PERGAMON

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition 37 (2004) 131–144

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Competitive EM algorithm for finite mixture models

Baibo Zhang, Changshui Zhang*, Xing Yi

State Key Laboratory of Intelligent Technology and Systems, Department of Automation, Tsinghua University, Beijing 100084, China

Received 27 December 2002; received in revised form 15 April 2003; accepted 15 April 2003

Abstract

In this paper, we present a novel competitive EM (CEM) algorithm for finite mixture models to overcome the two main drawbacks of the EM algorithm: often getting trapped at local maxima and sometimes converging to the boundary of the parameter space. The proposed algorithm is capable of automatically choosing the clustering number and selecting the "split" or "merge" operations efficiently based on the new competitive mechanism we propose. It is insensitive to the initial configuration of the mixture component number and model parameters.

Experiments on synthetic data show that our algorithm has very promising performance for the parameter estimation of mixture models. The algorithm is also applied to the structure analysis of complicated Chinese characters. The results show that the proposed algorithm performs much better than previous methods with slightly heavier computation burden.

Keywords: Clustering; EM algorithm; Competitive; Mixture models; SMEM; CEM

1. Introduction

In the probabilistic model construction for univariate and multivariate data, finite mixture models have been widely used due to their great flexibility and power [1]. The capability of representing arbitrary complex probability density functions (pdfs) enables it to have many applications not only in unsupervised learning fields [2], but also in (Bayesian) supervised learning scenarios or in parameter estimation of class-conditional pdfs [3].

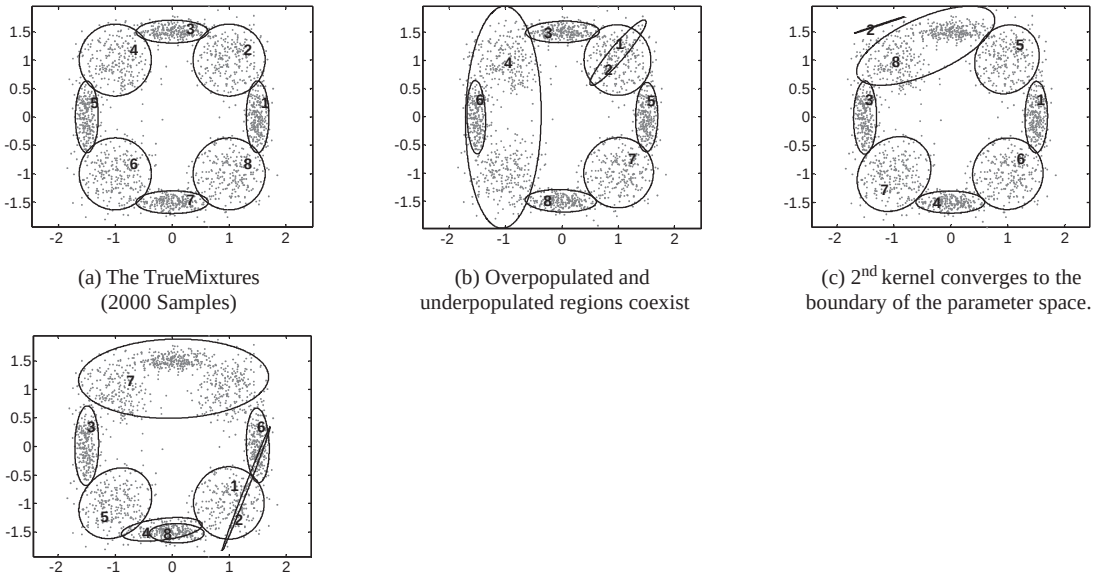
Expectation-maximization (EM) algorithm is a widely used class of iterative algorithms for maximum likelihood (ML) or maximum a posteriori (MAP) estimation in problems with incomplete data, e.g. fitting mixture models to observed data [4,5]. However, because of its greedy nature, the EM algorithm still has some defects: it is sensitive to the initial configuration and usually gets stuck at local maxima; for mixtures, it cannot choose the component number automatically and sometimes converges to the boundary of the parameter space. Most of the time, when trapped at a local maximum, the components' distribution in data space is inappropriate—overpopulated regions and underpopulated regions coexist. The components cannot move to the expected positions due to the difficulty of passing through some low likelihood regions [6].

Some defects of EM are illustrated in Fig. 1. The samples depicted in Fig. 1a come from 8-component two-dimensional *Gaussian mixture models* (GMM) and the initial configuration is randomly set. We apply EM to the estimation of the parameters of GMM with this data set and run it several times. Some typical local maxima are recorded and depicted in Figs. 1b–d. In some cases as in Fig. 1b, crowded regions (kernels 1 and 2) and sparse regions (kernel 4) coexist when EM encounters the local maximum; Fig. 1c shows that EM sometimes converges to the boundary of the parameter space (kernel 2); Fig. 1d depicts a situation when the above two cases appear at the same time.

* Corresponding author. Tel.: +86-10-6278-2447; fax: +86-10-6278-6911.

E-mail addresses: zhangbb99g@mails.tsinghua.edu.cn (B. Zhang), zcs@mail.tsinghua.edu.cn (C. Zhang), yixing97@mails.tsinghua.edu.cn (X. Yi).

0031-3203/04/$30.00 © 2003 Published by Elsevier Ltd on behalf of Pattern Recognition Society.
doi:10.1016/S0031-3203(03)00140-7



(a) The TrueMixtures (2000 Samples)

(b) Overpopulated and underpopulated regions coexist

(c) 2nd kernel converges to the boundary of the parameter space.

(d) Problems of (b) and (c) coexist

Fig. 1. Drawbacks of the basic EM algorithm.

1.1. Related work

Many variants of the EM algorithm have been proposed to try to solve the problems of EM. Ueda and Nakano [7] presented one method of deterministic annealing EM algorithm (DAEM) [7]. By introducing the temperature parameters to modify the posteriori probability in the E-step, they provided the EM with the ability to escape the local maxima in some cases. However, the DAEM and other similar extensions of EM are useless with respect to the problem of inappropriate distribution of the components in data space when locally trapped. Ueda et al. proposed the split and merge EM (SMEM) algorithm to address this issue [6]. Without changing the component number, this method chooses the candidates to split and merge by some given criteria and attempts splitting and merging operations when locally trapped. It simultaneously escapes the local maxima in many situations and performs more efficiently than DAEM. However, this deterministic algorithm cannot automatically determine the component number. Shoham presented one robust clustering algorithm by creating a deterministic agglomeration EM (DAGEM) with multivariate t-distributions [8]. It was derived from the DAEM algorithm and achieved encouraging performance. Because the initial component number is much larger than the true number, the computation load is one to two orders of magnitude heavier than EM [8].

The methods of model selection can be divided into two families. One is based on a random sampling mechanism, e.g. Markov Chain Monte Carlo (MCMC) methods [9]. Roberts has used the reversible jump Markov Chain Monte Carlo (RJMCMC) method to partition the data and automatically choose the clustering number [10]. The global optimal solution can be achieved but the computation burden is too heavy. The other one is based on some deterministic criteria [1,11]. Figueiredo et al. have designed one such criterion similar to the *minimum message length* (MML) criterion [11]. Combining this criterion and some annihilation operation, this extension of EM that we will refer to as AEM overcomes most of the problems depicted in Fig. 1. It can also determine the clustering component number in the optimization procedure. However, when there is great disparity between the components' prior probabilities, it sometimes fails to find the global optimal solution [11].

1.2. The competitive EM algorithm

To overcome the drawbacks of EM and to develop a more robust clustering method, we propose a novel competitive EM algorithm (CEM), which includes these operations: the basic iterative steps of EM, split, merge and annihilation operations. The mixture models' state is evaluated by the criterion Figueiredo proposed [11]. Our algorithm has the following impressive advantages which are revealed by our experiments:

- CEM is insensitive to the initial configuration of the component number and model parameters.
- CEM will avoid converging to the boundary of parameter space by eliminating components with sparse samples.
- The competitive mechanism has been introduced to select the "split" and "merge" operations more efficiently

according to the new criterion we design. When the current model needs more components, CEM will choose the "split" operation with large probability and vice versa.
- CEM is a very robust clustering method with a slightly heavier computation cost, especially in very complicated cases.

The remainder of the paper is organized as follows: Section 2 briefly reviews finite mixture models and the EM Algorithm. Section 3 presents the CEM algorithm in detail. Section 4 reports the experiments on synthesis data and gives a performance analysis of our algorithm. Section 5 investigates an application of the CEM algorithm on the structure analysis of Chinese characters. Section 6 concludes the paper and discusses some related open issues.

2. Learning finite mixture models

2.1. Finite mixture models

It is said a d-dimensional random variable $x = [x_1, x_2, \ldots, x_d]^{\mathrm{T}}$ follows a k-component finite mixture distribution, if its probability density function can be written as

$$p(x|\theta) = \sum_{m=1}^{k} \alpha_m p(x|\theta_m), \tag{1}$$

where α_m is the prior probability of the mth component and satisfies

$$\alpha_m \geqslant 0 \quad \text{and} \sum_{m=1}^{k} \alpha_m = 1, \tag{2}$$

where θ_m is the parameter of the mth density model and $\theta = \{(\alpha_m, \theta_m),\ m = 1, 2, \ldots, k\}$ is the parameter set of mixture models. Different descriptions of $p(x|\theta_m)$ can be assigned to different kinds of mixture models. We focus on finite mixture models and demonstrate the mechanism of our algorithm by means of Gaussian mixture models.

2.2. The EM algorithm

EM has been widely used in the parameter estimation of finite mixture models. We review it briefly and introduce EM notations for the convenience of later description of CEM. Suppose that one set $\mathbf{Z}$ consists of observed data $\mathbf{X}$ and unobserved data $\mathbf{Y}$, $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$ and $\mathbf{X}$ are called *complete data* and *incomplete data*, respectively. The E-step calculates the complete data expected log-likelihood function defined by the so-called Q function,

$$Q(\theta, \hat{\theta}(t)) \equiv E[\log p(\mathbf{X}, \mathbf{Y}|\theta)|\mathbf{X}, \hat{\theta}(t)]. \tag{3}$$

The M-step updates the parameters by

$$\hat{\theta}(t+1) = \arg\max_{\theta} Q(\theta, \hat{\theta}(t)). \tag{4}$$

The EM algorithm performs the E- and M-steps iteratively, and the convergence is theoretically guaranteed [4].

3. The proposed competitive EM algorithm

3.1. Model selection criterion

To estimate the appropriate number of components, many deterministic criteria are proposed [1,11]. We use the same criterion presented by Figueiredo [11]:

$$\hat{\theta} = \arg\max_{\theta} L(\theta, \mathbf{X}),$$

$$L(\theta, \mathbf{X}) = \log p(\mathbf{X}|\theta) - \frac{N}{2} \sum_{m=1}^{k} \log\left(\frac{n\alpha_m}{12}\right) - \frac{k}{2} \log \frac{n}{12} - \frac{k(N+1)}{2}, \tag{5}$$

where N is the number of parameters specifying each component and n is the total number of training samples. Figueiredo testified that this criterion is better than some other criteria such as Rissanen's *minimum description length* (MDL) criterion, Schwarz's *Bayesian inference criterion* (BIC), *Laplace-empirical criterion* (LEC) and the *integrated classification likelihood* (ICL) criterion, etc. by experiment [11].

The criterion is derived from *minimum message length* criterion, defined as

$$\hat{\theta}_{\mathrm{MML}} = \arg\max_{\theta} \left\{\log p(\mathbf{X}|\theta) + \log p(\theta) - \frac{1}{2} \log|I(\theta)| - \frac{c}{2}\left(1 + \log\frac{1}{12}\right)\right\}, \tag{6}$$

where $I(\theta) \equiv -E[D_0^2 \log p(\mathbf{X}|\theta)]$ is the expected Fisher information matrix and c is the dimension of θ (the symbol D_0^2 denotes the matrix of second derivatives, or Hessian). Generally, the Fisher information matrix for mixture models cannot be calculated analytically so Figueiredo replaced it with the complete data-based Fisher information matrix $I_c(\theta) \equiv -E[D_0^2 \log p(\mathbf{Z}|\theta)]$ and deduced the criterion above [11].

3.2. Split and merge mechanism

When EM encounters local maxima, the components usually overpopulate in some regions, i.e. the model over-fits the data, but underpopulate in other regions. The difficulty of passing through some low likelihood regions prevents them from getting to the expected positions. To overcome this problem, we do the split or merge operation when EM has converged to a local maximum, thus the components' distribution can self-adapt, and split will take place where the components are too few and merge will take place where the components are too many. This design enables our algorithm to escape the local extrema by modifying the mixtures model structure, not by moving components through those low likelihood regions, which is of great difficulty.

Many non-Gaussian measurement functions have been used as the split criteria, e.g. Kurtosis, Negentropy, etc.

[12]. Ueda presented one merge criterion, defining that if the posterior probabilities of two components are very similar, the merge happens [6]. Some people intuitively define that merge mostly happens where the two components are very close to each other. The drawbacks of defining the split and merge criteria separately are apparent: the algorithm cannot choose the split or merge operation according to the current state, which will cause the repeated occurrence of futile split and merge operation. This phenomenon can be observed in our experiments and greatly affects the convergence speed.

3.2.1. Split and merge probabilities

We use *local Kullback divergence* to measure the distance between the local data density $f_m(x)$ and model density $p_m(x)$ of the mth component as in SMEM [6].

$$J(m;\theta) = \int f_m(x;\theta) \log \frac{f_m(x;\theta)}{p_m(x;\theta_m)}\,\mathrm{d}x, \tag{7}$$

where the local data density $f_m(x)$ is the modified *empirical distribution* weighted by the posterior probability and defined as

$$f_m(x;\theta) = \frac{\sum_{i=1}^{n} \delta(x - x_i) P(m|x_i;\theta)}{\sum_{i=1}^{n} P(m|x_i;\theta)}. \tag{8}$$

We arrange that the split probability of the mth component is in direct proportion to $J(m;\theta)$ and the merge probability of the mth and lth components is in inverse proportion to $J(m';\theta')$, where m' and θ' denote, respectively, the new index of merged component and new parameters of mixture models if the mth and lth components merge. Because the new parameters can be easily calculated if the merge operation occurs, the additional computation load for $J(m';\theta')$ is not heavy.

The split and merge probabilities are formulated by

$$P_{\text{split}}(m;\theta) = \frac{J(m;\theta)}{Z(\theta)}, \tag{9}$$

$$P_{\text{merge}}(m,l;\theta) = \frac{\beta/J(m';\theta')}{Z(\theta)}, \tag{10}$$

where $Z(\theta)$ is a normalized factor to make

$$\sum_{m=1}^{k} P_{\text{split}}(m;\theta) + \sum_{m=1}^{k}\sum_{l=m+1}^{k} P_{\text{merge}}(m,l;\theta) = 1 \tag{11}$$

and β is a constant determined by experiments.

3.2.2. Split and merge operations

The operation type and the candidates of split or merge components are sampled by Eqs. (9) and (10).

Split Operation: $k \to k+1$. Suppose that the mth component is chosen to split, CEM will generate two new components from the current samples in the mth component. We initialize the two new components parameters by *random initialization* method and optimize them by the basic EM algorithm. The *random initialization* method employed in this paper means that the mean value and the prior probability of each kernel are randomly set; the covariance of each component is set to be $\sigma_0^2 I$, where $\sigma_0 = 0.15 dScale$. $dScale$ denotes the scale of the data points, which is formulated by,

$$dScale = \max_{\{1\leqslant m\leqslant d\},\{1\leqslant i,j\leqslant n\}} |x_m^{(i)} - x_m^{(j)}|. \tag{12}$$

Merge operation: $k \to k-1$. Suppose that the mth and lth components are selected to merge, the parameters of the merged component can be calculated directly from the original mth and lth components parameters. As with most kinds of finite mixture models, this can be done easily. In GMM, the formula of calculating those parameters is as follows:

$$\begin{aligned}
\alpha_m^{(M)} &= \alpha_m + \alpha_l,\\
\mu_m^{(M)} &= (\alpha_m\mu_m + \alpha_l\mu_l)/\alpha_m^{(M)},\\
\Sigma_m^{(M)} &= \{\alpha_m[\Sigma_m + (\mu_m - \mu_m^{(M)})(\mu_m - \mu_m^{(M)})^{\mathrm{T}}]\\
&\quad + \alpha_l[\Sigma_l + (\mu_l - \mu_m^{(M)})(\mu_l - \mu_m^{(M)})^{\mathrm{T}}]\}/\alpha_m^{(M)}.
\end{aligned} \tag{13}$$

3.2.3. Acceptance probability

After the operation type and operation candidates are chosen by sampling according to the split and merge probabilities, the acceptance probability is proposed to prevent poor operation. It is calculated by

$$P_a = \min\left(\exp\left[\frac{L(\theta(t+1),\mathbf{X}) - L(\theta(t),\mathbf{X})}{\gamma}\right], 1\right), \tag{14}$$

where γ is a constant determined by experiments. In this way, if the operation increases the value of model evaluation function, it will be accepted definitely. However, if the operation decreases the value of model evaluation function, it will not be rejected directly, but accepted via certain probability. This mechanism enables the model to have certain jumping capability and simultaneously prevents the model from diverging to overly worse state. The larger γ, the stronger the jumping capability of the model. In our experiments, $\gamma = 10$.

3.3. Component annihilation mechanism

To prevent the algorithm from converging to the bounds of parameter space, the mechanism of component annihilation is introduced. The component m will be eliminated, if

$$n\alpha_m < N. \tag{15}$$

As Figueiredo mentioned in his paper [11], many or even all components may die simultaneously if k is too large in the initial step. We use the *component-wise* EM for *mixtures* (CEM^2) algorithm to prevent this in the annihilation operation [13]. This algorithm updates parameters of components sequentially, not simultaneously. When one component dies, the CEM^2 can increase the survival probability of other components. The computation load of CEM^2 is only slightly heavier than EM [11].

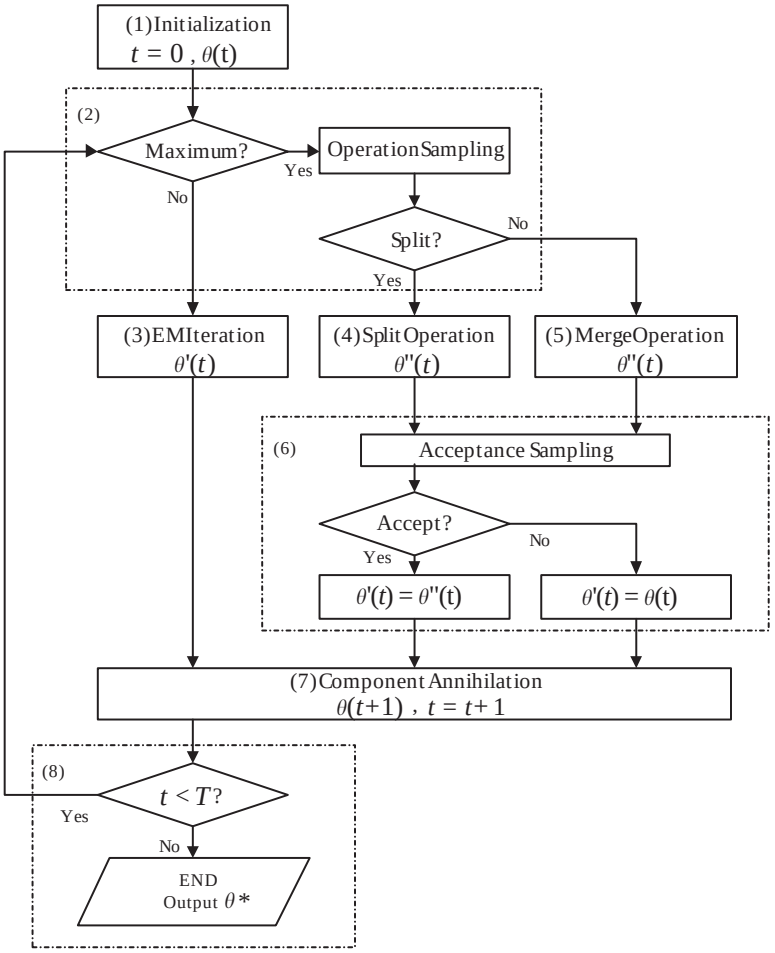


Fig. 2. Flow chart of CEM.

3.4. The competitive EM algorithm

In this section, we summarize the CEM algorithm and illustrate the flow as shown in Fig. 2.

(1) *Initialization*: Set k and the prior probabilities arbitrarily subject to Eq. (2) and set other parameters by the *random initialization* method. Set $t = 0$ (t is a counter).

(2) *Operation sampling*: If $\theta(t)$ is not a local maximum, go to (3); otherwise sample the operation type and split or merge candidates according to Eqs. (9) and (10). If the split operation is chosen, go to (4); otherwise go to (5).

(3) *EM iteration*: Implement the EM iteration once, denote the new parameters as $\theta'(t)$ and go to (7).

(4) *Split operation*: Generate two new components from the samples of the split candidate by the *random initialization* method and optimize them by EM. Denote the new parameters as $\theta''(t)$ and go to (6).

(5) *Merge operation*: Let the two candidates merge to one new component. Denote the new parameters as $\theta''(t)$ and go to (6).

(6) *Acceptance sampling*: Calculate the acceptance probability $P_u = \min(\exp\{[L(\theta''(t), \mathbf{X}) - L(\theta'(t), \mathbf{X})]/\gamma\}, 1)$. Sample $u \sim U[0, 1]$, if $u \leqslant P_a$, accept $\theta'(t) = \theta''(t)$, otherwise $\theta'(t) = \theta(t)$.

(7) *Component annihilation*: Any component satisfying condition (15) will die. Use CEM^2 to update parameters and perform step (7) until no component satisfies death criterion. Let $\theta(t+1)$ denote the final parameters of mixture models. Update $t = t + 1$.

(8) *End condition*: If $t < T$(T is a pre-defined large number), go to (2), otherwise output θ^* where $\theta^* = \arg\max_{\theta(t)} L(\theta(t), \mathbf{X})$, $t = 0, 1, \ldots, T$.

4. Experiments

In Section 4.1, three examples with three different synthetic data sets are presented to show how the proposed algorithm performs. In order to comprehend the necessity

of introducing the competitive mechanism, we illustrate the drawbacks of the separated split and merge criterion in Section 4.2. Performance analysis of the CEM algorithm in comparison with EM and AEM is reported in Section 4.3.

4.1. Synthetic data

4.1.1. The first example

In the first example, we use 2000 samples from an 8-component two-dimensional GMM shown in Fig. 1a. The parameters are:

$$\alpha_i = 1/8, \quad i = 1,2,\ldots,8,$$

$$\mu_1 = [1.5,0]^{\mathrm{T}},\ \mu_2 = [1,1]^{\mathrm{T}},\ \mu_3 = [0,1.5]^{\mathrm{T}},$$

$$\mu_4 = [-1,1]^{\mathrm{T}},\ \mu_5 = [-1.5,0]^{\mathrm{T}},\ \mu_6 = [-1,-1]^{\mathrm{T}},$$

$$\mu_7 = [0,-1.5]^{\mathrm{T}},\ \mu_8 = [1,-1]^{\mathrm{T}}.$$

$$\Sigma_1 = \Sigma_5 = \mathrm{diag}[0.01,0.1],$$

$$\Sigma_3 = \Sigma_7 = \mathrm{diag}[0.1,0.01],$$

$$\Sigma_2 = \Sigma_4 = \Sigma_6 = \Sigma_8 = \mathrm{diag}[0.1,0.1]. \tag{16}$$

In this experiment, we set three typical local maxima, as depicted in Fig. 1, as three initial states of GMM and run CEM from these states to demonstrate how it works. Fig. 3 shows the optimization procedure. Fig. 3(a–k) show what happens when CEM runs from the states depicted in Fig. 1b, c and d, respectively. When CEM runs from Fig. 1b, the kernels in the overpopulated region (kernels 1 and 2) merge in step 1 and kernel 3 in the sparse region is selected to split in step 4. In this way, the CEM quickly finds the global optimal solution shown in Fig. 3c. When CEM runs from Fig. 1c, kernel 2 which has converged to the boundary of the parameter space is chosen to die in step 1 because there are too few samples in this component. The inappropriate kernel 7 splits in step 5 and thus CEM achieves the global optimal solution quickly. When CEM runs from Fig. 1d where two kinds of local maxima coexist, it selects one annihilation operation, one merge operation and two split operations efficiently and quickly converges to the global optimal solution successfully.

The significance of the split and merge operations can be easily comprehended by these figures. The separated local maxima in the parameter space of the EM algorithm are linked by the split or merge operation, and the best solution can be achieved with a very high probability.

4.1.2. The second example

In the second example, we use 1000 samples from 4-component GMM shown in Fig. 4 (also see [7,11]). In this GMM, the two components (kernels 1 and 2) share a common mean, but have different covariance matrices. The prior probability of kernel 4 is a little lower than the other kernels. The parameters of GMM are given as follows:

$$\alpha_1 = \alpha_2 = \alpha_3 = 0.3, \quad \alpha_4 = 0.1,$$

$$\mu_1 = \mu_2 = [-4,-4]^{\mathrm{T}},$$

$$\mu_3 = [2,2]^{\mathrm{T}},\ \mu_4 = [-1,-6]^{\mathrm{T}},$$

$$\Sigma_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 6 & -2 \\ -2 & 6 \end{bmatrix},$$

$$\Sigma_3 = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \quad \Sigma_4 = \begin{bmatrix} 0.125 & 0 \\ 0 & 0.125 \end{bmatrix}. \tag{17}$$

Fig. 4 illustrates what happens during the optimization procedure when we run CEM from a challenging initial configuration. Here, we use the *random initialization* method mentioned in Section 3.2.2. In Fig. 4, many details have been plotted to show how CEM finds the global optimal solution by alternately implementing the following: using EM iteration to find the local maxima, selecting the split or merge operation by competitive mechanism and eliminating meaningless components with the annihilation operation.

4.1.3. The third example

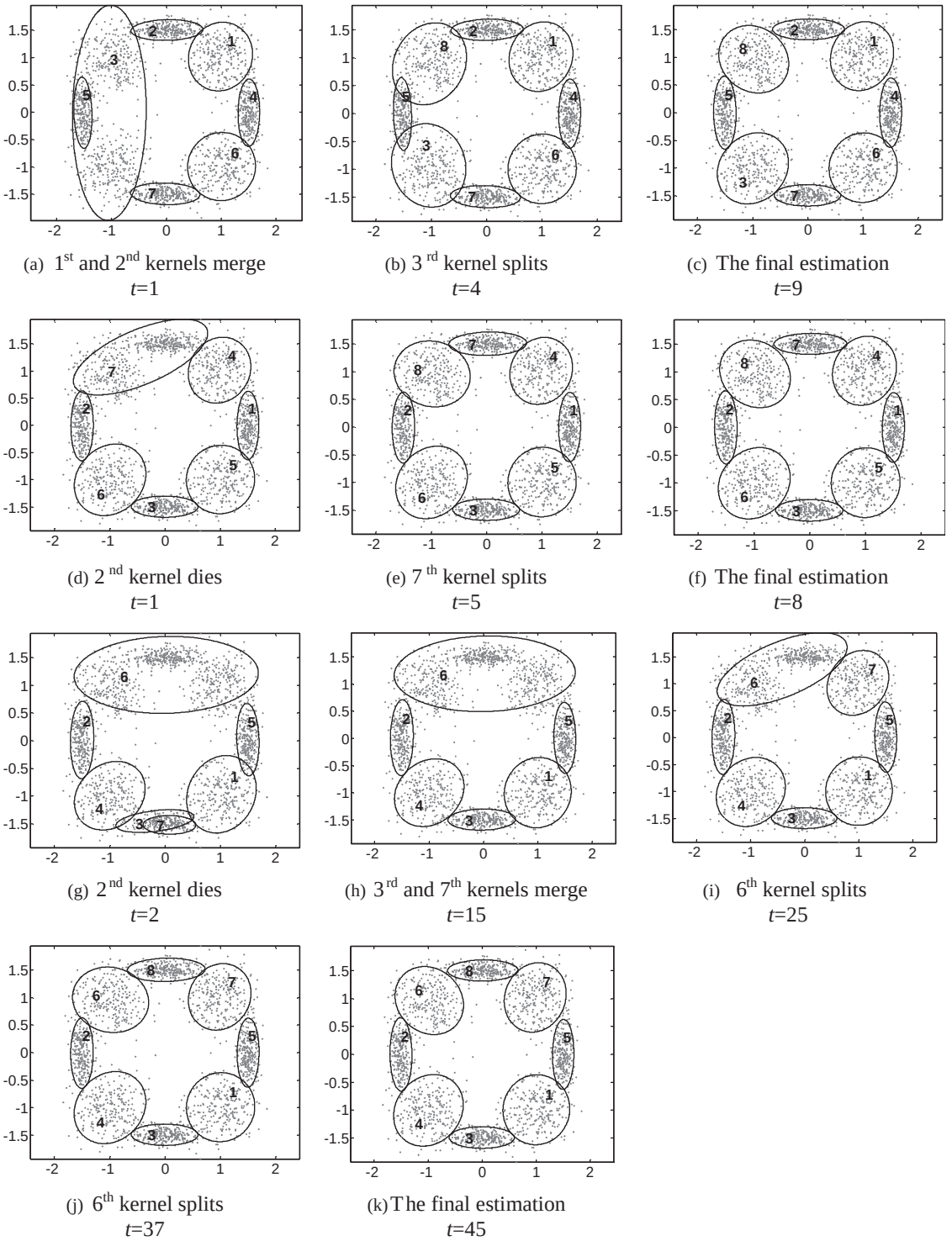
In the third example, we fit GMM to one complicated spiral manifold data set with 2000 samples shown in Fig. 5a. The data is generated by

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} (2+10t)\sin(7\pi t) \\ (2+10t)\cos(7\pi t) \end{bmatrix} + 0.5 * \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}, \tag{18}$$

where t is uniformly distributed in $[0,1]$, and n_1, n_2 are i.i.d. $N(0,1)$.

Fig. 5 shows the details of one experiment with this data set. Fig. 5b plots one random initial state ($k=30, T=1000$), Figs. 5c–f show some intermediate states of the searching procedure. It can be observed that CEM uses the CEM^2 method to eliminate 3 kernels when $t=1$; most components have moved to appropriate positions when $t=10$; CEM slightly simplifies the model after 100 steps; the acceptable solution has been achieved when $t=200$; the final parameter estimation of GMM by CEM is given in Fig. 5g (29kernels). Compared with Fig. 5h, which plots the components' principal axes of the optimal GMM, our algorithm has successfully found one appropriate solution.

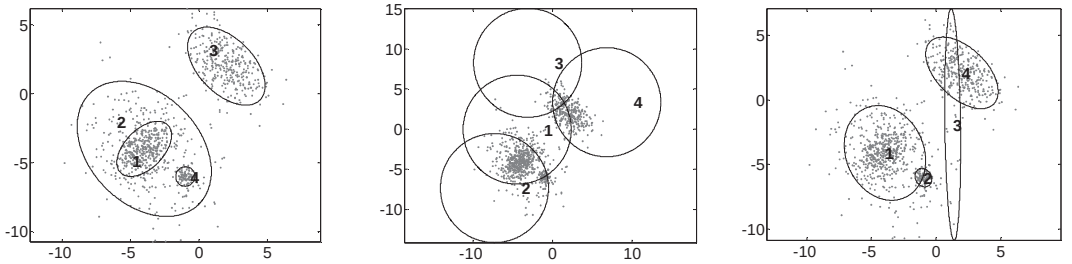
The proposed algorithm can usually converge to the state shown in Fig. 5g and another state (28 kernels) with a low probability, regardless of the kernel number in the initial state (e.g. $k=1$, 30, and 100) and which kind of initial methods are used (e.g. random initialization or k-mean initialization). It achieves very good solutions and never gets trapped in poor local maxima. In CEM, the necessary



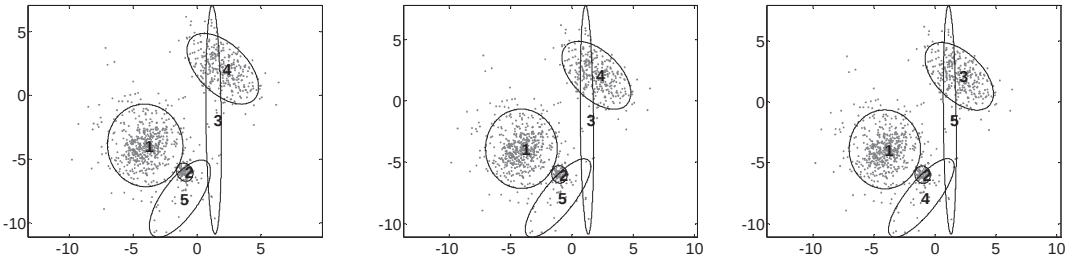
(a) 1st and 2nd kernels merge t=1
(b) 3 rd kernel splits t=4
(c) The final estimation t=9
(d) 2 nd kernel dies t=1
(e) 7 th kernel splits t=5
(f) The final estimation t=8
(g) 2 nd kernel dies t=2
(h) 3rd and 7th kernels merge t=15
(i) 6th kernel splits t=25
(j) 6th kernel splits t=37
(k) The final estimation t=45

Fig. 3. Example 1: CEM runs from local maxima:(a–c) from Fig. 1b, (d–f) from Fig. 1c, (g–k) from Fig. 1d.

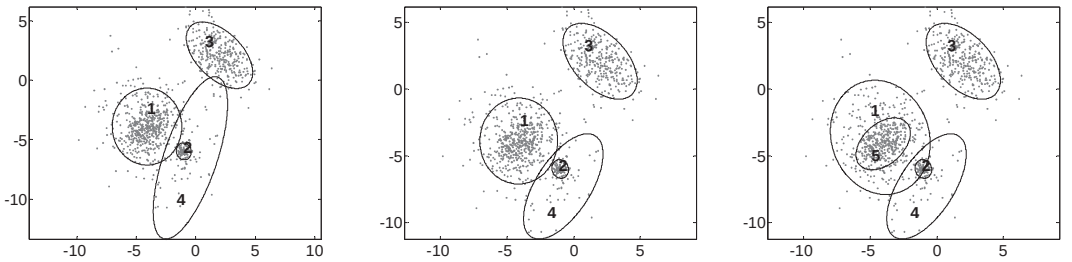
number of iterations to find the satisfactory solutions is correlated with the initial component number, e.g. when $k=100$, the requisite iterations are twice as many as in the case when $k=30$. If we use k-mean as the initialization method, we can avoid some ineffective split and merge operations and thus decrease the iteration number required (We should



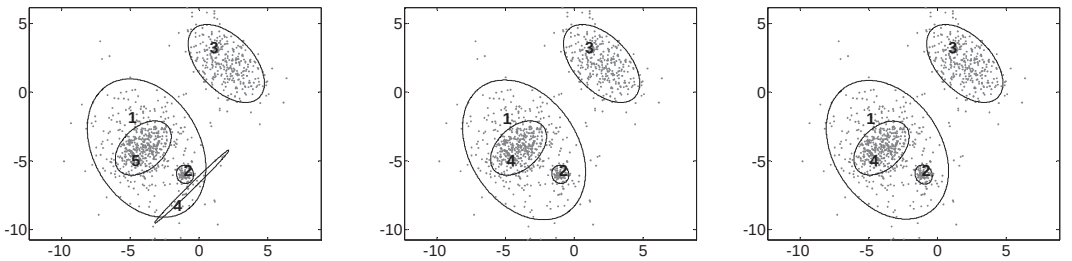
(a) The true mixtures (1000 samples) (b) Initialization (T=200) t=0 (c) The first maximum t=30



(d) 1 st kernel splits t=36 (e) The second maximum t=46 (f) 3 rd kernel splits to 2 kernels and one of them dies t=52



(g) 4th and 5th kernels merge t=60 (h) The third maximum t=86 (i) 1 st kernel splits t=89



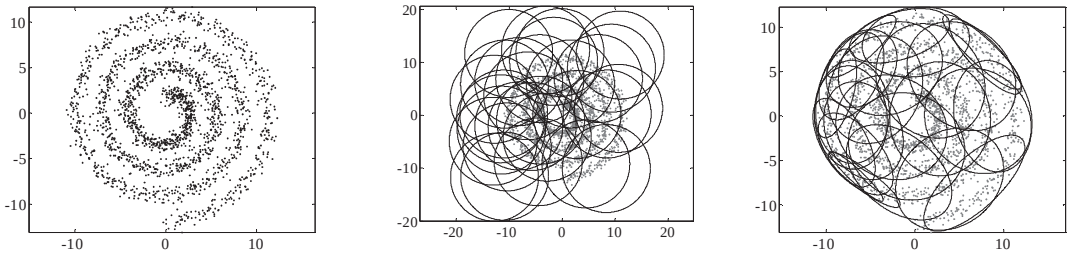
(j) The fourth maximum t=101 (k) 1 st and 4th kernels merge t=110 (l) The final estimation t=115

Fig. 4. Example 2: a convergence process of CEM.

emphasize that k-mean initialization cannot always decrease the necessary number of iterations, some statistics addressing this issue will be given in Section 4.3). In Fig. 5i-l, the approximate necessary number of iterations in each run is given below the graph (e.g. Fig. 5i, $T = 800$).

4.2. Drawbacks of the separated split and merge criteria

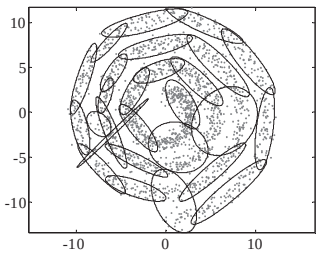
We will focus on drawbacks of the separated split and merge criteria in this section. In several sequential iterations, the same components will be chosen many times to



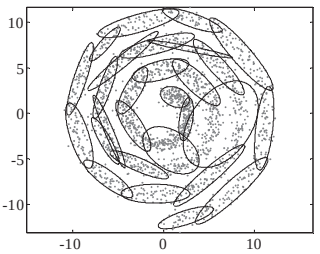
(a) 2000 samples

(b) Random initialization with 30 kernels (T=1000)

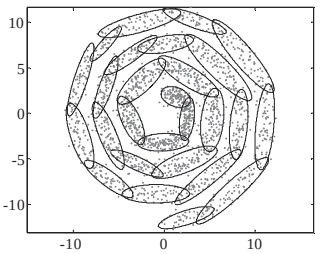
(c) 27 kernels t=1



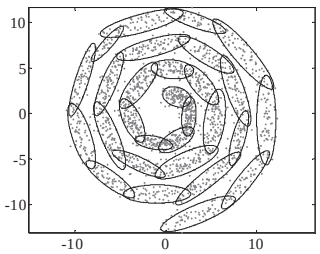
(d) 2 7 kernels t=10



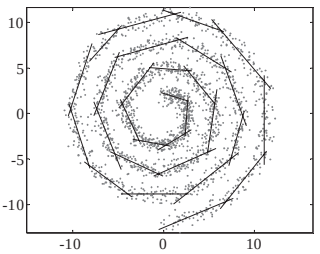
(e) 28 kernels t=100



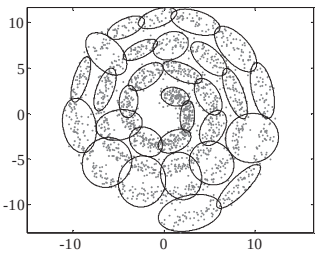
(f)27 kernels t=200



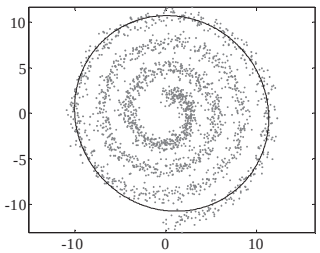
(g) The final estimation to 29 kernels



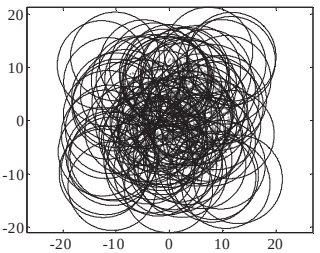
(h) The principal axes of Gaussian Mixtures



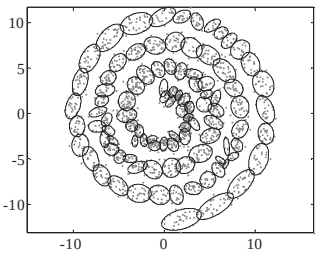
(i) Initialization with 30 kernels using k-means algorithm (T=800)



(j) Initialization with 1 kernel (T=1500)



(k) Random initialization with100 kernels (T=2500)



(l) Initialization with 100 kernels using k-means algorithm (T=2000)

Fig. 5. Example 3.

do merge and split operations, i.e. they merge and split, remerge and re-split. For example, in Fig. 6, if the current state is Fig. 6a, only one split operation is needed to reach the global optimal solution shown in Fig. 6d. But if the split and merge criteria are separated and the operation is sampled with equal probability, the algorithm may unfortunately select merge instead of split at this point and lead the model to the state shown in Fig. 6b. From this state to Fig. 6d, at least two split operations are required (e.g. first split to Fig. 6c then to Fig. 6d). For complicated situations, this

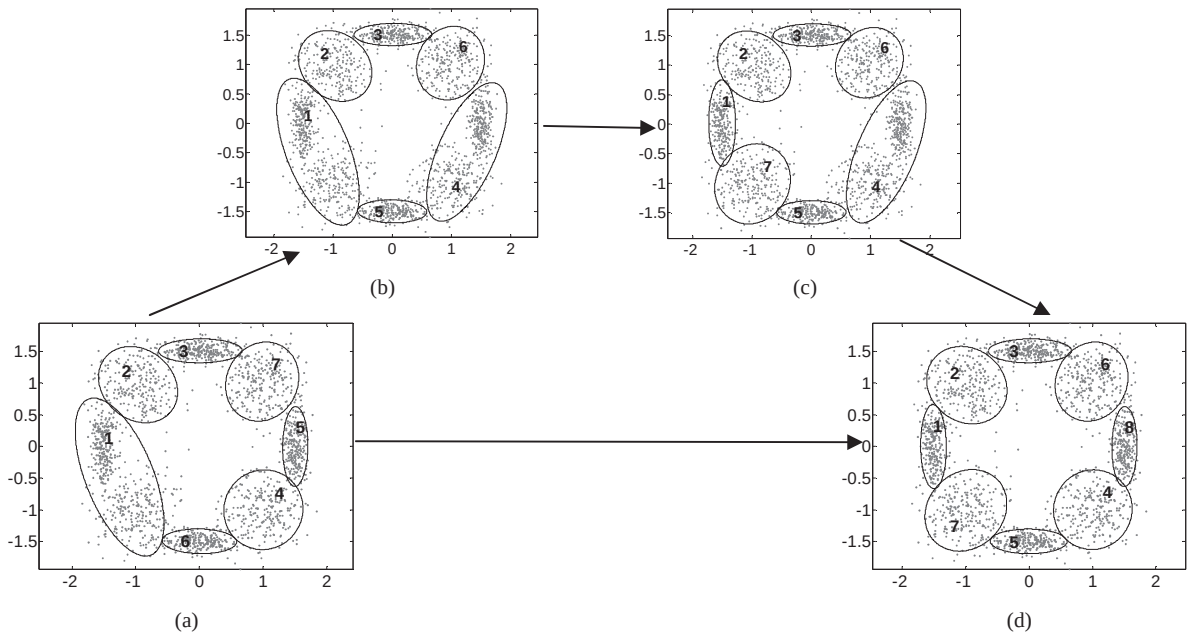


Fig. 6. EM variants with separated split and merge criteria may run in long path: *a–b–c–d*, CEM will go directly from a to d with high probability.

kind of ineffective operation will happen in different regions simultaneously causing a long convergence time. As for CEM, the competitive mechanism is introduced and it can choose the operation more efficiently according to the split or merge criterion. If one component needs to be split in some local region, the CEM algorithm will split it with high probability and vice versa. In this way, it avoids the long path (*a–b–c–d*) and chooses the short-cut (*a–d*) with high probability.

4.3. Performance analysis

In this section, we will investigate the capability of our algorithm with the data sets presented in examples 1 and 2. We performed a great number of experiments with two different initialization methods (the *random initialization* method and *k*-mean initialization method) and different initial component numbers. Compared with EM and AEM, these experiments show attractive performance for CEM.

Tables 1 and 2 report some statistics on the performance of EM, AEM and CEM with the test data sets. Each row represents the statistics for 1000 runs in one turn. Two indices concerning the performance of the algorithms are given, including P_s—the success ratio to find the global optimal solution in one turn and T_{avg}—the average count of iterations to find the global optimal solution in one turn (when the algorithm fails in one run, all the iterations in this run are counted, e.g. for CEM, T iterations will be counted). The second index reveals the average requisite computation load. The initial k is set to be: the optimal number, the lowest number (1) and one large number (20). When initial $k = 1$, random method and k-mean have no difference. Some observations on these tables are given below:

- By means of the random initialization method, AEM and CEM will search out the global optimal solution with a much higher probability than the basic EM. In the simple case when all the components have similar prior probability, CEM and AEM can always find the global optimal solution and EM can also obtain the global optimal solution with high probability (in Table 1, $P_s = 46.7\%$ for EM). When there is a great disparity between the prior probabilities of the components, CEM performs much better than AEM and EM rarely finds the global optimal solution, as in Table 2, $P_s = 19.0\%, 71.0\%, 90.2\%$ for EM, AEM and CEM ($T = 200, k = 4$) respectively. Concerning the average computation burden, EM is the lowest, and CEM is a little more when the k is initially set to be the optimal number or the lowest number (1). When the initial k is too high, both AEM and CEM will result in a high work load, as in Table 1, $T_{avg} = 24.7, 250.1, 33.6, 50.9, 209.1$ for EM, AEM, CEM($T = 200, k = 4$), CEM($T = 200, k = 1$) and CEM($T = 400, k = 20$), respectively.
- By means of the k-mean initialization method, EMs efficiency is greatly enhanced compared with the random initialization method. The computation burden to find the global optimal solution generally decreases (in Table 2,

Table 1
Statistics of experiments for Example 1

Algorithm	Initialization			% Successful (P_s)	Average Count of Iterations (T_{avg})
	T	k	Method		
EM	–	8	Random	46.7	24.7
			k-means	73.3	13.4
AEM	–	20	Random	100.0	250.1
			k-means	100.0	313.5
CEM	200	8	Random	99.8	33.6
			k-means	100.0	12.8
		1	–	99.6	50.9
	400	8	Random	100.0	32.5
			k-means	100.0	11.7
		1	–	100.0	52.4
		20	Random	98.7	209.6
			k-means	95.8	250.5
	800	20	Random	100.0	211.4
			k-means	100.0	249.5

Table 2
Statistics of experiments for Example 2

Algorithm	Initialization			% Successful (P_s)	Average count of iterations (T_{avg})
	T	k	Method		
EM	–	4	Random	19.0	23.9
			k-means	59.2	29.2
AEM	–	20	Random	71.0	242.1
			k-means	60.3	435.6
CEM	200	4	Random	90.4	68.9
			k-means	93.7	43.7
		1	–	92.2	59.9
	400	4	Random	95.6	79.2
			k-means	96.3	54.5
		1	–	96.9	70.6
		20	Random	72.2	280.4
			k-means	37.0	368.3
	800	20	Random	92.0	342.7
			k-means	83.0	506.4

the statistics of the EMs T_{avg} seems to be exceptional only because EM will easily get trapped in local maxima in a few steps with random initial method). Furthermore, if the initial k is similar or equal to the optimal component number, k-mean enhances CEM as well as EM. When the initial k is much more than the optimal number, k-mean only has negative effects on AEM and CEM: decreasing the probability of finding the optimal solution and increasing the computation load. With respect to CEM, it increases the required number of iterations, thus when T is fixed to a number that is not big enough, it is difficult for CEM to find the global optimal solution (it can be observed in Table 2, where $T = 400$, $k = 20$).

In general, concerning the capability of finding the global optimal solution, CEM and AEM perform better than EM with CEM being more stable in complicated situations. The computation burden of AEM and CEM increases greatly when the initial kernel number is much larger than the optimal one. CEM has outstanding performance using various initial methods. To get better results, CEM requires a higher computation burden than AEM (when the initial k is set to

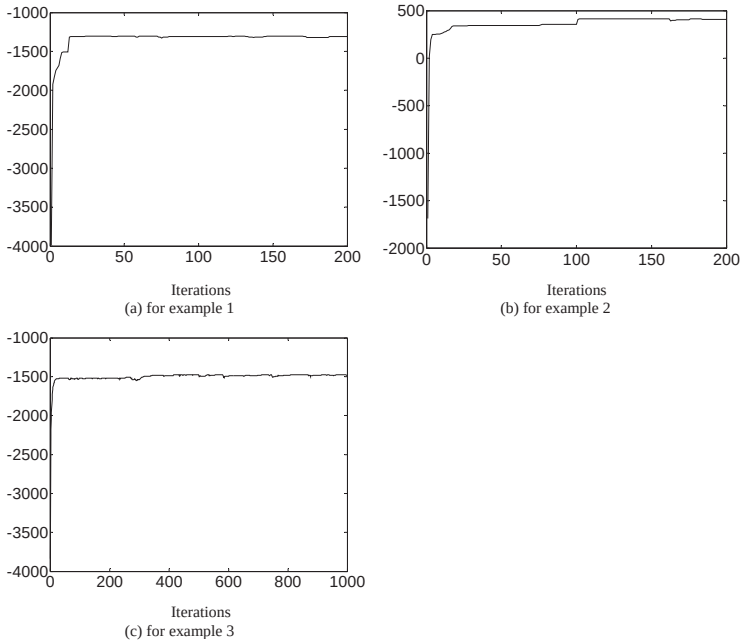


Fig. 7. Evolution of criterion function $L(\theta(t), \mathbf{X})$ for 3 examples.

be too large, CEM requires 1–3 times as much calculation as AEM).

The typical convergence curves of CEM are presented in Fig. 7. When CEM encounters local maxima, it attempts to find a better solution by efficiently selecting the split or merge operation to change the GMM structure and quickly obtains the global optimal solution.

5. Application for structure analysis of Chinese characters

We applied the CEM algorithm to the structure analysis of Chinese characters possessing very complicated and greatly varied structures. The samples came from the characters' images and the CEM algorithm was applied to extract the strokes of these characters.

Fig. 8 shows the stroke extraction results for two styles of complicated Chinese characters. The fist two rows are print style characters "龍 (dragon)" and "兼 (both)", the last two rows are handwritten style characters "贏 (win)" and "谢 (thank)". The original images of these characters are shown in Fig. 8a. The smallest rectangle boundary of the object character is $w \times h$(pixels). It is extracted and scaled to be $100w/\max(w,h) \times 100h/\max(w,h)$ (pixels). The coordinates of all the foreground pixels in these scaled fields are recorded and normalized to $[-1, 1]$. The final samples are shown in Fig. 8b. Different characters will have different numbers of sample points depending on complexity (about 500–2500 points). Fig. 8c shows the final parameter estimations of GMMs. Fig. 8d shows the stroke-extraction results directly from the GMMs, which are better than expected. Even in very complicated cases, CEM achieves promising performance, e.g. the radical "月 (moon)" located in the bottom left of the character "贏 (win)" has great overlap of two strokes, but the algorithm is able to extract the strokes. The structure analysis of Chinese characters can start from these strokes, which we hope to study in future.

All these experiments prove that the CEM algorithm we propose is very effective for the parameter estimation of complex problems.

6. Conclusion

A novel CEM algorithm for finite mixture models is presented in this paper. It includes stages of EM iteration, split, merge and annihilation operations. The initial component number and model parameters can be set arbitrarily and the split and merge operation can be selected efficiently by a competitive mechanism we have proposed. Using the annihilation operation, we have overcome the problem of converging to the boundaries of parameter space. CEM can easily escape all kinds of local extrema and automatically determine the appropriate component number.

The experimental results of employing GMM for complicated synthetic data and the structure analysis of intricate Chinese characters show that our algorithm can find excellent solutions with higher probability than other methods, especially in complicated situations. Although the computation burden of our algorithm is a little heavier than Figueiredo's, it is still acceptable given the better overall performance.

Later work will involve the design of better end criterion and some other simple measurements of the distance

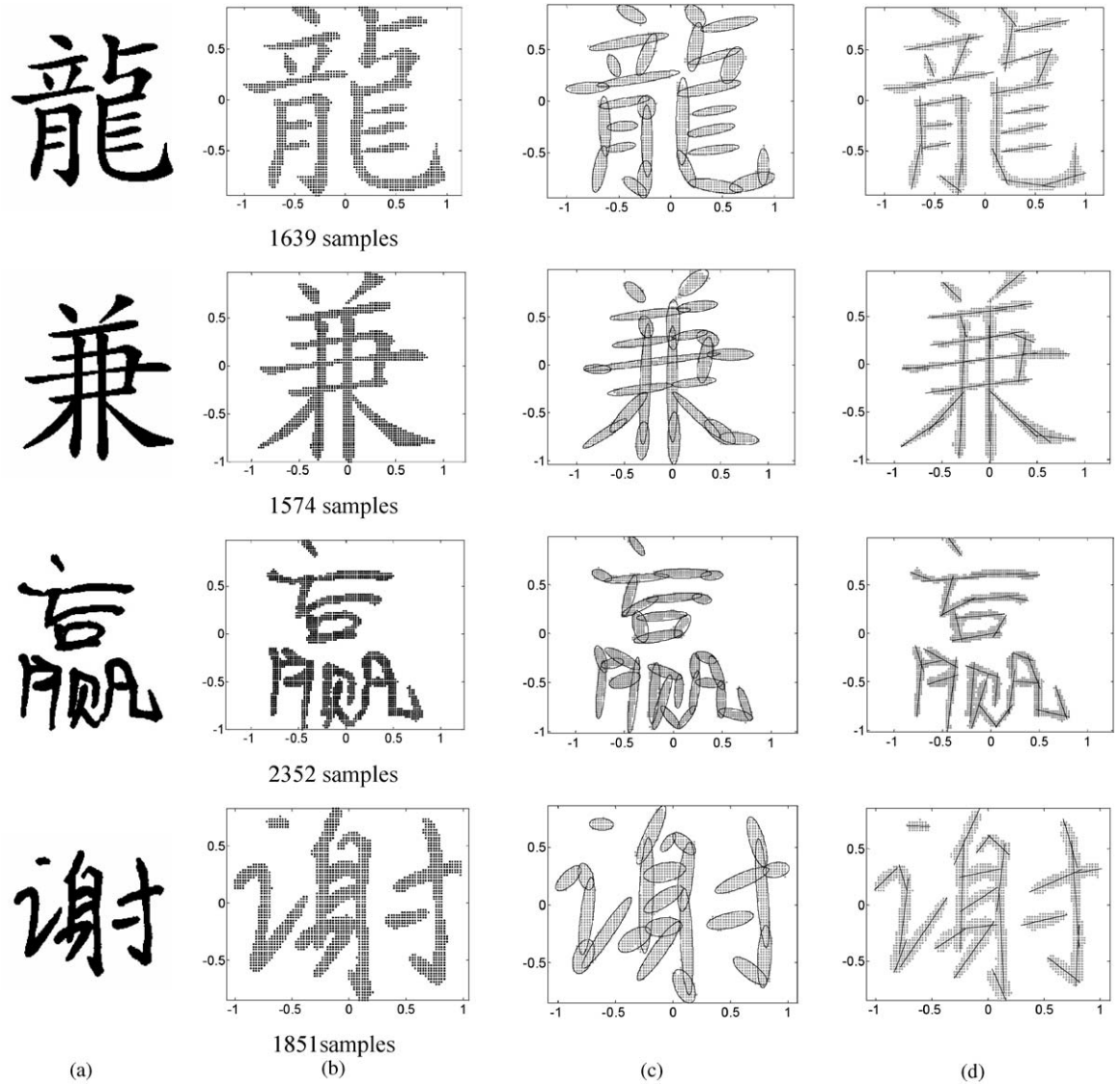


Fig. 8. Stroke-Extraction results from the complicated Chinese characters "龍 (dragon)" and "兼 (both)" and handwritten Chinese characters "赢 (win)" and "谢 (thank)". (a) The characters' images; (b) Samples of the images, (c) The final estimation of GMM, (d) Stroke-Extraction results directly from the GMMs.

between two probability distributions. More theoretical research is needed to better choose the parameters β and γ, which are now determined by experiment.

References

[1] G. McLachlan, D. Peel, Finite Mixture Models, Wiley, New York, 2000.

[2] Anil K. Jain, R. Dubes, Algorithm for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[3] G. Hinton, P. Dayan, M. Revow, Modeling the manifolds of images of handwritten digits, IEEE Trans. Neural Networks 8 (1997) 65–74.

[4] J.A. Bilmes, A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models, ICSI Technical report TR-97-021, http://www.icsi.berkeley.edu/∼bilmes/papers/em.ps.gz, 1997.

[5] A.P. Dempster, N.M. Laird, D.B. Rubin, maximum-likelihood from incomplete data via the EM algorithm, J. R. Statist. Soc. B 39 (1977) 1–38.

[6] N. Ueda, R. Nakano, Z. Gharhamani, G. Hinton, SMEM algorithm for mixture models, Neural Comput. 12 (2000) 2109–2128.

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

[8] S. Shoham, Robust clustering by deterministic agglomeration EM of mixtures of multivariate *t*-distributions, Pattern Recognition 35 (2002) 1127–1142.
[9] P. Green, Reversible jump Markov Chain Monte Carlo computation and Bayesian model determination, Biometrika 82 (1995) 711–732.
[10] S. Roberts, C. Holmes, D. Denison, Minimum-entropy data partitioning using reversible jump Markov Chain Monte Carlo, IEEE Trans. Pattern Analy. Matching Intelli. 23 (8) (2001) 909–914.
[11] M.A.T. Figueiredo, A.K. Jain, Unsupervised learning of finite mixture models, IEEE Trans. Pattern Anal. Matching Intelli. 24 (3) (2002) 381–396.
[12] S. Roberts, R. Everson, Independent Component Analysis: Principles and Practice, Cambridge University Press, Cambridge, 2001.
[13] G. Celeux, S. Chretien, F. Forbes, A. Mkhadri, A component-wise EM algorithm for mixtures, Technical report 3746, INRIA Rhone-Alpes, Frances, 1999, http://www.inria.fr/RRRT/RR-3746.html.

About the Author—BAIBO ZHANG received the B.E. degree in Automation Technology from Tsinghua University, Beijing, China in 1999. He is a Ph.D. candidate in the Department of Automation, Tsinghua University. His current research is mainly focused on unsupervised learning, pattern recognition and handwritten Chinese characters analysis.

About the Author—CHANGSHUI ZHANG was born in 1965. He received the B.S. degree in Mathematics from Peking University in 1986, and Ph.D. degree from Department of Automation, Tsinghua University in 1992. He is currently a professor in the Department of Automation, Tsinghua University. His interests include pattern recognition, artificial intelligence, image processing and evolutionary computation etc.

About the Author—XING YI received his B.E. degree in Automation Technology from Tsinghua University, Beijing, China in 2001. He is now a graduate student in the Department of Automation, Tsinghua University. His current research is mainly focused on machine learning, pattern recognition and image processing.