Journal of VLSI Signal Processing 26, 133–140, 2000

Split and Merge EM Algorithm for Improving Gaussian Mixture Density Estimates

NAONORI UEDA AND RYOHEI NAKANO
NTT Communication Science Laboratories, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237, Japan

ZOUBIN GHAHRAMANI AND GEOFFREY E. HINTON
Gatsby Computational Neuroscience Unit, University College London, 17 Queen Square, London WC1N 3AR, UK

Received March 25, 1999; Revised September 28, 1999

Abstract. The EM algorithm for Gaussian mixture models often gets caught in local maxima of the likelihood which involve having too many Gaussians in one part of the space and too few in another, widely separated part of the space. We present a new EM algorithm which performs split and merge operations on the Gaussians to escape from these configurations. This algorithm uses two novel criteria for efficiently selecting the split and merge candidates. Experimental results on synthetic and real data 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.

1. Introduction

Gaussian mixtures have been extensively used in the field of statistical pattern recognition including neural networks [1–4]. The EM algorithm [5] has been well known as a convenient and efficient tool to iteratively compute the maximum likelihood estimates of Gaussian mixtures. There are, however, two serious problems in practice: singularities and local maxima. Although these problems have been pointed out by many researchers, the best way to solve them in practice is still an open question.

Ormoneit and Tresp [3] have recently proposed some sophisticated regularization methods to solve the singularity problem. Regarding the local maximum problem, two of the authors have proposed the deterministic annealing EM (DAEM) algorithm [6, 7], where a modified posterior probability parameterized by *temperature* is derived to avoid local maxima. However, in the case of Gaussian mixture density estimation, local maxima arise when there are too many Gaussians in one part of the space and too few in another. It is not possible to move a Gaussian from the overpopulated region to the underpopulated region without passing through positions that give lower likelihood. We therefore introduce a discrete move that simultaneously merges two Gaussians in an overpopulated region and splits a Gaussian in an underpopulated region without changing the number of Gaussians.

The idea of split and merge operations has been successfully applied to clustering or vector quantization (e.g., [8]). In this paper, we try to incorporate the split and merge operations into the EM algorithm for Gaussian mixture density estimates to overcome the local maxima problem. New criteria presented in this paper can efficiently select the split and merge candidates. Although the proposed method, unlike the DAEM algorithm, is limited to mixture models, we show experimentally that our split and merge EM algorithm obtains better solutions than the DAEM algorithm.

2. Gaussian Mixture Density Estimation via the EM Algorithm

The probability density function (pdf) of a finite Gaussian mixture is

$$p(\boldsymbol{x}; \Theta) = \sum_{m=1}^{M} \alpha_m g(\boldsymbol{x}; \boldsymbol{\mu}_m, \Sigma_m), \qquad (1)$$

where $\alpha_m,\ m = 1, \ldots, M$ are mixing proportions and satisfy

$$\alpha_m \geq 0 \quad \text{and} \quad \sum_{m=1}^{M} \alpha_m = 1. \tag{2}$$

The $g(\boldsymbol{x}; \boldsymbol{\mu}_m, \Sigma_m)$ is a d-dimensional normal density corresponding to the mth component given by:

$$g(\boldsymbol{x}; \boldsymbol{\mu}_m, \Sigma_m) = (2\pi)^{-\frac{d}{2}} \det(\Sigma_m)^{-\frac{1}{2}} \times \exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_m)^T \Sigma_m^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_m)\right\}.$$

Here $\det(A)$ is the determinant of matrix A and T denotes the transpose operation. A set of unknown parameters, in this case, is $\Theta = \{(\alpha_m, \boldsymbol{\mu}_m, \Sigma_m),\ m = 1, \ldots, M\}$.

Given a set of iid data $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, the maximum likelihood estimate of the unknown parameters Θ is efficiently obtained by the EM algorithm [5]. In the EM algorithm, the parameters Θ are iteratively estimated by using two steps, E (for Expectation) and M (for Maximization). The E-step computes the expectation of the complete data log-likelihood using the posterior probability that $\boldsymbol{x}$ belongs to the mth component based on the current parameters $\Theta^{(t)}$:

$$Q\big(\Theta \mid \Theta^{(t)}\big) = \sum_{\boldsymbol{x} \in \mathcal{X}} \sum_{m=1}^{M} P\big(m \mid \boldsymbol{x}; \Theta^{(t)}\big) \log \alpha_m g(\boldsymbol{x}; \boldsymbol{\mu}_m, \Sigma_m), \tag{3}$$

where

$$P\big(m \mid \boldsymbol{x}; \Theta^{(t)}\big) = \frac{\alpha_m^{(t)} g\big(\boldsymbol{x}; \boldsymbol{\mu}_m^{(t)}, \Sigma_m^{(t)}\big)}{\sum_{l=1}^{M} \alpha_l^{(t)} g\big(\boldsymbol{x}; \boldsymbol{\mu}_l^{(t)}, \Sigma_l^{(t)}\big)}. \tag{4}$$

Next, the M-step maximizes this Q function with respect to Θ to estimate the new parameter values $\Theta^{(t+1)}$. More specifically, we rewrite (3) as

$$Q\big(\Theta \mid \Theta^{(t)}\big) = \sum_{m=1}^{M} \sum_{\boldsymbol{x} \in \mathcal{X}} P\big(m \mid \boldsymbol{x}; \Theta^{(t)}\big) \log \alpha_m + \sum_{m=1}^{M} \sum_{\boldsymbol{x} \in \mathcal{X}} P\big(m \mid \boldsymbol{x}; \Theta^{(t)}\big) \log g(\boldsymbol{x}; \boldsymbol{\mu}_m, \Sigma_m). \tag{5}$$

Then, by maximizing the first term of the right hand side of (5) with respect to α_m subject to (2), we have

$$\alpha_m^{(t+1)} = \frac{1}{N} \sum_{\boldsymbol{x} \in \mathcal{X}} P\big(m \mid \boldsymbol{x}; \Theta^{(t)}\big). \tag{6}$$

Next, by maximizing the second term of the right hand side of (5) with respect to $\boldsymbol{\mu}_m$ and Σ_m, respectively, we have

$$\boldsymbol{\mu}_m^{(t+1)} = \frac{\sum_{\boldsymbol{x} \in \mathcal{X}} \boldsymbol{x} P\big(m \mid \boldsymbol{x}; \Theta^{(t)}\big)}{\sum_{\boldsymbol{x} \in \mathcal{X}} P\big(m \mid \boldsymbol{x}; \Theta^{(t)}\big)}, \tag{7}$$

$$\Sigma_m^{(t+1)} = \frac{\sum_{\boldsymbol{x} \in \mathcal{X}} \big(\boldsymbol{x} - \boldsymbol{\mu}_m^{(t+1)}\big)\big(\boldsymbol{x} - \boldsymbol{\mu}_m^{(t+1)}\big)^T P\big(m \mid \boldsymbol{x}; \Theta^{(t)}\big)}{\sum_{\boldsymbol{x} \in \mathcal{X}} P\big(m \mid \boldsymbol{x}; \Theta^{(t)}\big)}. \tag{8}$$

To prevent the covariance from being singular, the following update rule based on the Bayesian regularizarion is available [3]:

$$\Sigma_m^{(t+1)} = \frac{\sum_{\boldsymbol{x} \in \mathcal{X}} \big(\boldsymbol{x} - \boldsymbol{\mu}_m^{(t+1)}\big)\big(\boldsymbol{x} - \boldsymbol{\mu}_m^{(t+1)}\big)^T P\big(m \mid \boldsymbol{x}; \Theta^{(t)}\big) + \lambda I_d}{\sum_{\boldsymbol{x} \in \mathcal{X}} P\big(m \mid \boldsymbol{x}; \Theta^{(t)}\big) + 1}, \tag{9}$$

where I_d is the d-dimensional unit matrix and λ is a regularization constant determined by some validation data.

3. Split and Merge EM Algorithm

3.1. *The Algorithm*

Let Θ^* denote the parameter values estimated by the usual EM algorithm. Then after the EM algorithm converged, (3) can be rewritten in the form of a direct sum:

$$Q^* = Q_i^* + Q_j^* + Q_k^* + \sum_{m, m \neq i,j,k} Q_m^*, \tag{10}$$

where

$$Q_m^* = \sum_{\boldsymbol{x} \in \mathcal{X}} P(m \mid \boldsymbol{x}; \Theta^*) \log \alpha_m^* g(\boldsymbol{x}; \boldsymbol{\mu}_m^*, \Sigma_m^*). \tag{11}$$

We then try to increase the first term of the right-hand side of (10) by merging the ith and jth Gaussians to produce the i'th Gaussian, and splitting the kth Gaussian into the j'th and k'th Gaussians. The split and merge operations are simultaneously performed so that the number of componets is unchanged. That is, we note that our goal here is not to solve the model selection problem, but to solve the local maxima problem.

To reestimate the parameters of these new Gaussians, we have to initialize the parameters corresponding to them using Θ^*.

The initial parameter values for the merged i'th Gaussian can be set as:

$$\alpha_{i'} = \alpha_i^* + \alpha_j^* \tag{12}$$

$$\theta_{i'} = \frac{\theta_i \alpha_i^* + \theta_j^* \alpha_j^*}{\alpha_i^* + \alpha_j^*}, \tag{13}$$

where θ_i corresponds to $\boldsymbol{\mu}_i$ and Σ_i. Noting that from (6) it follows

$$\alpha_l^* = \frac{1}{N} \sum_{\boldsymbol{x} \in \mathcal{X}} P(l \mid \boldsymbol{x}; \Theta^*),$$

one can see that the initial values of mean vector and covariance matrix are generated by a linear combination of the original ones before merge weighted by the posteriors, and this might be intuitively reasonable. Moreover, Guassian with parameters given by Eqs. (12) and (13) is, among all possible Gaussians, the one minimizing the Kullback-Leibler divergence to the density given by the ith and jth Guassians.

On the other hand, as for the j'th and k'th Gaussians, we initialize

$$\alpha_{j'} = \alpha_{k'} = \frac{1}{2}\alpha_k^* \tag{14}$$

$$\Sigma_{j'} = \Sigma_{k'} = \det(\Sigma_k^*)^{\frac{1}{d}} I_d. \tag{15}$$

That is, each covariance matrix is initialized as a unit matrix with the same volume as Σ_k^*. The mean vectors $\boldsymbol{\mu}_{j'}$ and $\boldsymbol{\mu}_{k'}$ are determined by performing the K-means algorithm on the data that has the highest posterior probability $P(k \mid \boldsymbol{x}; \Theta^*)$ under Gaussian k (e.g. the 10 data points most likely to have been generated from Gaussian k). Alternatively, we simply use some random perturbation vector $\boldsymbol{\epsilon}_m$, $m = 1, 2$ ($\|\boldsymbol{\epsilon}_m\| \ll \|\boldsymbol{\mu}_k^*\|$), and set $\boldsymbol{\mu}_{j'} = \boldsymbol{\mu}_k^* + \boldsymbol{\epsilon}_1$ and $\boldsymbol{\mu}_{k'} = \boldsymbol{\mu}_k^* + \boldsymbol{\epsilon}_2$.

The parameter reestimation for $m = i', j'$ and k' can be done by using EM steps shown in Sec. 2. Note that (4) should be replaced with (16) so that this reestimation does not affect the other Gaussians.

$$P(m' \mid \boldsymbol{x}; \Theta^{(t)}) = \frac{\alpha_{m'}^{(t)} g(\boldsymbol{x}; \boldsymbol{\mu}_{m'}^{(t)}, \Sigma_{m'}^{(t)})}{\sum_{l=i',j',k'} \alpha_l^{(t)} g(\boldsymbol{x}; \boldsymbol{\mu}_l^{(t)}, \Sigma_l^{(t)})} \times \sum_{m=i,j,k} P(m \mid \boldsymbol{x}; \Theta^*), \tag{16}$$

where $m' = i', j', k'$.

Clearly

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

always holds during the reestimation process. That is, the sum of posterior probabilities for the i'th, j'th, and k'th Gaussians becomes equal to the sum of posterior probabilities for the ith, jth, and kth Gaussians just before split and merge. By this, we can reestimate the parameters for the i'th, j'th, and k'th Gaussians consistently without affecting the other Gaussians. Since the initial parameter values given by Eqs. (12)–(15) are often poor, these newly generated Guassians should be first trained with fixing the other Gaussians.

For convenience, we call this EM procedure the *partial EM procedure*. After this partial EM procedure, the usual EM steps described in Sec. 2, called the *full EM procedure*, are performed as a post processing. After these procedures, if Q is improved, then we accept the new estimate and repeat the above after setting the new paramters to Θ^*. Otherwise reject and go back to Θ^* and try another candidate. We summarize these procedures as follows:

[Split and Merge EM Algorithm]

1. Perform the usual EM updates by using (4), (6), (7) and (9). Let Θ^* and Q^* denote the estimated parameters and corresponding Q function value, 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 initial parameter settings based on Θ^*, perform the *partial EM procedure* for $\{i, j, k\}_c$ and then perform the *full EM procedure*. Let Θ^{**} be the obtained parameters and Q^{**} be the corresponding Q function value. 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 at Step 3, the other successive candidates are ignored. There is therefore no guarantee that the split and the 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 for the partial EM steps, it is clear that the SMEM algorithm still maintain the global convergence properties of the EM algorithm.

3.2. Split and Merge Criteria

Each of the split and merge candidates can be evaluated by its Q function value after Step 3 of the split and merge EM algorithm mentioned in Sec. 3.1. However, since there are so many candidates, some reasonable criteria for ordering the split and merge candidates should be utilized to accelerate the split and merge EM algorithm.

In general, when there are many data points each of which has almost equal posterior probabilities for any two Gaussians, it can be thought that these two Gaussians might be merged. To numerically evaluate this, we define the following merge criterion:

$$J_{merge}(i, j; \Theta^*) = \mathbf{P}_i(\Theta^*)^T \mathbf{P}_j(\Theta^*), \qquad (17)$$

where $\mathbf{P}_i(\Theta^*) = (P(i \mid \boldsymbol{x}_1; \Theta^*), \ldots, P(i \mid \boldsymbol{x}_N; \Theta^*))^T \in \mathcal{R}^N$ is the N-dimensional vector consisting of posterior probabilities for the ith Gaussian. Clearly, the ith and jth Gaussians with larger $J_{merge}(i, j; \Theta^*)$ should be merged.[1]

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

$$J_{split}(k; \Theta^*) = \int p_k(\boldsymbol{x}; \Theta^*) \log \frac{p_k(\boldsymbol{x}; \Theta^*)}{g(\boldsymbol{x}; \boldsymbol{\mu}_k^*, \Sigma_k^*)} d\boldsymbol{x}, \qquad (18)$$

which is the distance between two distributions: the local data density $p_k(\boldsymbol{x})$ around the kth Gaussian and the kth Gaussian density specified by the current parameter estimate $\boldsymbol{\mu}_k^*$ and Σ_k^*. The local data density is defined as:

$$p_k(\boldsymbol{x}; \Theta^*) = \frac{\sum_{n=1}^N \delta(\boldsymbol{x} - \boldsymbol{x}_n) P(k \mid \boldsymbol{x}_n; \Theta^*)}{\sum_{n=1}^N P(k \mid \boldsymbol{x}_n; \Theta^*)}. \qquad (19)$$

This is a modified empirical distribution weighted by the posterior probability so that the data around the kth Gaussian are focused. Note that when the weights are equal, i.e., $P(k \mid \boldsymbol{x}; \Theta^*) = 1/M$, (19) is the usual empirical distribution, i.e., $p_k(\boldsymbol{x}; \Theta^*) = (1/N) \sum_{n=1}^N \delta(\boldsymbol{x} - \boldsymbol{x}_n)$. Since it can be thought that the Gaussian with the largest $J_{split}(k; \Theta^*)$ has the worst estimate of the local density, we should try to split it.

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

4. Experiments

First, we show the results of two-dimensional synthetic data to visually demonstrate the usefulness of the split and merge operations. Figure 1(a) shows the true Gaussian mixture distribution with a common covariance ($\Sigma = 0.1$ I) and a synthetically generated data set ($N = 300$) from the distribution. The usual EM algorithm converged to the local maximum solution shown in Fig. 1(c). However, our split and merge EM algorithm converges to the superior solution shown in Fig. 1(f) very close to the true one. Figure 1(d) and (e) show two of accepted split and merge results. Note that t does not include the rejected estimation steps.

The 1st Gaussian shown in Fig. 1(c) was further split as shown in Fig. 1(d). This seems to be a redundant split, but as shown in Fig. 1(e) they are successfully merged and the original two Gaussians (1st and 6th Gaussians in Fig. 1(c)) were improved as shown in Fig. 1(e). This result indicates that the split and merge operations not only play an important role to appropriately assign the number of Gaussians in a local data space, but can also contribute to better estimate the Gaussian parameters themselves. The log-likelihood values normalized by sample size were −2.210 (EM) and −2.162 (SMEM) for training data, while −2.198 (EM) and −2.223 (SMEM) for test data. The log-likelihood values were actually improved for both training and test data.

Next, we tested the proposed algorithm using high-dimensional real data where the local maxima make the optimization difficult. We used 20-dimensional feature vectors which were extracted from the facial images (photographs). Data size was 103 for training

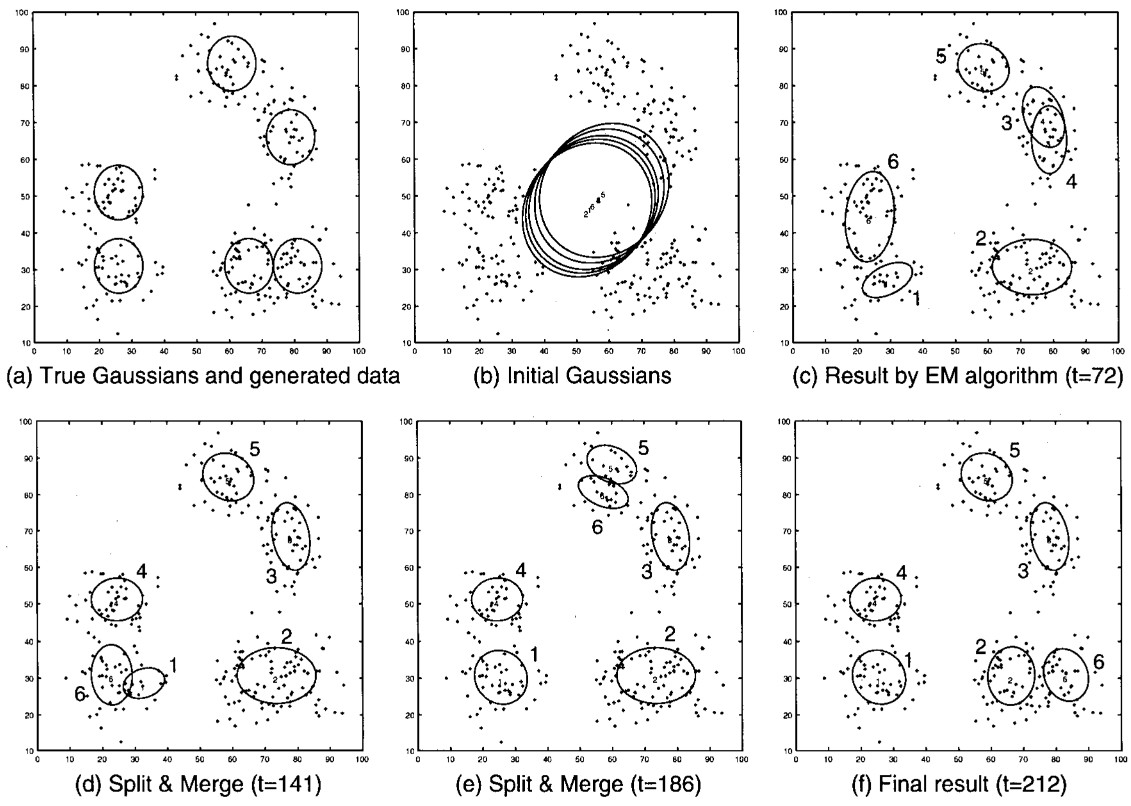


Figure 1. Results by the EM and SMEM algorithms for a two-dimensional Gaussian mixture density estimation problem. (a) Contours of true Gaussians, (b) initial density, (c) result by the EM algorithm, (d) and (e) examples of split and merge operations by the SMEM algorithm, and (f) the final result.

($\mathcal{X}$) and 103 for test ($\mathcal{X}'$). We performed three algorithms (EM, DAEM, and split and merge EM). Note that the result of the EM algorithm can be obtained at Step 1 in the split and merge EM algorithm as mentioned in Sec. 3.1. In this experiment, we designated $M = 5$ and a diagonal covariance model (i.e., $\Sigma_m = \text{diag}(\sigma^2_{m1}, \ldots, \sigma^2_{m20}),\ m = 1, \ldots, 5$) for each Gaussian. We initialized mean vectors by using the K-means algorithm and initialized covariance matrices as 20-dimensional unit matrices. Since the K-means algorithm depends on the initial mean vectors, we initialized 10 times. For each initialization, we tested the three algorithms.

Table 1 summarizes the log-likelihood values for each of training and test sets obtained by running each of three algorithms ten times. These values are normalized by the data size. The proposed algorithm is referred to as 'SMEM' in Table 1. Since the test data were not used to obtain Θ^* and the training data size was not large compared to the dimensionality, the log-likelihood values for the test data were much smaller than those for the training data. However, the proposed algorithm certainly improved the log-likelihood not only for the training data, but also for the test data. Moreover, the worst solution found by the split and merge EM algorithm was better than the best solutions

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

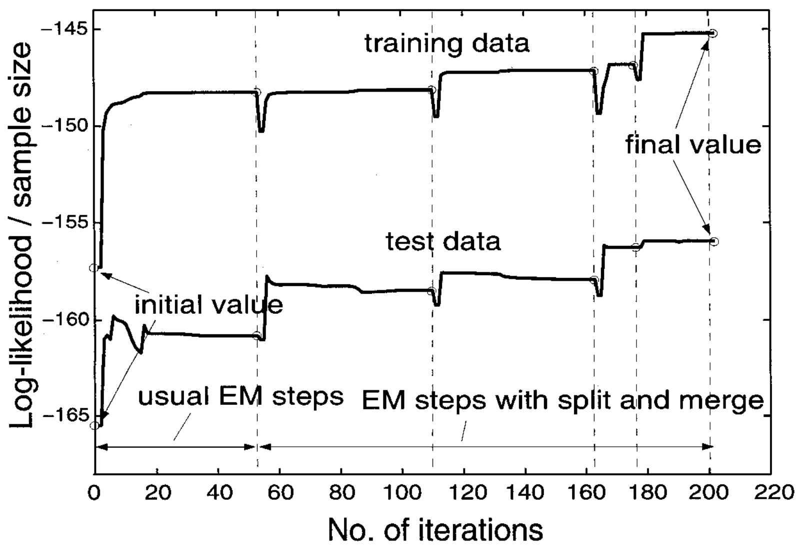


Figure 2. Trajectories of log-likelihood. The upper (lower) result corresponds to the training (test) data.

found by the other algorithms on both training and test data.

From Table 1, one can see that the SMEM algorithm obtained the best result among three algorithms (EM, DAEM, and SMEM). In other words, the DAEM algorithm may less effective than the SMEM algorithm, at avoiding local maxima associated with the mixture models. Although the DAEM algorithm, unlike the SMEM algorithm, can be used for any models, the SMEM algorithm is better suitable for the Gaussian mixture density estimation problem than the DAEM algorithm.

Figure 2 shows log-likelihood value trajectories during the estimation process. Actually, there are some branching trajectories each of which corresponds to the split and merge candidate executed at Step 3. However, for clarity, only trajectories corresponding to the accepted estimation process at Step 3 are shown in Fig. 2. In Fig. 2, the upper (lower) trajectory corresponds to the training (test) data.

The lower one was obtained by successively computing the log-likelihood using the current parameter estimate and the test data during the estimation process. Dotted lines in Fig. 2 denote the starting points of Step 2; therefore, Step 2 was executed five times in the run. The average number of accepted split and merge operations (i.e., the average counts that Step 2 was executed) was 4.7 (STD = 0.8).

Note that it is due to the initialization at Step 3 that the log-likelihood decreases just after the split and merge in Fig. 2. Since newly merged and split Gaussians were, as described in Sec. 3.1, initialized in some ad hoc manner, the log-likelihood values corresponding to these initial parameter values were usually smaller than those just before the split and merge. In the lower trajectory, unlike the upper one, the log-likelihood did not necessarily increase monotonically in each reestimation process at Step 3, which may be natural because the trajectory was for the test data. However, comparing the convergence points at Step 3 marked by the '$\circ$' symbol in Fig. 2, one can see that the successive split and merge operations improved the log-likelihood for the test data, as we expected.

Table 2 compares the number of iterations (no. of EM steps) executed by the three algorithms. Note that in the SMEM algorithm, the EM-steps corresponding to rejected split and merge operations are not counted. 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 work very well. Therefore, the SMEM algorithm was about $155 \times 1.8/47 \simeq 6$ times slower than the original EM algorithm.

Table 2. The number of EM-steps.

	EM	DAEM	SMEM
Mean	47	147	155
Std	16	39	44
Max	65	189	219
Min	37	103	109

We further fairly compared the SMEM and EM algorithms under the same computational complexity. That is, we performed the EM algorithm six times for every SMEM run using the same 20-dimensional data. We repeated this comparison ten times. Table 3 shows the log-likelihood values for each of training and test sets over the ten trials. We confirmed that the SMEM algorithm obtained still better results than the EM algorithm for both training and test sets.

5. Conclusion

We have shown how simultaneous split and merge operations can be used to move Gaussians from regions of the space in which there are too many Gaussians to regions in which there are too few. Such moves cannot be accomplished by methods that continuously move Gaussians through intermediate locations because the likelihood is 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. 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 a mixture of Gaussians.

To make the split and merge method efficient we have introduced criteria for deciding which splits and merges to consider and have shown that these criteria work well for a low-dimensional synthetic dataset and for a high-dimensional real dataset. Our split and merge EM algorithm consistently outperforms standard EM, and it also consistently outperforms a deterministic annealing version of EM that initially keeps the variances of the Gaussians large so that it is easier to move them across regions of the space that separate data clusters.

Recently, we have extended the SMEM algorithm so as to make it applicable to general mixture models including more sophisticated mixture models such as mixtures of latent variable models (e.g., mixtures of probabilistic principal component analyzers [9] or mixtures of factor analyzers [10]). These extensions are described in Refs. [11, 12].

Note

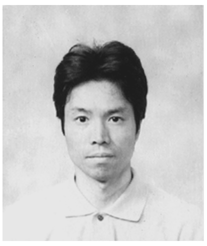
1. This merge criterion favors merges with larger classes. To avoid this bias, a modified ciriterion:

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

can be used. In our experiments, however, we used Eq. (17) for simplicity and the merge criterion did work well as shown later.

References

1. G. MacLachlan and K. Basford, *Mixture Models: Inference and Application to Clustering*, Marcel Dekker, 1988.
2. N. Kambhatla and T.K. Leen, “Classifying with Gaussian Mixtures and Clusters,” in *Advances in Neural Information Processing Systems 7*, Cambridge MA: MIT Press, 1995, pp. 681–687.
3. D. Ormoneit and V. Tresp, “Improved Gaussian Mixture Density Estimates Using Bayesian Penalty Terms and Network Averaging,” in *Advances in Neural Information Processing Systems 8*, D.S. Touretzky, G. Tesauro and T.K. Leen (eds.), Cambridge MA: MIT Press, 1996, pp. 542–548.
4. L. Rabiner and Juang Biing-Hwang, *Fundamentals of Speech Recognition*, PTR Prentice-Hall, 1993.
5. A.P. Dempster, N.M. Laird, and D.B. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *Journal of Royal Statistical Society B*, vol. 39, 1977, pp. 1–38.
6. N. Ueda and R. Nakano, “Deterministic Annealing Variant of the EM Algorithm,” *Advances in Neural Information Processing Systems 7*, D.S. Touretzky, G. Tesauro and T.K. Leen (eds.), Cambridge MA: MIT Press, 1995, pp. 545–552.
7. N. Ueda and R. Nakano, “Deterministic Annealing EM Algorithm,” *Neural Networks*, vol. 11, 1998, pp. 271–282.
8. N. Ueda and R. Nakano, “A New Competitive Learning Approach Based on an Equidistortion Principle for Designing Optimal Vector Quantizers,” *Neural Networks*, vol. 7, no. 8, 1994, pp. 1211–1227.
9. M.E. Tipping and C.M. Bishop, “Mixtures of Probabilistic Principal Component Analysers,” Tech. Rep. NCRG-97-3, Aston Univ. Birmingham, UK, 1997.
10. Z. Ghahramani and G.E. Hinton, “The EM Algorithm for Mixtures of Factor Analyzers,” Tech. Report CRG-TR-96-1, Univ. of Toronto, 1997. `http://www.gatsby.ucl.ac.uk/~zoubin/papers/tr-96-1.ps.gz.`
11. N. Ueda, R. Nakano, Z. Ghahramani, and G.E. Hinton, “SMEM Algorithm for Mixture Models,” in *Advances in Neural Information Processing Systems 11*, M.S. Kearns, S.A. Solla and D.A. Cohn (eds.), Cambridge MA: MIT Press, 1999, pp. 599–605.
12. N. Ueda, R. Nakano, Z. Ghahramani, and G.E. Hinton, “SMEM Algorithm for Mixture Models,” *Neural Computation*, MIT Press, to appear.



Naonori Ueda received the B.Eng., M.Eng., and Ph.D. degrees in Communication Engineering from Osaka University in 1982, 1984, and 1992, respectively. In 1984, he joined the Electrical Communication Laboratories, NTT, Kanagawa, japan. From 1993 to 1994,

he was a visiting researcher at Purdue University, West Lafayette, USA. He is currently a group leader at Emergent Learning Research Group, NTT Communication Science Laboratories, Kyoto, Japan. He is also a guest associate professor of Nara Advanced Institute of Science and Technology. Dr. Ueda received Japanese Neural Network Society (JNNS) Research Award and the 12th Telecommunication Advancement Foundation Award (Telecom System Technology Prize) in 1995 and 1997, respectively. He works on statistical learning theory and its application to pattern recognition.
ueda@cslab.kecl.ntt.co.jp

Ryohei Nakano received his B.Eng and PhD in Mathematical Engineering from the University of Tokyo in 1971 and 1990 respectively. He is currently a Professor at the Department of Intelligence and Computer Science at Nagoya Institute of Technology, as well as a Guest Professor at Nara Advanced Institute of Science and Technology, Japan. From 1971 to 1999 he worked for NTT by developing a distributed database system architecture and doing research on artificial intelligence with the focus on neural and doing research on artificial intelligence with the focus on neural and evolutionary computation. He got the 12th Telecom. System Technology Prize in 1997, and two Paper Awards of Information Processing Society of Japan and Japanese Society for Artificial Intelligence in 1997 and 1999 respectively. His main interest is in theories and algorithms on learning, adaptation, and optimization.

Zoubin Ghahramani received a BA degree in Cognitive Science and a BSEng degree in Computer Science in 1990 from the University of Pennsylvania. He obtained a Ph.D. in Cognitive Neuroscience from the Massachusetts Institute of Technology with a Fellowship from the McDonnell-Pew Foundation. He is currently a Lecturer at the Gatsby Computational Neuroscience Unit at University College London, with cross-appointments in the Departments of Computer Science and Psychology. His research interests include studying how the brain controls movement and integrates information from different senses, developing computational theories of learning in biological and artificial systems, and applying probabilistic approaches to computer intelligence.

Geoffrey Hinton received his BA in experimental psychology from Cambridge in 1970 and his PhD in Artificial Intelligence from Edinburgh in 1978. He is currently the director of the Gatsby Computational neuroscience Unit at University College London. He does research on ways of using neural networks for learning, memory, perception and symbol processing and has over 150 publications in these areas. He was one of the researchers who introduced the back-propagation algorithm. His other contributions to neural network research include Boltzmann machines, distributed representations, time-delay neural nets, mixture of experts and Helmholtz machines. His current main interest is learning procedures for neural networks with rich sensory input.