SMEM Algorithm for Mixture Models

Naonori Ueda   Ryohei Nakano
NTT Communication Science Laboratories
Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237 Japan

Zoubin Ghahramani   Geoffrey E. Hinton
Gatsby Computational Neuroscience Unit, University College London
17 Queen Square, London WC1N 3AR, UK

Contact:

Naonori Ueda
NTT Communication Science Laboratories
Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237 Japan

Tel +81 774 93 5131
Fax +81 774 93 5155
E-mail ueda@cslab.kecl.ntt.co.jp

Running Title:

SMEM Algorithm for Mixture Models

SMEM Algorithm for Mixture Models

Abstract We present a split and merge EM (SMEM) algorithm to overcome the local maxima problem in parameter estimation of finite mixture models. In the case of mixture models, local maxima often involve having too many components of a mixture model in one part of the space and too few in another, widely separated part of the space. To escape from such configurations we repeatedly perform simultaneous split and merge operations using a new criterion for efficiently selecting the split and merge candidates. We apply the proposed algorithm to the training of Gaussian mixtures and mixtures of factor analyzers using synthetic and real data and show the effectiveness of using the split and merge operations to improve the likelihood of both the training data and of held-out test data. We also show the practical usefulness of the proposed algorithm by applying it to image compression and pattern recognition problems.

1 Introduction

Mixture density models, in particular normal mixtures, have been used extensively in the field of statistical pattern recognition (MacLachlan and Basford 1987). Recently, more sophisticated mixture density models such as mixtures of latent variable models (*e.g.,* probabilistic PCA and factor analysis) have been proposed to approximate the underlying data manifold (Hinton, Dayan, and Revow 1997; Tipping and Bishop 1997; Ghahramani and Hinton 1997). The parameters of these mixture models can be estimated using the EM algorithm (Dempster, Laird, and Rubin 1977) based on the maximum likelihood framework. A common and serious problem associated with these EM algorithms, however, is the local maxima problem. Although this problem has been pointed out by many researchers, the best way to solve it, in practice, is still an open question.

Two of the authors have proposed the deterministic annealing EM (DAEM) algorithm (Ueda and Nakano 1998), where a modified posterior probability parameterized by *temperature* is derived to avoid local maxima. However, when mixture density models are involved, local maxima arise when there are too many components of a mixture model in one part of the space and too few in another. The DAEM algorithm and other algorithms are not very effective at avoiding such local maxima because they are not able to move a component from an overpopulated region to an underpopulated region without passing through positions that give a lower likelihood. We therefore introduce a discrete move that simultaneously merges two components in an overpopulated region and splits a component in an underpopulated region.

The idea of performing split and merge operations has been successfully applied to clustering *e.g.*, (Ball and Hall 1967) or vector quantization, *e.g.*, (Ueda and Nakano 1994). Recently, split and merge operations have also been proposed for Bayesian normal mixture analysis (Richardson and Green 1997). Since they use split and merge operations with a Markov Chain Monte Carlo method, it is computationally much costly than our algorithm. In addition, we introduce new split and merge criteria to efficiently select the split and merge candidates.

Although the proposed method, unlike the DAEM algorithm, is limited to mixture models, we have experimentally comfirmed that our split and merge EM (SMEM) algorithm obtains better solutions than the DAEM algorithm. We have already given a basic idea of the SMEM algorithm (Ueda, Nakano, Ghahramani, and Hinton, 1999). This paper describes the algorithm in detail and shows real applications including image compression and pattern recognition.

2 EM Algorithm

Before describing our SMEM algorithm, we briefly review the EM algorithm. Suppose that a set $\mathcal{Z}$ consists of *observed data* $\mathcal{X}$ and *unobserved data* $\mathcal{Y}$. $\mathcal{Z} = (\mathcal{X}, \mathcal{Y})$ and $\mathcal{X}$ are called *complete data* and *incomplete data*, respectively. Assume that the joint probability density of $\mathcal{Z}$ is parametrically given as $p(\mathcal{X}, \mathcal{Y}; \Theta)$, where Θ denotes parameters of the density to be estimated. The maximum likelihood estimate of Θ is a value of Θ that maximizes the incomplete data log-likelihood function:

$$\begin{aligned}\mathcal{L}(\Theta; \mathcal{X}) &\stackrel{\text{def}}{=} \log p(\mathcal{X}; \Theta) \\ &= \log \int p(\mathcal{X}, \mathcal{Y}; \Theta) d\mathcal{Y}. \qquad (1)\end{aligned}$$

The characteristic of the EM algorithm is to maximize the incomplete data log-likelihood function by iteratively maximizing the expectation of the complete data log-likelihood function:

$$\mathcal{L}_c(\Theta; \mathcal{Z}) \stackrel{\text{def}}{=} \log p(\mathcal{X}, \mathcal{Y}; \Theta). \qquad (2)$$

Suppose that $\Theta^{(t)}$ denotes the estimate of Θ obtained after the tth iteration of the algorithm. Then, at the $t+1$th iteration, the E-step computes the expected complete data log-likelihood function denoted by $Q(\Theta|\Theta^{(t)})$ and defined by:

$$Q(\Theta|\Theta^{(t)}) \stackrel{\text{def}}{=} \mathrm{E}\{\mathcal{L}_c(\Theta; \mathcal{Z})|\mathcal{X}; \Theta^{(t)}\}, \qquad (3)$$

and the M-step finds the Θ maximizing $Q(\Theta|\Theta^{(t)})$. The convergence of the EM steps is theoretically guaranteed (Dempster, Laird and Rubin 1977).

3 Split and Merge EM Algorithm

3.1 Split and Merge Operations

In this paper, we restrict ourselves to mixture density models. The probability density function (pdf) of a mixture of M density models is given by

$$p(\boldsymbol{x}; \Theta) = \sum_{m=1}^{M} \alpha_m p_m(\boldsymbol{x}; \theta_m), \qquad (4)$$

where α_m is the mixing proportion of the mth model[1] and satisfies $\alpha_m \geq 0$ and $\sum_{m=1}^{M} \alpha_m = 1$. The $p_m(\boldsymbol{x}; \theta_m)$ is a d-dimensional density model corresponding to the mth model. Clearly, $\Theta = \{(\alpha_m, \theta_m),\ m = 1, \ldots, M\}$ is an unknown parameter set.

[1] Strictly speaking, we should call it "the mth model component" or "model component ω_m", but in order to make the notation and wording simpler we call it "model m" or "the mth model" hereafter.

In the case of mixture models, the model index $m \in \{1, \ldots, M\}$ is unknown for an observed data $\boldsymbol{x}_n$ and therefore m corresponds to the unobserved data mentioned in Sec. 2. Noting that the pdf of the complete data is $p(\boldsymbol{x}, m; \Theta) = \alpha_m p_m(\boldsymbol{x}; \theta_m)$ in this case, we have

$$Q(\Theta|\Theta^{(t)}) = \sum_{n=1}^{N} \sum_{m=1}^{M} \{\log \alpha_m p_m(\boldsymbol{x}_n; \theta_m)\} P(m|\boldsymbol{x}_n; \Theta^{(t)}). \tag{5}$$

Here, $P(m|\boldsymbol{x}_n; \Theta^{(t)})$ is a posterior probability and is computed as

$$P(m|\boldsymbol{x}_n; \Theta^{(t)}) = \frac{\alpha_m^{(t)} p_m(\boldsymbol{x}_n; \theta_m^{(t)})}{\sum_{l=1}^{M} \alpha_l^{(t)} p_l(\boldsymbol{x}_n; \theta_l^{(t)})}. \tag{6}$$

N is the number of observed data points.

Looking at equation 5 carefully, one can see that the Q function can be represented in the form of a direct sum:

$$Q(\Theta|\Theta^{(t)}) = \sum_{m=1}^{M} q_m(\theta_m|\Theta^{(t)}), \tag{7}$$

where

$$q_m(\theta_m|\Theta^{(t)}) = \sum_{n=1}^{N} P(m|\boldsymbol{x}_n; \Theta^{(t)}) \log \alpha_m p_m(\boldsymbol{x}_n; \theta_m) \tag{8}$$

and depends only on α_m and θ_m.

Let Θ^* denote the parameter values estimated by the usual EM algorithm. Then, after the EM algorithm has converged, the Q function can be rewritten as

$$Q^* = q_i^* + q_j^* + q_k^* + \sum_{m, m \neq i,j,k} q_m^*. \tag{9}$$

We then try to increase the first three terms of the right-hand side of equation 9 by merging models i and j to produce a model i', and splitting the model k into two models j' and k'.

3.1.1 Initialization

To reestimate the parameters of these new models, we have to initialize the parameters corresponding to the new models using Θ^*. Intuitively natural initializations are given below. The initial parameter values for the merged model i' are set as linear combinations of the original ones before the merge:

$$\alpha_{i'} = \alpha_i^* + \alpha_j^* \quad \text{and} \quad \theta_{i'} = \frac{\alpha_i^* \theta_i^* + \alpha_j^* \theta_j^*}{\alpha_i^* + \alpha_j^*}. \tag{10}$$

Noting that the mixing proportion is estimated as the average of *posterior* over data: $\alpha_l^* = 1/N \sum_{n=1}^{N} P(l|\boldsymbol{x}_n; \Theta^*)$, one can see that the initialization of $\theta_{i'}$ is the linear combination of θ_i^* and θ_j^*, weighted by the posteriors.

On the other hand, as for models j' and k', we set

$$\alpha_{j'} = \alpha_{k'} = \frac{\alpha_k^*}{2} \qquad \theta_{j'} = \theta_k^* + \epsilon \quad \text{and} \quad \theta_{k'} = \theta_k^* + \epsilon', \tag{11}$$

where ϵ or ϵ' is some small random perturbation vector or matrix (*i.e.*, $\|\epsilon\| \ll \|\theta_k^*\|$). In the case of mixture Gaussians, covariance matrices $\mathbf{\Sigma}_{j'}$ and $\mathbf{\Sigma}_{k'}$ should be positive definite. In this case, we can initialize them as

$$\mathbf{\Sigma}_{j'} = \mathbf{\Sigma}_{k'} = \det(\mathbf{\Sigma}_k^*)^{1/d} I_d \tag{12}$$

instead of equation 11. Here, $\det(\mathbf{\Sigma})$ denotes the determinant of matrix $\mathbf{\Sigma}$ and I_d is the d-dimensional identity matrix. Figure 1 shows a simple example of the initialization steps for a two-dimensional Gaussian case using equations 10, 11 and 12.

3.1.2 Partial EM steps

The parameter reestimation for $m' = i', j'$, and k' can be done by using EM steps, but note that instead of eqution 6 we use the following modified posterior probability:

$$P(m'|\boldsymbol{x}; \Theta^{(t)}) = \frac{\alpha_{m'}^{(t)} p_{m'}(\boldsymbol{x}; \theta_{m'}^{(t)})}{\sum_{l=i',j',k'} \alpha_l^{(t)} p_l(\boldsymbol{x}; \theta_l^{(t)})} \times \sum_{m=i,j,k} P(m|\boldsymbol{x}; \Theta^*), \quad \text{for } m' = i', j', k'. \tag{13}$$

Using equation 13, the sum of posterior probabilities for models i', j' and k' becomes equal to the sum of posterior probabilities for models i, j, and k just before split and merge. That is,

$$\sum_{m'=i',j',k'} P(m'|\boldsymbol{x}; \Theta^{(t)}) = \sum_{m=i,j,k} P(m|\boldsymbol{x}; \Theta^*) \tag{14}$$

always holds during the reestimation process. By this, we can reestimate the parameters for models i', j', and k' consistently without affecting the other models. We call these EM steps *partial EM steps*. These partial EM steps make the total algorithm efficient.

3.2 SMEM Algorithm

After the partial EM steps, the usual EM steps, called the *full EM steps*, are performed as a post processing operation. After these steps, if Q is improved, then we accept the new estimate and repeat the above after setting the new parameters to Θ^*. Otherwise, we reject it and go back to Θ^* and try another candidate. We summarize these as the following SMEM algorithm:

SMEM Algorithm

1. Perform the usual EM updates from some initial parameter value Θ until convergence. Let Θ^* and Q^* denote the estimated parameters and corresponding Q function value after the EM algorithm has converged, respectively.

2. Sort the split and merge candidates by computing split and merge criteria (described in the next section) based on Θ^*. Let $\{i, j, k\}_c$ denote the cth candidate.

3. For $c = 1, \ldots, C_{max}$, perform the following: After making the initial parameter settings based on Θ^*, perform the *partial EM steps* for $\{i, j, k\}_c$ and then perform the *full EM steps* until convergence. Let Θ^{**} be the obtained parameters and Q^{**} be the corresponding Q function value after the full EM has converged. If $Q^{**} > Q^*$, then set $Q^* \leftarrow Q^{**}$, $\Theta^* \leftarrow \Theta^{**}$ and go to Step 2.

4. Halt with Θ^* as the final parameters.

Note that when a certain split and merge candidate which improves the Q function value is found in Step 3, the other successive candidates are ignored. There is no guarantee therefore that the split and merge candidates that are chosen will give the largest possible improvement in Q. This is not a major problem, however, because the split and merge operations are performed repeatedly. If there were no heuristics for ordering potential split and merge operations, we would have to consider them all $C_{max} = M(M-1)(M-2)/2$, but experimentally we have confirmed that $C_{max} \simeq 5$ may be enough because the split and merge criteria do work well.

The SMEM algorithm monotonically increases the Q function value and if the Q function value does not increase for all $c = 1, \ldots, C_{max}$, then the algorithm stops. Since the full EM steps equivalent to the original EM steps are performed after the convergence of the partial EM steps, it is clear that the SMEM algorithm still maintains the global convergence properties of the EM algorithm.

Note that in the SMEM algorithm the split and merge operations are simultaneously performed so that the total number of mixture components is unchanged. In general, the Q function value increases as the number of parameters (or the number of mixture components) increases. Thus, in order to check whether Q is improved by the rearrangement of model components at Step 3, it is necessary to keep the number of model components unchanged.

Intuitively, a simultaneous split and merge can be viewed as a way of tunneling through low-likelihood barriers, thereby eliminating many poor local optima. In this respect, it has some similarities with simulated annealing, but the moves that are considered are long-range and are very specific to the particular problems that arise when fitting mixture models.

3.3 Split and Merge Criteria

Each of the split and merge candidates can be evaluated by its Q function value after Step 3 of the SMEM algorithm mentioned in Sec. 3.2. However, since there are so many candidates, some reasonable criteria for ordering the split and merge candidates should be utilized to accelerate the SMEM algorithm.

Merge criterion

In general, when there are many data points each of which has almost equal posterior probability given by equation 6 for any two components, it can be thought that these two components might be merged. To numerically evaluate this, we define the following merge criterion:[2]

$$J_{merge}(i,j;\Theta^*) = \mathbf{P}_i(\Theta^*)^T \mathbf{P}_j(\Theta^*), \tag{15}$$

where $\mathbf{P}_i(\Theta^*) = (P(i|\boldsymbol{x}_1;\Theta^*),\ldots,P(i|\boldsymbol{x}_N;\Theta^*))^T \in \mathcal{R}^N$ is an N-dimensional vector consisting of the posterior probabilities for the ith model. T denotes the transpose operation. $\|\cdot\|$ denotes the Euclidean vector norm. Clearly, two components ω_i and ω_j with large $J_{merge}(i,j;\Theta^*)$ are good candidates for a merge.

Split criterion

As a split criterion (J_{split}), we define the *local Kullback divergence* as:

$$J_{split}(k;\Theta^*) = \int f_k(\boldsymbol{x};\Theta^*) \log \frac{f_k(\boldsymbol{x};\Theta^*)}{p_k(\boldsymbol{x};\theta_k^*)} d\boldsymbol{x}, \tag{16}$$

which is the distance between two distributions: the local data density $f_k(\boldsymbol{x})$ around the kth model and the density of the kth model specified by the current parameter estimate Θ^*. The local data density is defined as:

$$f_k(\boldsymbol{x};\Theta^*) = \frac{\sum_{n=1}^{N} \delta(\boldsymbol{x}-\boldsymbol{x}_n) P(k|\boldsymbol{x}_n;\Theta^*)}{\sum_{n=1}^{N} P(k|\boldsymbol{x}_n;\Theta^*)}. \tag{17}$$

This is a modified *empirical distribution* weighted by the posterior probability so that the data around the kth model is focused on. Note that when the weights are equal, *i.e.,* $P(k|\boldsymbol{x};\Theta^*) = 1/M$, equation 17 is the usual empirical distribution:

$$p_k(\boldsymbol{x};\Theta^*) = \frac{1}{N}\sum_{n=1}^{N} \delta(\boldsymbol{x}-\boldsymbol{x}_n). \tag{18}$$

Since it can be thought that the model with the largest $J_{split}(k;\Theta^*)$ has the worst estimate of the local density, we should try to split it.

The split criterion defined by equation 16 can be viewed as a likelihood ratio test. That is, $f_k(\boldsymbol{x};\Theta^*)/p_k(\boldsymbol{x};\theta_k^*)$ can be interpreted as the likelihood ratio test statistic. In this sence, our split criterion is similar to a cluster validity test formula proposed by Wolfe (1970).

Sorting candidates

Using J_{merge} and J_{split}, we sort the split and merge candidates as follows. First, the merge candidates are sorted based on J_{merge}. Then, for each sorted merge candidate $\{i,j\}_c$, the split candidates excluding $\{i,j\}_c$ are sorted as $\{k\}_c$. By combining these results and renumbering them, we obtain $\{i,j,k\}_c,\ c = 1,\ldots,M(M-1)(M-2)/2$.

[2] This merging criterion favors merges with larger classes. To avoid this bias, a modified criterion:

$$J_{merge}(i,j;\Theta^*) = \frac{\mathbf{P}_i(\Theta^*)^T \mathbf{P}_j(\Theta^*)}{\|\mathbf{P}_i(\Theta^*)\|\|\mathbf{P}_j(\Theta^*)\|},$$

can be used. In our experiments, however, we used equation 15 for simplicity and the merge criterion did work well as shown later.

4 Application to Density Estimation by Mixture of Gaussians

4.1 Synthetic Data

We apply the proposed algorithm to a density estimation problem using a mixture of Gaussians. In this case, $p_m(\boldsymbol{x};\theta_m)$ on the right hand side of equation 4 becomes

$$p_m(\boldsymbol{x};\theta_m) = (2\pi)^{-d/2}\mathrm{det}(\Sigma_m)^{-1/2}\exp\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_m)^T\Sigma_m^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_m)\}. \tag{19}$$

Clearly, θ_m corresponds to mean vector $\boldsymbol{\mu}_m$ and covariance matrix $\boldsymbol{\Sigma}_m$.

In the case of density estimation by a mixture of Gaussians, as is well known, the global maxima of the likelihood correspond to singular solutions with zero covariances. Therefore, the SMEM algorithm may converge to the singular solutions. To prevent this, we used a Bayesian regularization method (Ormoneit and Tresp 1996). That is, the update equation for the covariance matrix is given by

$$\Sigma_m^{(t+1)} = \frac{\sum_{\boldsymbol{x}\in\mathcal{X}}(\boldsymbol{x}-\boldsymbol{\mu}_m^{(t+1)})(\boldsymbol{x}-\boldsymbol{\mu}_m^{(t+1)})^T P(m|\boldsymbol{x};\Theta^{(t)}) + \lambda I_d}{\sum_{\boldsymbol{x}\in\mathcal{X}} P(m|\boldsymbol{x};\Theta^{(t)}) + 1}, \quad m = 1,\ldots,M. \tag{20}$$

Here I_d is the d-dimensional unit matrix and λ is a regularization constant determined by some validation data. In the experiment we set $\lambda = 0.1$.

First, we utilized the two-dimensional synthetic data in Figure 2 to visually demonstrate the usefulness of the split and merge operations. The initial mean vectors and covariance matrices were, as shown in Figure 2(b), set to near the means of all of the data and unit matrices, respectively. The usual EM algorithm converged to the local maximum solution shown in Figure 2(c), whereas the SMEM algorithm converged to the superior solution shown in Figure 2(f), very close to the true one. The split of the 1st Gaussian shown in Figure 2(d) appeared to be redundant, but as shown in Figure 2(e) they are successfully merged and the original two Gaussians were improved. This indicates that the split and merge operations not only appropriately assign the number of Gaussians in a local data space, but can also improve the Gaussian parameters themselves.

4.2 Real Data

Next, we tested the proposed algorithm using 20-dimensional real data[3] (facial images processed into feature vectors) where the local maxima made the optimization difficult. The data size was 103 for training and 103 for test. We ran three algorithms (EM, DAEM, and SMEM) for ten different initializations using the k-means clustering algorithm. We set $M = 5$ and used a diagonal covariance for each Gaussian. Table 1 shows the summary statistics (mean, standard deviation (std), maximum, and minimum) of log-likelihood values per sample size obtained by each of the EM, DAEM, and SMEM algorithms for ten differernt initializations. As shown in Table 1, even the worst solution found by the SMEM algorithm was better than the best solutions found by the other algorithms on both the training and test data. Moreover, the log-likelihoods achieved by the SMEM algorithm have lower variance than those achieved by the EM algorithm.

[3] See (Ueda and Nakano, 1998) for the details.

Figure 3 shows log-likelihood value trajectories accepted in Step 3 of the SMEM algorithm during the estimation process. The dotted lines in Figure 3 denote the starting points of Step 2. Note that it is due to the initialization in Step 3 that the log-likelihood decreases just after the split and merge. Comparing the convergence points at Step 3 marked by the '$\circ$' symbol in Figure 3, one can see that the successive split and merge operations improved the log-likelihood for both the training and test data, as we expected. Table 2 compares the number of iterations executed by the three algorithms. Note that in the SMEM algorithm, the number includes EM-steps not only partial and full EM steps for accepted operations, but also EM-steps for rejected ones. From Table 2, the SMEM algorithm was about 8.7 times slower than the original EM algorithm. The average rank of the accepted split and merge candidates was 1.8 (std=0.9), which indicates that the proposed split and merge criteria worked very well.

5 Application to Dimensionality Reduction using Mixture of Factor Analyzers

5.1 Factor Analyzers

A single factor analyzer (FA) (Anderson 1984) assumes that an observed p-dimensional variable $\boldsymbol{x}$ is generated as a linear transformation of some lower q-dimensional *latent* variable $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ plus additive Gaussian noise $\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, \Psi)$. Ψ is a diagonal matrix. That is, the generative model can be written as

$$\boldsymbol{x} = \mathbf{W}\boldsymbol{z} + \boldsymbol{v} + \boldsymbol{\mu}. \tag{21}$$

Here, $\mathbf{W} \in \mathcal{R}^{p \times q}$ is a transformation matrix and is called a *factor loading matrix*. $\boldsymbol{\mu}$ is a mean vector. Then, from a simple calculation, the pdf of the observed data by an FA model can be obtained by

$$p(\boldsymbol{x}; \Theta) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^T + \Psi). \tag{22}$$

Noting that $\mathbf{W}$ can be represented by linearly independent column vectors as $\mathbf{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_q]$ where $\boldsymbol{w}_i \in \mathcal{R}^{p \times 1}$, we can rewrite equation 21 as

$$\boldsymbol{x} = \sum_{i=1}^{q} z_i \boldsymbol{w}_i + \boldsymbol{v} + \boldsymbol{\mu}, \tag{23}$$

where z_i is the ith component of $\boldsymbol{z}$. Clearly, equation 23 shows that $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_q$ form the basis in a latent space. Hence, FA can be interpreted as a dimensionality reduction model extracting a linear manifold (affine subspace) $\mathcal{M} = L^{(q)} + \boldsymbol{\mu}$ underlying the given observed data space, where $L^{(q)}$ denotes the linear subspace of $\mathcal{R}^p$ spanned by the basis $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_q$.

5.2 Mixture of Factor Analyzers

A mixture of factor analyzers (MFA) proposed by (Ghahramani and Hinton 1997) is an extension of single FA. That is, MFA is defined as the combination of M mixture of FAs and can be thought of as a reduced dimensional mixture of Gaussians. The MFA model extracts q-dimensional *locally* linear manifolds $\mathcal{M}_m = L_m^{(q)} + \boldsymbol{\mu}_m$ for $m = 1, \ldots, M$ underlying the given high dimensional data. More intuitively, the MFA model can perform clustering and dimensionality reduction simultaneously. Since the MFA model extracts a *globally* nonlinear manifold, it is more flexible than the single FA model.

The pdf of the observed data by M mixtures of FAs is given by

$$p(\boldsymbol{x};\Theta) = \sum_{m=1}^{M} \alpha_m \mathcal{N}(\boldsymbol{\mu}_m, \mathbf{W}_m\mathbf{W}_m^T + \Psi_m). \tag{24}$$

See (Ghahramani and Hinton 1997) for the details. Note that equation 24 is a natural extension of equation 22. The unknown parameters $\Theta = \{\alpha_m, \boldsymbol{\mu}_m, \mathbf{W}_m | m = 1, \ldots, M\}$ of the MFA model can be estimated by the EM algorithm. In this case, the complete data log-likelihood becomes

$$\begin{aligned} \mathcal{L}_c &= \log \prod_{n=1}^{N}\prod_{m=1}^{M} p(\boldsymbol{x}_n, \boldsymbol{z}_n, m; \Theta_m)^{u_m} \\ &= \sum_{n=1}^{N}\sum_{m=1}^{M} u_m \log p(\boldsymbol{x}_n, \boldsymbol{z}_n, m; \Theta_m) \end{aligned} \tag{25}$$

Here, u_m is a mixture indicator variable, where if $\boldsymbol{x}_n$ is generated by the mth model, $u_m = 1$; otherwise, $u_m = 0$. Since $\mathcal{L}_c$ can be represented in the form of a direct sum, the Q function is also decomposable and therefore the SMEM algorithm is straightforwardly applicable to the parameter estimation of the MFA model.

5.3 Demonstration

Figure 4 shows results of extracting a one-dimensional manifold from three-dimensional data (noisy shrinking spiral) using the EM and SMEM algorithms. The data points in Figure 4 were generated by

$$(X_1, X_2, X_3) = ((13 - 0.5t)\cos t, -(13 - 0.5t)\sin t, t) + \textit{additive noise}, \tag{26}$$

where $t \in [0, 4\pi]$. In this case, each factor loading matrix $\mathbf{W}_m$ becomes a three-dimensional column vector corresponding to each thick line in Figure 4. The center position and the direction of each thick line are μ_m and $\mathbf{W}_m$, respectively. In addition, the length of each thick line is $2\|\mathbf{W}_m\|$.

Although the EM algorithm converged to poor local maxima as shown in Figure 4(c), the SMEM algorithm successfully extracted the data manifold shown in Figure 4(f).

Table 3 compares average log-likelihoods per data point over ten different initializations. The log-likelihood values were drastically improved on both the training and test data by the SMEM algorithm.

5.4 Practical Applications

5.4.1 Image compression

An MFA model is available for block transform image coding. In this method, as in usual block transform coding approaches such as Karhunen-Lòeve transformation or principal component analysis (PCA), an image is subdivided into nonoverlapping blocks of $b \times b$ pixels.

Typically, $b = 8$ or $b = 16$ is employed. Each block is regarded as a $d(= b \times b)$ dimensional vector $\boldsymbol{x}$. Let $\mathcal{X}$ be a set of obtained $\boldsymbol{x}$ values. Then, using $\mathcal{X}$, an image is transformed by the following steps:

Compression Algorithm

1. Set the desired dimensionality q and the number of mixture components M.
2. Estimate $\boldsymbol{\mu}_m$ and $\mathbf{W}_m$, for $m = 1, \ldots, M$ by fitting an MFA model to $\mathcal{X}$.
3. For each $\boldsymbol{x} \in \mathcal{X}$,
   (a) compute
   $$\hat{\boldsymbol{x}}^{(m)} = \mathbf{W}_m(\mathbf{W}_m^T\mathbf{W}_m)^{-1}\mathbf{W}_m^T(\boldsymbol{x} - \boldsymbol{\mu}_m) + \boldsymbol{\mu}_m, \quad \text{for} \quad m = 1, \ldots, M. \tag{27}$$
   (b) assign $\hat{\boldsymbol{x}}^{(m^*)}$ as a reconstructed vector that minimizes the squared error $\|\hat{\boldsymbol{x}}^{(m)} - \boldsymbol{x}\|^2$ by its reconstruction.
4. Transform each $\hat{\boldsymbol{x}}^{(m^*)}$ into a block image (reconstructed block image).

The derivation of equation 27 is as follows. The least-squares reconstructed vector $\hat{\boldsymbol{x}}$ is, as shown in Figure 5, obtained by orthogonally projecting $\boldsymbol{x} - \boldsymbol{\mu}$ onto $L_m^{(q)}$ and adding $\boldsymbol{\mu}_m$. That is,

$$\hat{\boldsymbol{x}} = P_m(\boldsymbol{x} - \boldsymbol{\mu}_m) + \boldsymbol{\mu}_m, \tag{28}$$

where P_m is the projection matrix of $L_m^{(q)}$ and is computed from

$$P_m = \mathbf{W}_m(\mathbf{W}_m^T\mathbf{W}_m)^{-1}\mathbf{W}_m^T. \tag{29}$$

Substituing equation 29 into equation 28, we obtain equation 27.

Figure 6 shows an example of compressed image by MFA with $q = 4$ and $M = 10$. Figure 6(a) shows a sample image used for the training data $\mathcal{X}$. We should use test images independent of the training image, but for simplicity we used the same image for the training and test. For comparison, we also tried usual PCA-based image compression method (Figure 6(b)). In the case of image compression by PCA, $\mathbf{W}$ is an orthogonal matrix composed of q column vectors (eigenvectors) corresponding to the q largest eigenvalues of the sample autocorrelation matrix:

$$S = \frac{1}{|\mathcal{X}|}\sum_{\boldsymbol{x} \in \mathcal{X}} \boldsymbol{x}\boldsymbol{x}^T. \tag{30}$$

Here, $|\mathcal{X}|$ denotes the number of components of $\mathcal{X}$. Since $\mathbf{W}\mathbf{W}^T = I_q$ (*i.e.,* a set of column vectors $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_q\}$ is orthogonal), the reconstructed vector by PCA is given by

$$\hat{\boldsymbol{x}} = \mathbf{W}\mathbf{W}^T\boldsymbol{x} \tag{31}$$

The major difference between the PCA and MFA approaches is that the former finds a global subspace from $\mathcal{X}$, while the latter simultaneously clusters the data and finds an optimum subspace for each cluster. Therefore, it is natural that the results by the MFA approaches (Figures 6(c),(d)) were much richer than that by the PCA approach (Figure 6(b)).

Note that a mixture of PCA (MPCA) (*e.g.,* Tipping and Bishop 1997) is a much better model than PCA but we have not compared it here because our purpose was to compare the SMEM algorithm with the EM algorithm for the MFA-based image compression.

Comparing Figure 6(c) against Figure 6(d), one can see that the quality of the reconstructed image using the SMEM algorithm is better than that using the EM algorithm. The mean squared errors per block of these constructed images were 15.8×10^3, 10.1×10^3, and 7.3×10^3 for Figures 6(b), (c), and (d), respectively.

5.4.2 Application to Pattern Recognition

The MFA model is also applicable to pattern recognition tasks (Hinton, Dayan, and Revow 1997) since once an MFA model is fitted to each class, we can compute the posterior probability for each data point. More specifically, the pdf of class ω_i by the MFA model is given by

$$p_i(\boldsymbol{x}; \Theta_i) = \sum_{m=1}^{M} P_{im} \mathcal{N}(\boldsymbol{\mu}_{im}, \mathbf{W}_{im}\mathbf{W}_{im}^T + \boldsymbol{\Psi}_{im}), \tag{32}$$

where Θ_i is a set of MFA model parameters for class ω_i. That is, $\Theta_i = \{P_{im}, \boldsymbol{\mu}_{im}, \mathbf{W}_{im}, \boldsymbol{\Psi}_{im} | \ m = 1, \ldots, M\}$. P_{im} is the mixing proportion of the mth model for class ω_i.

Using equation 32, the posterior probability of class ω_i given $\boldsymbol{x}$ is computed from

$$P(\omega_i|\boldsymbol{x}) = \frac{P_i \ p_i(\boldsymbol{x}; \Theta_i)}{\displaystyle\sum_{j=1}^{C} P_j \ p_j(\boldsymbol{x}; \Theta_j)}. \tag{33}$$

Here, C is the number of classes and P_i is the prior of class ω_i and is estimated from

$$P_i = \frac{N_i}{N}, \tag{34}$$

where N_i is the number of training samples of class ω_i and N is the total sample size of all classes. Then, the optimum class i^* for $\boldsymbol{x}$ based on the *Bayes decision rule* is written as follows:

$$\begin{aligned} i^* &= \arg\max_i P(\omega_i|\boldsymbol{x}) \\ &= \arg\max_i N_i \sum_{m=1}^{M} P_{im} \mathcal{N}(\boldsymbol{\mu}_{im}, \mathbf{W}_{im}\mathbf{W}_{im}^T + \boldsymbol{\Psi}_{im}) \end{aligned} \tag{35}$$

We compared the MFA model with another classification method based on clustering and dimensionality reduction, called the Multiple Subspace Method (MSS) proposed by (Sugiyama and Ariki 1998). In order to define the MSS method we will first describe the simpler Subspace (SS) method for classification known as CLAFIC (Oja 1983). In the CLAFIC method, a linear subspace is extracted from the training data for each class and then the distance between an input vector $\boldsymbol{x}$ to be classified and its projected vector onto the linear subspace is computed for each class. Next, the input vector is classified as class ω_i^* with the minimum distance. See (Oja 1983) for the details.

In the SS method, a single subspace is extracted for each class. On the other hand, in the MSS method, first, the data is divided into several clusters by using the k-means algorithm. Then, for each cluster the CLAFIC method is performed. This MSS method can be expected to improve the classification performance of the CLAFIC method, but due to the absence of a probability density model it does not perform Bayes classification.

We tried a digit recognition task (10 digits (classes)) using three methods (MSS method, MFA-based method with the EM algorithm, and MFA-based method with the SMEM algorithm). The data was created using (degenerate) Glucksman's features (16-dimensional data) by NTT labs (Ishii 1989). The data size was 200/class for training and 200/class for test.

The recognition accuracy values for the training and test data obtained by these methods are given in Figure 7. In Figure 7, q denotes the dimensionality of the latent space and M is the number of components in mixture. Note that the SS (CLAFIC) method corresponds to $M = 1$ in the MSS method shown in Figure 7(a). Clearly, the MFA-based method with the SMEM algorithm consistently outperformed both the MSS method and the MFA-based method with the EM algorithm. The recognition accuracy by the 3-nearest neighbor (3NN) classifier was 88.3%. It is interesting that the MFA approach by the SMEM algorithm could outperform the nearest neighbor approach when $q = 3$ and $M = 5$ (91.9%). This suggests that the intrinsic dimensionality of the data might be three or so.

6 Conclusion

We have shown how simultaneous split and merge operations can be used to move components of a mixture model from regions in a space in which there are too many components to regions in which there are too few. Such moves cannot be accomplished by methods that continuously move components through intermediate locations because the likelihoods are lower at these locations. A simultaneous split and merge can be viewed as a way of tunneling through low-likelihood barriers, thereby eliminating many non-global optima.

Note that the SMEM algorithm is applicable to a wide variety of mixture models, as long as decomposition 7 holds. To make the split and merge method more efficient, we have introduced criteria for deciding which splits and merges to consider and have shown that these criteria work well for low-dimensional synthetic datasets and for higher-dimensional real datasets. Our SMEM algorithm consistently outperforms the standard EM algorithm and therefore it can be very useful in practice.

In the SMEM algorithm, as mentioned before, the split and merge operations are utilized to improve the parameter estimates within the maximum likelihood framework. However, by introducing probability measures over model we could also utilize the split and merge operations to determine the appropriate number of components within the Bayesian framework. This extension is now in progress.

References

[1] Anderson T. W. 1984. *An Introduction to Multivariate Statistical Analysis, 2nd Edition*, chap. 14, John Wiley & Sons.

[2] Ball G. H. and Hall D. J. 1967. A clustering technique for summarizing multivariate data. *Behavioral Science*, 12, 153–155.

[3] Dempster A. P., Laird N. M., and Rubin D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, 39, 1–38.

[4] Hinton G. E., Dayan P., and Revow M. 1997. Modeling the minifolds of images of handwritten digits. *IEEE Trans. PAMI*, 8(1), 65–74.

[5] Ishii K. 1989. Design of a recognition dictionary using artificially distorted characters. *Systems and computers in Japan*, 21(9), 669–677.

[6] MacLachlan, G. and Basford K. 1988. Mixture models: Inference and applications to clustering, *Marcel Dekker*.

[7] Erkki Oja. 1983. Subspace methods of pattern recognition. *Research Studies Press Ltd.*

[8] Ormoneit D. and Tresp V. 1996. Improved Gaussian mixture density estimates using Bayesian penalty terms and network averaging. to appear *Neural Information Processing Systems 8(NIPS96)*, MIT Press, Cambridge MA, 542–548.

[9] Richardson S. and Green P. J. 1997. On Bayesian analysis of mixtures with an unknown number of components. *J. R. Statist. Soc.*, B, 59(4),731–792.

[10] Sugiyama Y. and Ariki Y. 1998. Automatic classification of TV sports news videos by multiple subspace method. *Trans. of The Institute of Electronics, Information and Communication Engineers* (in Japanese), DII, 9, 2112–2119.

[11] Tipping M. E. and Bishop C. M. 1997. Mixtures of probabilistic principal component analysers. *Tech. Rep. NCRG-97-3*, Aston Univ. Birmingham, UK.

[12] Ueda N. and Nakano R. 1998. Deterministic annealing EM algorithm. *Neural Networks*, 11(2), 271–282.

[13] Ueda N. and Nakano R. 1994. A new competitive learning approach based on an equidistortion principle for designing optimal vector quantizers. *Neural Networks*, 7(8), 1211–1227.

[14] Ueda N., Nakano R., Ghahramani Z., and Hinton G. E. 1999. SMEM algorithm for mixture models, to appear *Neural Information Processing Systems 11(NIPS98)*, MIT Press, Cambridge MA.

[15] Wolfe J. H. 1970. Pattern clustering by multivariate mixture analysis. *Multivariate Behavioral Research*, 5, 329–350.

[16] Ghahramani Z. and Hinton G. E. 1997. The EM algorithm for mixtures of factor analyzers. *Tech. Report CRG-TR-96-1, Univ. of Toronto.*
`http://www.gatsby.ucl.ac.uk/~zoubin/papers/tr-96-1.ps.gz`

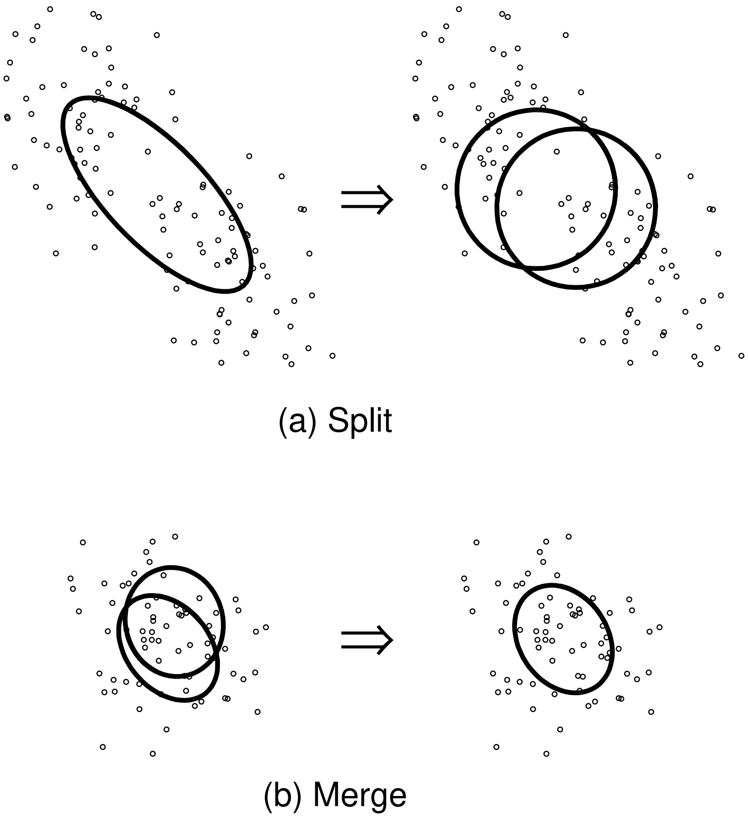


Figure 1: An example of initializtion in a two-dimensional Gaussian case. (a) A Gaussian just before split (left) and initialized Gaussians just after split (right). (b) Two Gaussians just before merge (left) and an initialized Gaussian just after merge (right).

Figure 1, Ueda

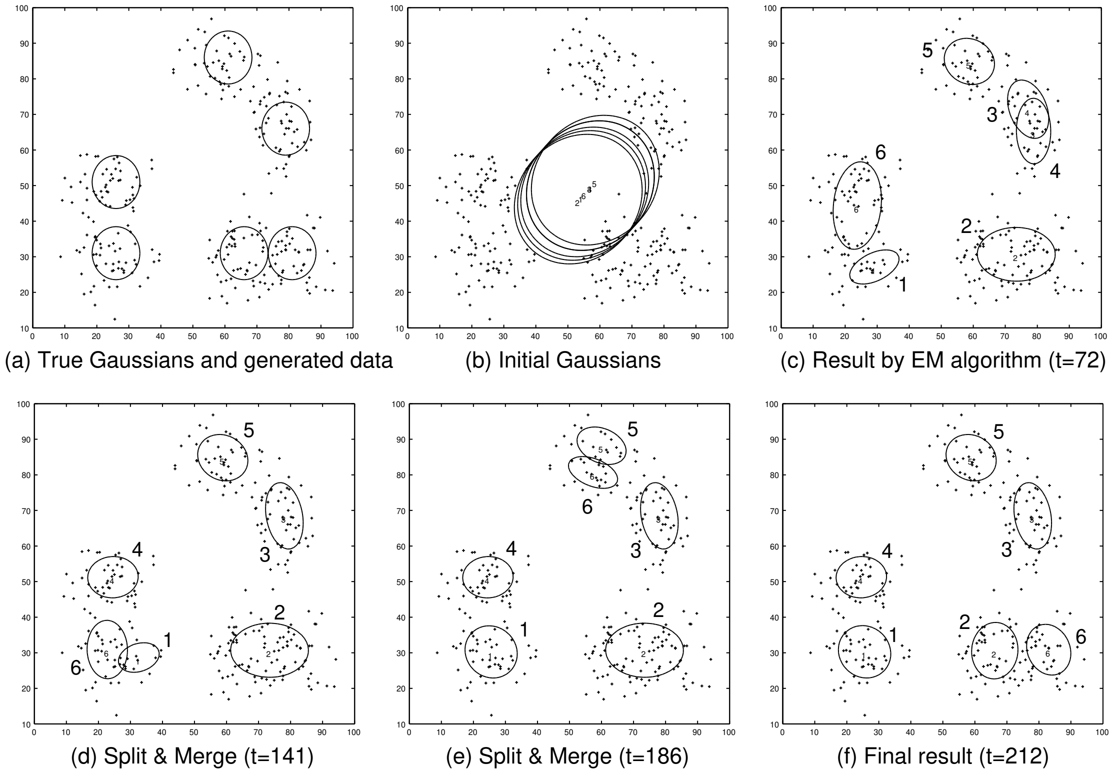


Figure 2: Results by the EM and SMEM algorithms for a two-dimensional Gaussian mixture density estimation problem. (a) Contours of true Gaussians, (b) initial density, (c) result by the EM algorithm, (d)-(e) examples of split and merge operations by the SMEM algorithm, and (f) the final result.

Figure 2, Ueda

Table 1: Log-likelihood/sample size.

		Initial value	EM	DAEM	SMEM
Training	mean	-159.1	-148.2	-147.9	-145.1
	std	1.77	0.24	0.04	0.08
	max	-157.3	-147.7	-147.8	-145.0
	min	-163.2	-148.6	-147.9	-145.2
Test	mean	-168.2	-159.8	-159.7	-155.9
	std	2.80	1.00	0.37	0.09
	max	-165.5	-158.0	-159.6	-155.9
	min	-174.2	-160.8	-159.8	-156.0

Table 1, Ueda

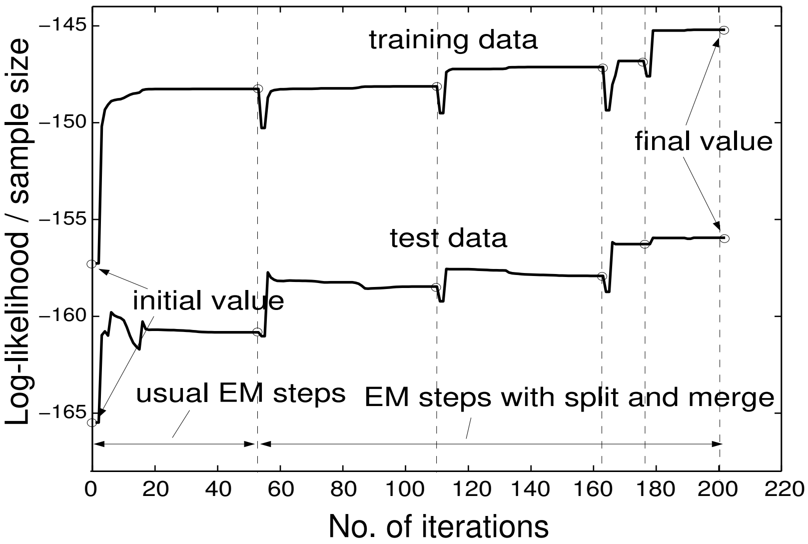


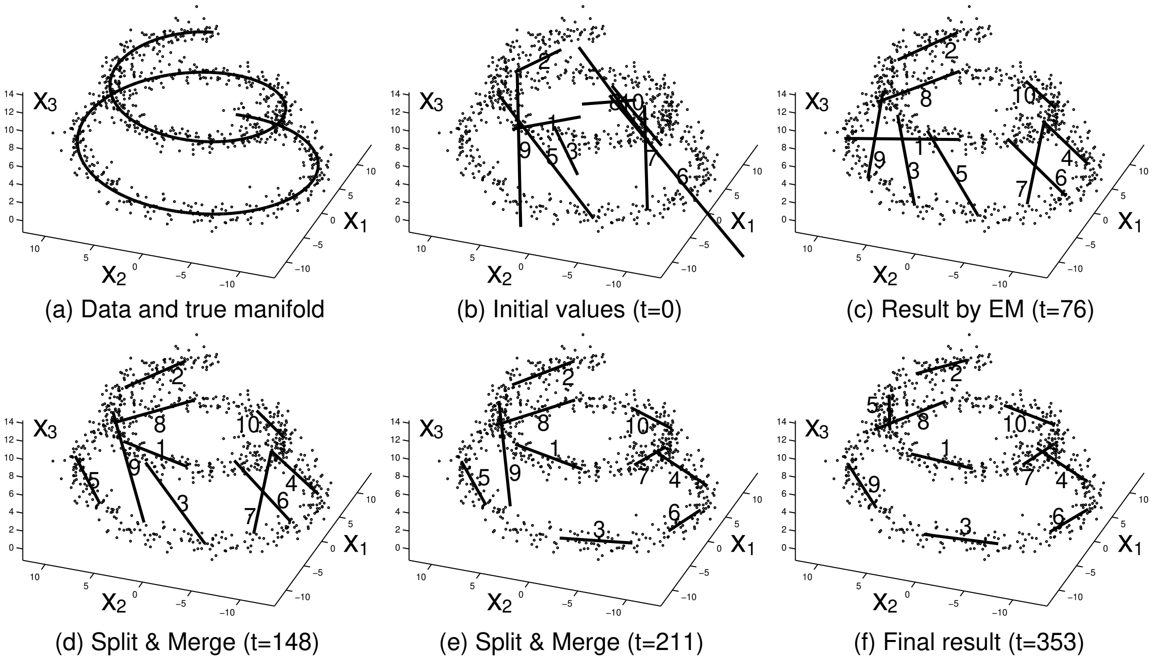
Figure 3: Trajectories of log-likelihood. The upper (lower) result corresponds to the training (test) data.

Figure 3, Ueda

Table 2: The number of EM-steps.

	EM	DAEM	SMEM
mean	47	147	409
std	16	39	84
max	65	189	616
min	37	103	265

Table 2, Ueda



(a) Data and true manifold

(b) Initial values (t=0)

(c) Result by EM (t=76)

(d) Split & Merge (t=148)

(e) Split & Merge (t=211)

(f) Final result (t=353)

Figure 4: Results by the EM and SMEM algorithms for a one-dimensional manifold extraction problem. (a) True manifold and generated data, (b) initial estimate, (c) result by the EM algorithm, (d)-(e) examples of split and merge operations , and (f) the final result.

Figure 4, Ueda

Table 3: Log-likelihood/sample size.

	EM	SMEM
Training	-7.68 (0.151)	-7.26(0.017)
Test	-7.75 (0.171)	-7.33(0.032)

Table 3, Ueda

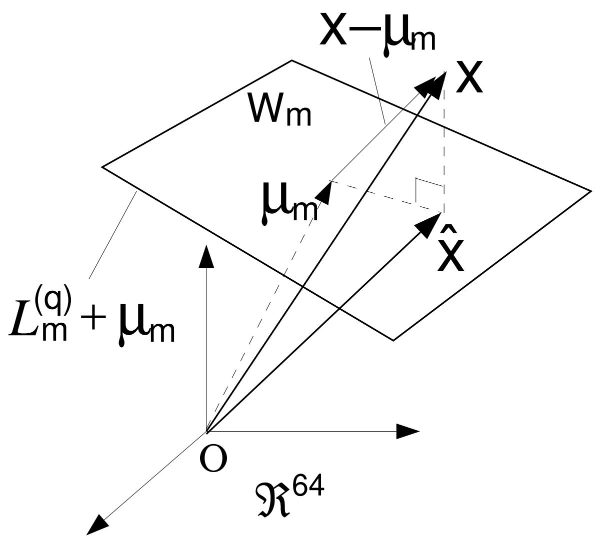


Figure 5: Illustration of image reconstruction.

Figure 5, Ueda

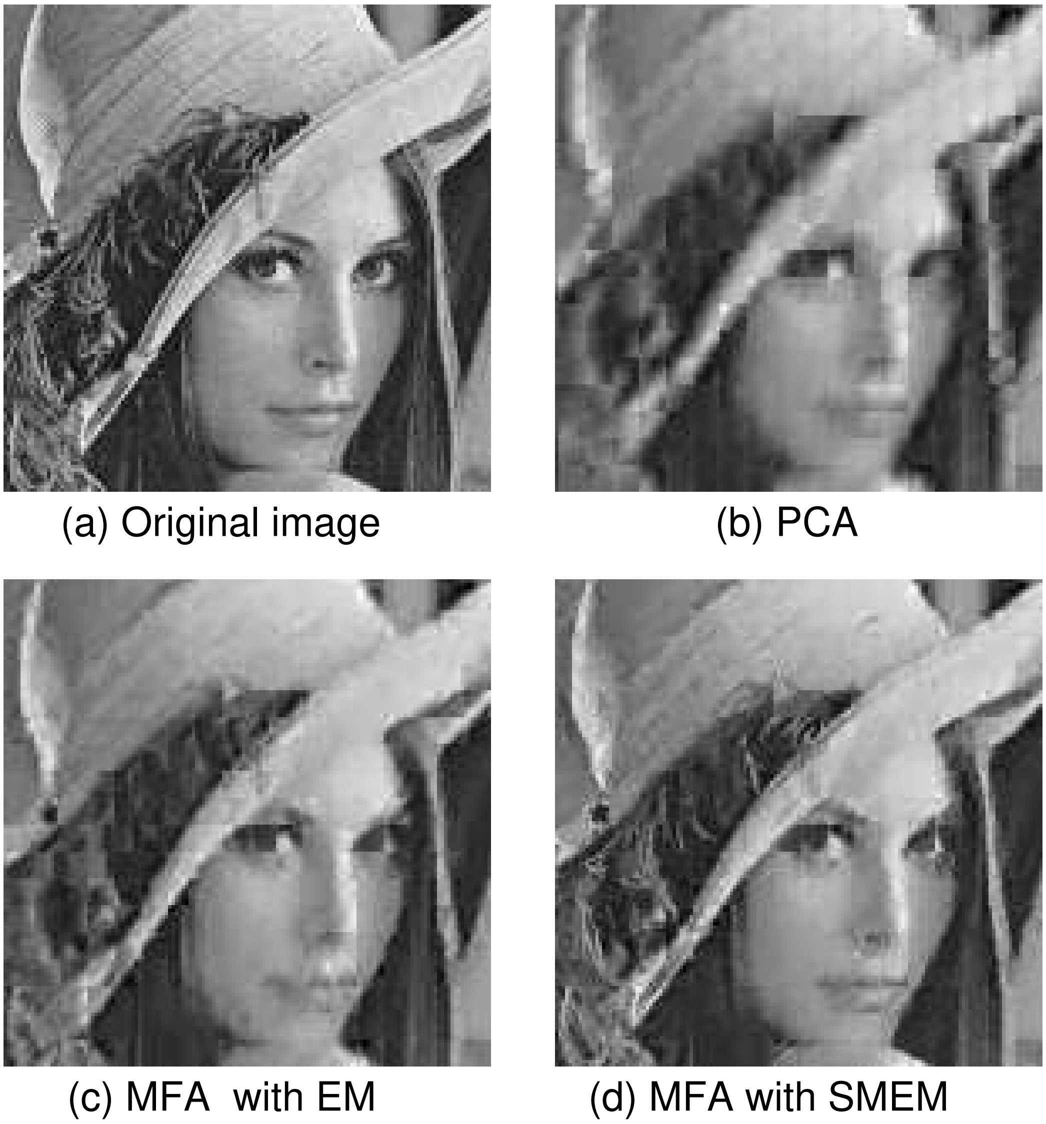


Figure 6: An example of image reconstruction. (a) Original image, (b) Result by PCA, (c)((d)) Result by an MFA model trained by the usual EM (SMEM) algorithm.

Figure 6, Ueda

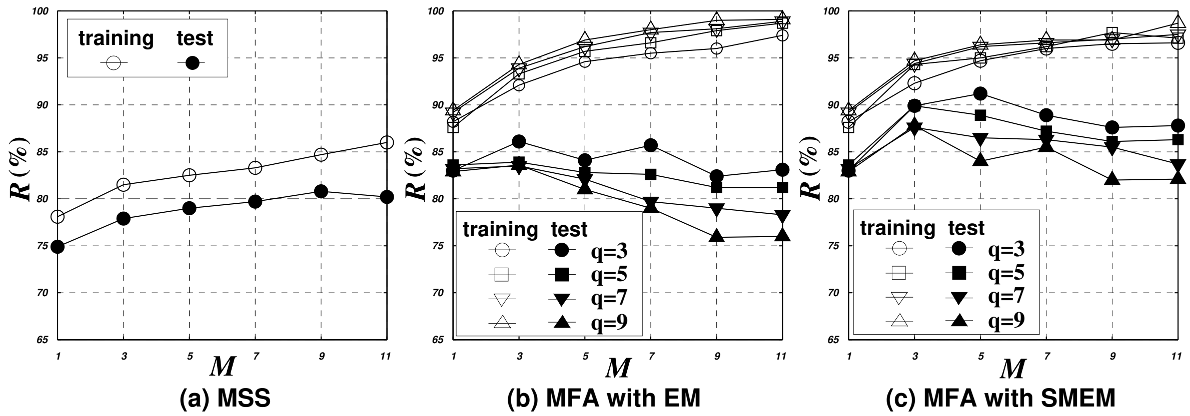


Figure 7: Recognition rates for hand-written digit data. (a) Results by MSS methods ($M = 1$ corresponds to the SS method). (b) Results by an MFA-based method trained by the EM algorithm. (c) Result by an MFA-based method trained by the SMEM algorithm.

Figure 7, Ueda