that the latter models are a more flexible alternative for GMMs and often lead to improved results.

1 Introduction

Density estimation is an important issue in machine learning with applications to data visualization, modelling of class-conditional densities, and initialization of radial basis function networks. In this paper, we focus on semi-parametric density estimation based on *mixture* distributions. A well-known approach is the use of Gaussian mixture models (GMMs, for example [1]). The use of a GMM with *full* covariance matrices leads to a huge number of parameters for a high-dimensional input space and presents the risk of over-fitting. Therefore, the covariance matrices are often constrained to be *spherical*, with a single parameter for the whole covariance structure, or *diagonal*. The latter constraint leads to a model in which the axes of the Gaussians are aligned with the data axes and which does not capture correlation amongst the variables. Thus, each of these parameterizations has its disadvantages. A compromise between these extremes can be found in the recently introduced mixture of *latent variable* models [5, 11] which form a mixture of *constrained* Gaussians. The advantage of using mixtures of latent variable models is that one can avoid the constraint of aligned axes (thus capturing correlations) without needing a full covariance matrix. This can be done by using the freedom we have in choosing the dimension of the so-called latent space: the covariance matrices of the Gaussians are specified and controlled through a mapping from this latent space to the data space. GMMs and mixtures of latent variable models are described in section 2.

The contribution of this paper, is an experimental comparison of GMMs and mixtures of latent variable models for density estimation on two artificial and eight real-world data sets, in section 3. This is the condensed version of a more elaborate technical report [7].

2 Mixture Models

A mixture model is defined as a linear combination of m component densities $p_j(\mathbf{x})$:

$$p(\mathbf{x}) = \sum_{j=1}^{m} \alpha_j p_j(\mathbf{x}), \qquad (1)$$

where the α_j are the mixing coefficients which are non-negative and sum to one.

2.1 Gaussian Mixture Models

Gaussian mixture models are a standard tool for density estimation and are described in many textbooks (for example, [1]). A GMM is defined as a mixture model (1) with component distributions that are Gaussian with a covariance matrix $\boldsymbol{\Sigma}_j$ that is chosen to be full, diagonal or spherical (as stated in the introduction) and mean $\boldsymbol{\mu}_j$:

$$p_j(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j),$$

The parameters of a GMM can be determined in a maximum likelihood framework by the EM algorithm of which we give a short outline here in the case of a full covariance matrix. The negative log-likelihood of a GMM for a data set $\{\mathbf{x}^n\}$ is:

$$E = -\sum_{n} \ln \sum_{j=1}^{m} \alpha_j p_j(\mathbf{x}^n),$$

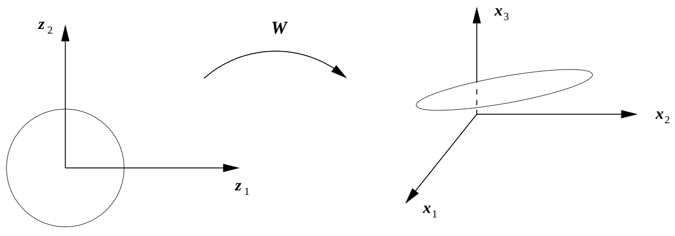

Figure 1: A generative model from a latent space of dimension 2 to a data space of dimension 3.

which is the error function that needs to be minimized. The values for the parameters $\boldsymbol{\mu}_j$, $\boldsymbol{\Sigma}_j$, and α_j can be found by iteratively performing the two steps of the EM algorithm which guarantees convergence to a local minimum [1]:

1. E-step $\rightarrow$ Estimation of the posteriors:
$$h_j(\mathbf{x}^n) = \frac{\alpha_j p_j(\mathbf{x}^n)}{\sum\limits_{i=1}^{m} \alpha_i p_i(\mathbf{x}^n)}.$$

2. M-step $\rightarrow$ Re-estimation of the parameters of the GMM (new estimations are denoted with a prime):
$$\alpha'_j = \frac{1}{N}\sum_n h_j(\mathbf{x}^n)$$
$$\boldsymbol{\mu}'_j = \frac{\sum_n h_j(\mathbf{x}^n)\mathbf{x}^n}{\sum_n h_j(\mathbf{x}^n)}$$
$$\boldsymbol{\Sigma}'_j = \frac{\sum_n h_j(\mathbf{x}^n)(\mathbf{x}^n - \boldsymbol{\mu}'_j)(\mathbf{x}^n - \boldsymbol{\mu}'_j)^T}{\sum_n h_j(\mathbf{x}^n)}.$$

With full covariance matrices, each EM step requires $O(md^2n)$ operations, where n is the number of vectors in data space and d is the dimension of the data space. Spherical or diagonal covariance matrices are often used to limit the computational complexity to $O(mdn)$ and to restrict the amount of data needed for reliable estimation.

In the experiments with a full covariance matrix described in section 3, Bayesian regularization is used to avoid singular matrices as proposed by Ormoneit and Tresp [8]. This requires only some additional factors in the M-step update of the covariance matrix and is numerically more stable.

2.2 Latent Variable Models

A latent variable model relates a d-dimensional observed data vector $\mathbf{x}$ to a l-dimensional ($l < d$) latent vector $\mathbf{z}$ by defining a noise model and a prior on the distribution of the latent variables. In this paper, we are interested in linear latent variable models:

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \varepsilon. \qquad (2)$$

The idea behind the model is illustrated in figure 1. The prior distribution of the latent variables is a simple Gaussian distribution $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ (left-hand part of figure 1) over the latent space. The first two terms on the right-hand side of (2) are the mean $\boldsymbol{\mu}$, and the $(d \times l)$ generative matrix $\mathbf{W}$, that maps the latent space into the data space. Their effect is to stretch, rotate, and translate the Gaussian ball into the data space resulting in a sort of l-dimensional pancake in d-dimensional space (right-hand part of figure 1) . This pancake is then convolved in data space with a Gaussian distribution $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ with a restricted covariance matrix $\mathbf{R}$. Depending on the specific choice for $\mathbf{R}$ we have:

- $\mathbf{R} = \sigma^2\mathbf{I}$: the latent variable model is called probabilistic principal component analysis [11] or sensible principal component analysis [9]. This terminology has been chosen while with $\sigma^2 \rightarrow 0$ conventional PCA is recovered.

- $\mathbf{R} \sim$ diagonal matrix: the latent variable model is standard factor analysis [10].

The advantage of such linear latent variable models is that the distribution of the observed data vectors is also Gaussian (and all marginal and conditional distributions for that matter):

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{R} + \mathbf{W}\mathbf{W}^T).$$

This means in specific, that the model can be viewed as a way of capturing the covariance structure $\mathbf{R} + \mathbf{W}\mathbf{W}^T$ of the d-dimensional observed data using only $(l + dl)$ parameters. Modelling the full

Data set	# attr.	# classes	# examples	# attr. (after pre-processing)	missing data
Dermatology	34	6	366	34	•
Glass	9	6	214	9	
Letter	16	26	20,000	16	
Optical	64	10	3,823	64	
Pen	16	10	7,494	16	
Soybean	35	19	683	134	•
Twos	256	10	1,948	256	
Vowel	10	11	990	10	
Waveform	21	3	600	21	
Waveform-noise	40	3	600	40	

Table 1: Properties of the data sets used in the experiments.

covariance matrix in the observed data space requires $(d(d+1)/2)$ parameters. The parameters $\mathbf{W}$, $\mathbf{R}$, and $\boldsymbol{\mu}$, of these linear latent variable models can be estimated by the EM algorithm [9, 10, 11] in which each step requires $O(ldn)$ operations.

Since these linear latent variable models define a proper probability model, they can be extended to mixture models which can also be trained efficiently with the EM algorithm [5, 11]. The mixture model (1) is then a linear combination of linear latent variable component distributions:

$$p_j(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_j, \mathbf{R}_j + \mathbf{W}_j\mathbf{W}_j^T).$$

With $\mathbf{R}_j$ isotropic, the model is called a mixture of principal component analysers (MPCA) [11] and with $\mathbf{R}_j$ diagonal, it is called a mixture of factor analysers (MFA) [5]. Having less constraints on its noise model, a MFA is therefore a more expressive model than a MPCA. These mixtures can be interpreted as a mixture of constrained Gaussians in which the number of parameters can be controlled through the dimension of the latent space (and hence, the size of $\mathbf{W}$) without putting too strong constraints on the flexibility of the model, that is, the form of the covariance matrix. Each step of the EM algorithm for a mixture of latent variable models requires $O(mldn)$ operations. This means that both in terms of the number of parameters and of computational complexity, MPCAs/MFAs smoothly cover the range between diagonal $(l=1)$ and full $(l=d-1)$ covariance matrices in a GMM.

In the recent literature, mixtures of latent variable models have been used successfully on some isolated density estimation problems. Tipping and Bishop compared MPCAs and GMMs on a 3-dimensional synthetic data set [11]. Both MPCAs and MFAs have also been applied to handwritten digit recognition, by fitting a mixture to each class and classifying digits according to the most likely model [6, 11]. The next section provides a more extensive comparison of GMMs and mixtures of latent variable models on a large and varied set of benchmarks problems.

3 Experiments

3.1 Experimental Set-Up

The experiments were done with various data sets out of the Irvine repository [3] and a subset of twos out of the NIST special database 3 of handwritten digits. We limited ourselves to classification problems since that was our focus in the experiments in [7]. Of course, for the current experiments only the input space of the data sets plays a role. An overview of the main characteristics of the different data sets is given in table 1. As can be seen from this table, the benchmarks largely differ in input dimension and number of patterns.

The raw data has been pre-processed in various ways. First of all, the ordinal inputs have been normalized to have zero mean and unit standard deviation on the training data. For the "soybean" data set part of the inputs are categorical and these are mapped to a 1-of-c coding, thus increasing the number of attributes (see the fifth column of Table 1). Finally, for the data sets indicated with a •, some of the inputs are missing for some patterns. For ordinal inputs, the missing value has been replaced by zero (the mean value after normalization) and for categorical inputs, an extra bit was added to the 1-of-c coding to encode the presence of a missing value.

The training of the models consisted of an initialization phase followed by 10 itera-

Mixture	train	test	5×2cv
Full	22.4(0.18)	26.1(0.34)	
Spherical	25.4(0.08)	25.7(0.28)	
Diagonal	24.9(0.10)	25.3(0.26)	$<$
MPCA-1	24.6(0.14)	25.2(0.31)	
MPCA-3	24.0(0.15)	25.1(0.27)	
MFA-3	23.5(0.17)	24.4(0.24)	$<$
MFA-1	23.8(0.19)	24.3(0.25)	$<$

Table 2: Input density modelling on the waveform data with 3 mixture components. Scores are in average negative log likelihood.

tions of the EM algorithm. The initialization of all mixture models used k-means clustering to determine the means. The mixing coefficients α_j were computed from the proportion of examples belonging to each cluster. Covariance matrices of the GMMs were calculated as the sample covariance of the points associated with (that is, closest to) the corresponding centres. Generative matrices $\mathbf{W}_j$ of the mixtures of latent variable models were initialized using a PCA on the points associated with the corresponding centres. The noise models $\mathbf{R}_j$ were initialized using the variance lost in the PCA projections found for each cluster. The number of iterations of the EM algorithm was chosen to be 10 because this turned out to be sufficient for maximizing the likelihood on the training set without additional over-fitting.

The 5×2cv test (a paired t-test) [4] was used on all data sets for testing the statistically significant difference. In this test, five replications of two-fold cross-validation are performed. The entries in the tables are the averages of the negative log-likelihood per data point over 10 simulations; the standard deviation is given between parentheses. A $<$-sign in the tables with results, indicates whether the score on the test set is significantly better (95%) than the one on the previous row. MFA-l and MPCA-l denote a mixture of latent variable models with l factors (that is, dimension of latent space l). The number of mixture components was varied for each benchmark, but only one representative choice is shown in this paper. Full details can be found in [7].

3.2 Artificial Data

As a first test, experiments were performed on two often used artificial classification problems with code for generating the data at the Irvine repository [3]: the waveform

Mixture	train	test	5×2cv
Full	45.4(0.27)	60.1(0.69)	
MPCA-3	51.5(0.14)	53.7(0.50)	$<$
Spherical	52.8(0.10)	53.4(0.48)	$<$
MPCA-1	52.3(0.12)	53.4(0.48)	
Diagonal	51.5(0.13)	52.4(0.48)	$<$
MFA-3	50.0(0.19)	51.9(0.50)	$<$
MFA-1	50.7(0.19)	51.7(0.51)	$<$

Table 3: Input density modelling on the waveform-noise data with three mixture components. Scores are in average negative log likelihood.

and the waveform-noise data (the last two rows of Table 1). The waveform data is generated according to:

$$\begin{aligned} x_i &= uh_1(i) + (1-u)h_2 + \varepsilon_i && \text{Class 1} \\ x_i &= uh_1(i) + (1-u)h_3 + \varepsilon_i && \text{Class 2} \\ x_i &= uh_2(i) + (1-u)h_3 + \varepsilon_i && \text{Class 3,} \end{aligned}$$

where $i = 1, 2, \dots 21$, u is uniform on $(0, 1)$, $\varepsilon_i \sim \mathcal{N}(0, \mathbf{I})$, and the h_i are shifted triangular waveforms: $h_1(i) = \max(6 - |i - 11|, 0)$, $h_2(i) = h_1(i-4)$, and $h_3(i) = h_1(i+4)$. For the waveform-noise data, the 19 additional attributes are all noise attributes with mean 0 and variance 1.

The results on the waveform and waveform-noise data are in tables 2 and 3. The GMM with a full covariance matrix obtains the best likelihood on the training set but the worst score on the test set: the model is over-fitting the data due to its many parameters. For all the other models the score on the test is only slightly worse than the score on the training set which suggests the absence of over-fitting. On the waveform data (table 2), the best results are obtained with the mixtures of latent variable models. It is especially worth noting that the MFA model with only one factor, performs much better than the GMM with a diagonal covariance matrix which has about the same number of free parameters. This shows that the axial alignment constraint of the diagonal model is not appropriate in this case. On the waveform-noise data (table 3), the best results are again obtained with MFAs but MPCAs are not performing that good. This is most likely due to the fact that the 19 additional inputs for this data set are just white noise and MFAs can separately model correlations and variance ($\mathbf{R}_j$ is diagonal) whereas MPCAs cannot ($\mathbf{R}_j$ is isotropic).

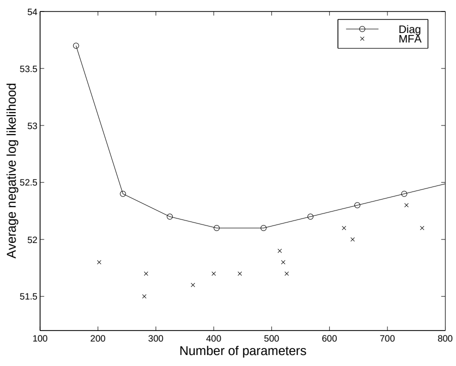

Figure 2: Comparison of a GMM with diagonal covariance matrices and a MFA on the waveform-noise data: the number of parameters versus the average negative log-likelihood on the test set.

We have also investigated the influence of the number of parameters of the mixture model on the results. Figure 2 illustrates that for a fixed number of parameters, GMMs with diagonal covariance matrices (varying the number of mixture components from 2 to 10) are always outperformed by MFAs (varying the number of factors and mixture components) on the waveform-noise data.

3.3 Real-World Data

Do the good results for mixtures of latent variable models on both artificial data sets, carry over to real-world data? To answer this question, experiments have been performed on the other databases listed in Table 1. The results are shown in Table 4, where the best method and the ones that are not significantly worse (95% with the 5×2cv test) are set in bold face. This has not been done for the glass data, where only the GMM with spherical covariance matrices performs significantly worse than the one with the lowest average (GMM with diagonal covariance matrices). The training set for the glass data consists of only 107 examples and this leads to highly variable results.

A quick inspection of the bold face results in Table 4 shows that the results are not uniform. However, a few general conclusions can still be drawn. Firstly, a GMM with spherical covariance matrices is, in general, too constrained to model the data. Only if the number of examples is small and the dimension of the data is high (soybean and optical data sets), it can outperform the other

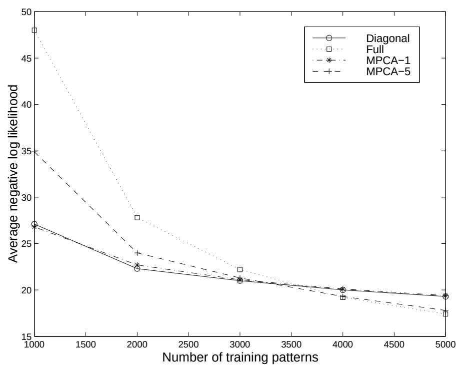

Figure 3: Comparison of GMMs and MPCAs on the letter data: the number of examples in the training set versus the average negative log-likelihood on the test set.

GMMs. Secondly, as expected, a GMM with full covariance matrices is highly sensitive to over-fitting if the number of parameters to be estimated is big compared to the number of examples. This is the case for the first five data sets in Table 4. However, for the other three benchmarks (letter, pen, and vowel), the number of examples is sufficient and a full GMM is amongst the best models. In this case, the model is not over-fitting the training data as can be seen from the fact that the errors on the test set (as given in Table 4) are close to the training errors for a GMM with full covariance matrices on letter: 13.1, on pen: 4.6, and on vowel: 8.4. Thirdly, on two data sets (soybean and optical) MFAs clearly outperform MPCAs but, in general, the results are quite similar.

Most importantly, for almost all data sets, an appropriate number of factors can be found such that the results are as good as or better than the best results obtained with a GMM. Quite surprisingly, diagonal GMMs sometimes outperform MFA-1 and MPCA-1 (which have a comparable number of parameters). This is most striking for dermatology, but this might be due to the representation of the data that seems to match very well the axial alignment constraint of the diagonal GMM. When rotating the coordinate system, the performance for a diagonal GMM is similar to the performance with MFA-1. Of course, if a full GMM performs well, choosing a high number of factors improves the results for a mixture of latent variable models.

Finally, we have also investigated in some more detail the dependence on the number

Data	GMM: covariance matrix			MFA: # factors		MPCA: # factors	
	Spher.	Diag.	Full	$l = 1$	$l = 5$	$l = 1$	$l = 5$
Dermatology (4)	39.4(0.7)	**13.0(4.8)**	33.6(5.4)	30.8(1.3)	31.7(7.9)	38.1(0.6)	37.7(0.5)
Glass (4)	9.7(1.3)	4.9(1.4)	15.8(8.6)	12.8(5.3)	16.0(8.2)	10.1(2.7)	14.9(7.2)
Optical (20)	67.6(1.1)	80.8(0.7)	82.4(3.9)	39.4(2.7)	**6.1(6.3)**	61.4(1.1)	50.3(1.7)
Soybean (4)	22.3(0.9)	25.9(3.3)	148.8(9.5)	-61.4(5.6)	**-102.8(27.5)**	9.4(1.0)	-15.5(1.5)
					$l = 10$		$l = 10$
Twos (5)	324.9(5.3)	251.6(5.9)	348.5(6.1)	**168.6(4.4)**	**160.1(6.5)**	300.5(3.4)	**163.2(4.2)**
					$l = 15$		$l = 15$
Letter (10)	19.6(0.1)	17.7(0.3)	**13.3(0.2)**	18.5(0.1)	14.7(0.1)	18.3(0.1)	13.8(0.2)
Pen (10)	16.1(0.3)	10.0(1.3)	**5.2(0.5)**	13.3(0.2)	7.3(0.2)	13.5(0.3)	6.6(0.2)
					$l = 8$		$l = 8$
Vowel (11)	12.3(0.1)	12.1(0.2)	**10.7(0.4)**	**11.0(0.3)**	**10.8(0.3)**	11.7(0.2)	**10.8(0.4)**

Table 4: Input density modelling with GMMs, MFAs, and MPCAs. The number of mixture components is indicated between parentheses after the name of each data set. Scores are in average negative log likelihood on the test set. The best scores are set in bold.

of training patterns. Figure 3, illustrates on the letter data that simple models (diagonal GMM and MPCA-1) perform best when the number of training patterns is small. When adding more patterns, these are gradually outperformed by more complex models.

4 Conclusions

Mixtures of latent variable models are a flexible alternative for standard Gaussian mixture models, the complexity of which can be tuned by varying the dimension of the latent space. They are expected to be especially useful when modelling high-dimensional data while having only a small number of examples. The choice of the number of mixture components and factors can be dealt with by standard techniques for model selection, such as cross-validation, an issue not dealt with in this paper. A recent Bayesian treatment of PCA [2] in which the appropriate number of factors can be determined automatically, looks especially promising.

Acknowledgements

The author gratefully acknowledges the Swiss National Science Foundation (FN:21-45621.95) for their support of this research. Matlab code for MPCAs was based on Michael Tipping's PhiVis code (`www.ncrg.aston.ac.uk/PhiVis`).

References

[1] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, 1995.

[2] C.M. Bishop. Bayesian PCA. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11. MIT Press, 1999. *To appear.*

[3] C. Blake, E. Keogh, and C. J. Merz. UCI repository of machine learning databases. Irvine: University of California, Department of Information and Computer Sciences. `www.ics.uci.edu/~mlearn/`, 1998.

[4] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[5] Z. Ghahramani and G. E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, University of Toronto, 1996.

[6] G. E. Hinton, P. Dayan, and M. Revow. Modelling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks*, 8(1):65–74, 1997.

[7] P. Moerland. Localized mixtures of experts. IDIAP-RR 98-14, IDIAP, `http://www.idiap.ch/`, 1998.

[8] D. Ormoneit and V. Tresp. Improved Gaussian mixture density estimates using Bayesian penalty terms and network averaging. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in NIPS*, volume 8, pages 542–548, Cambridge MA, 1996. MIT Press.

[9] S. Roweis. EM algorithms for PCA and SPCA. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in NIPS*, volume 10, pages 626–632, Cambridge MA, 1998. MIT Press.

[10] D. B. Rubin and D. T. Thayer. EM algorithms for ML factor analysis. *Psychometrika*, 47(1):69–76, 1982.

[11] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, February 1999.