Hindawi Publishing Corporation
Computational and Mathematical Methods in Medicine
Volume 2012, Article ID 425730, 11 pages
doi:10.1155/2012/425730

Research Article

A Batch Rival Penalized Expectation-Maximization Algorithm for Gaussian Mixture Clustering with Automatic Model Selection

Jiechang Wen,[1] Dan Zhang,[2] Yiu-ming Cheung,[2] Hailin Liu,[1] and Xinge You[3]

[1] *Faculty of Applied Mathematics, Guangdong University of Technology, Guangzhou 510520, China*
[2] *Department of Computer Science, Hong Kong Baptist University, Kowloon, Hong Kong*
[3] *Department of Electronics and Information Engineering, Huazhong University of Science &Technology, Wuhan, China*

Correspondence should be addressed to Yiu-ming Cheung, ymc@comp.hkbu.edu.hk

Received 31 August 2011; Accepted 9 October 2011

Academic Editor: Sheng-yong Chen

Within the learning framework of maximum weighted likelihood (MWL) proposed by Cheung, 2004 and 2005, this paper will develop a batch Rival Penalized Expectation-Maximization (RPEM) algorithm for density mixture clustering provided that all observations are available before the learning process. Compared to the adaptive RPEM algorithm in Cheung, 2004 and 2005, this batch RPEM need not assign the learning rate analogous to the Expectation-Maximization (EM) algorithm (Dempster et al., 1977), but still preserves the capability of automatic model selection. Further, the convergence speed of this batch RPEM is faster than the EM and the adaptive RPEM in general. The experiments show the superior performance of the proposed algorithm on the synthetic data and color image segmentation.

1. Introduction

As a typical statistical technique, clustering analysis has been widely applied to a variety of scientific areas such as data mining [1], vector quantization [2, 3], image processing [4–7], and so forth. In particular, clustering analysis provides a useful tool to solve the several computer vision problems, for example, multithresholding of gray level images, analysis of the Hough space, and range image segmentation, formulated in the feature space paradigm [8]. In general, one kind of clustering analysis can be formulated as a density mixture modeling, in which each mixture component represents the density distribution of a data cluster. Subsequently, the task of clustering analysis is to identify the dense regions of the input (also called *observation* interchangeably) densities in a mixture. Such a clustering is therefore called a density mixture clustering.

In general, the Expectation-Maximum (EM) algorithm [9, 10] has provided a general solution for the parameter estimation of a density mixture model. Nevertheless, it needs to preassign an appropriate number of density components, that is, the number of clusters. Roughly, the mixture may overfit the data if too many components are utilized, whereas a mixture with too few components may not be flexible enough to approximate the true underlying model. Subsequently, the EM almost always leads to a poor estimate result if the number of components is misspecified. Unfortunately, from the practical viewpoint, it is hard or even impossible to know the exact cluster number in advance. In the literature, one promising way is to develop a clustering algorithm that is able to perform a correct clustering without preassigning the exact number of clusters. Such algorithms include the RPCL algorithm [11] and its improved version, namely, RPCCL [12]. More recently, Cheung [13, 14] has proposed a general learning framework, namely, Maximum Weighted Likelihood (MWL), through which an adaptive Rival Penalized EM (RPEM) algorithm has been proposed for density mixture clustering. The RPEM learns the density parameters by making mixture component compete each other at each time step. Not only are the associated parameters of the winning density component updated to adapt to an input, but also all rivals' parameters are penalized with the strength proportional to the corresponding posterior density probabilities. Therefore, this intrinsic rival penalization mechanism enables the RPEM to automatically select an appropriate number of densities by gradually fading out the

redundant densities from a density mixture. Furthermore, a simplified version of RPEM has included RPCL and RPCCL as its special cases with some new extensions.

In the papers [13, 14], the RPEM algorithm learns the parameters via a stochastic gradient ascending method; that is, we update the parameters immediately and adaptively once the current observation is available. In general, the adaptiveness of the RPEM makes it more applicable to the environment changed over time. Nevertheless, the convergence speed of the RPEM relies on the value of learning rate. Often, by a rule of thumb, we arbitrarily set the learning rate at a small positive constant. If the value of learning rate is assigned too small, the algorithm will converge at a very slow speed. On the contrary, if it is too large, the algorithm may even oscillate. In general, it is a nontrivial task to assign an appropriate value to the learning rate, although we can pay extra efforts to make the learning rate dynamically change over time, for example, see [15].

In this paper, we further study the MWL learning framework and develop a batch RPEM algorithm accordingly provided that all observations are available before the learning process. Compared to the adaptive RPEM, this batch one need not assign the learning rate analogous to the EM, but still preserves the capability of automatic model selection. Further, the convergence speed of this batch RPEM is faster than the EM and the adaptive RPEM in general. The experiments have shown the superior performance of the proposed algorithm on the synthetic data and color image segmentation.

The remainder of this paper is organized as follows. Section 2 reviews the MWL learning framework. In Section 3, we present the batch RPEM algorithm in detail, in which the weights involve a coefficient ε. We will therefore further explore the assignment of ε in Section 4. Section 5 shows the detailed experiment results. Finally, we draw a conclusion in Section 6.

2. Overview of Maximum Weighted Likelihood (MWL) Learning Framework

Suppose that an input $\mathbf{x} \in \Re^d$ comes from the following density mixture model:

$$P(\mathbf{x} \mid \Theta) = \sum_{j=1}^{k} \alpha_j p\left(\mathbf{x} \mid \boldsymbol{\theta}_j\right), \quad \sum_{j=1}^{k} \alpha_j = 1, \quad \alpha_j > 0, \quad \forall 1 \le j \le k, \tag{1}$$

where Θ is the parameter set of $\{\alpha_j, \boldsymbol{\theta}_j\}_{j=1}^{k}$. Furthermore, k is the number of components, α_j is the mixture proportion of the jth component, and $p(\mathbf{x} \mid \boldsymbol{\theta}_j)$ is a multivariate probability density function (pdf) of $\mathbf{x}$ parameterized by $\boldsymbol{\theta}_j$. As long as we know the value of Θ, an input $\mathbf{x}$ can be classified into a certain cluster via its posterior probability:

$$h(j \mid \mathbf{x}, \Theta) = \frac{\alpha_j p\left(\mathbf{x} \mid \boldsymbol{\theta}_j\right)}{P(\mathbf{x} \mid \Theta)} \tag{2}$$

using the winner-take-all rule, that is, assigning an input $\mathbf{x}$ to Cluster c if $c = \arg\max_j h(j \mid \mathbf{x}, \Theta)$ or taking its soft version which assigns $\mathbf{x}$ to Cluster j with the probability $h(j \mid \mathbf{x}, \Theta)$. Therefore, how to estimate the parameter set Θ, particularly without knowing the correct value of k in advance, is a key issue in density mixture clustering.

In the MWL learning framework [13, 14], the parameter set Θ is learned via maximizing the following Weighted Likelihood (WL) cost function:

$$l(\Theta) = \omega(\Theta; \mathbf{x}) + \nu(\Theta; \mathbf{x}) \tag{3}$$

with

$$\omega(\Theta; \mathbf{x}) = \int \sum_{j=1}^{k} g(j \mid \mathbf{x}, \Theta) \ln\left[\alpha_j p\left(\mathbf{x} \mid \boldsymbol{\theta}_j\right)\right] dF(\mathbf{x}),$$
$$\nu(\Theta; \mathbf{x}) = -\int \sum_{j=1}^{k} g(j \mid \mathbf{x}, \Theta) \ln h\left(j \mid \mathbf{x}, \Theta\right) dF(\mathbf{x}), \tag{4}$$

where $g(j \mid \mathbf{x}, \Theta)$'s are the designable weights satisfying the two conditions:

(1) $\sum_{j=1}^{k} g(j \mid \mathbf{x}, \Theta) = 1$,

(2) for all j, $g(j \mid \mathbf{x}, \Theta) = 0$ if $h(j \mid \mathbf{x}, \Theta) = 0$.

Suppose that a set of N i.i.d. observations, denoted as $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, comes from the density mixture model in (1). The empirical WL function of (3), written as $Q(\Theta; \mathbf{X})$, can be given as

$$Q(\Theta; \mathbf{X}) = \omega(\Theta; \mathbf{X}) + \nu(\Theta; \mathbf{X}) \tag{5}$$

with

$$\omega(\Theta; \mathbf{X}) = \frac{1}{N} \sum_{t=1}^{N} \sum_{j=1}^{k} g\left(j \mid \mathbf{x}_t, \Theta\right) \ln\left[\alpha_j p\left(\mathbf{x}_t \mid \boldsymbol{\theta}_j\right)\right],$$
$$\nu(\Theta; \mathbf{X}) = -\frac{1}{N} \sum_{t=1}^{N} \sum_{j=1}^{k} g(j \mid \mathbf{x}_t, \Theta) \ln h(j \mid \mathbf{x}_t, \Theta). \tag{6}$$

Moreover, the weights $g(j \mid \mathbf{x}_t, \Theta)$'s have been generally designed as [13, 14]

$$g(j \mid \mathbf{x}_t, \Theta) = (1 + \varepsilon_t) I(j \mid \mathbf{x}_t, \Theta) - \varepsilon_t h(j \mid \mathbf{x}_t, \Theta), \tag{7}$$

where ε_t is a coefficient varying with the time step t in general. Please note that $g(j \mid \mathbf{x}_t, \Theta)$'s in (7) can be negative as well as positive. For simplicity, we hereinafter set ε_t as a constant, denoted as ε. Moreover, $I(j \mid \mathbf{x}_t, \Theta)$ is an indicator function with

$$I(j \mid \mathbf{x}_t, \Theta) = \begin{cases} 1, & \text{if } j = c = \arg\max_{1 \le j \le k} h(j \mid \mathbf{x}_t, \Theta), \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Subsequently, under a specific weight design, the papers [13, 14] have presented the adaptive RPEM to learn Θ via maximizing the WL function of (5) using a stochastic gradient ascent method, which is able to fade out the redundant densities gradually from a density mixture. Consequently, it can automatically select an appropriate number of density components in density mixture clustering.

Interested readers may refer to the paper [14] for more details. We summarize the main steps of the adaptive RPEM in Algorithm 1. Although the experiments have shown the superior performance of the adaptive RPEM on automatic model selection, its convergence speed, however, relies on the value of learning rate. Under the circumstances, we will present a batch version without the learning rate in the next section.

3. Batch RPEM Algorithm

To estimate the parameter set within the MWL framework, we have to maximize the empirical WL function $Q(\boldsymbol{\Theta}; \mathbf{X})$ in (5). In general, we update the parameters via maximizing the first term of (5), that is, $\omega(\boldsymbol{\Theta}; \mathbf{X})$, by fixing the second term $\nu(\boldsymbol{\Theta}; \mathbf{X})$. Subsequently, we need to solve the following nonlinear optimization problem:

$$\widetilde{\boldsymbol{\Theta}} = \arg\max_{\boldsymbol{\Theta}} \{\omega(\boldsymbol{\Theta}; \mathbf{X})\} \tag{9}$$

subject to the constraints as shown in (1). We adopt the Lagrange method analogous to the EM by introducing a Lagrange multiplier λ into the Lagrange function. Subsequently, we have

$$\mathcal{L}(\boldsymbol{\Theta}, \lambda) = \omega(\boldsymbol{\Theta}; \mathbf{X}) + \lambda\left(\sum_{j=1}^{k} \alpha_j - 1\right). \tag{10}$$

In this paper, we concentrate on the Gaussian mixture model only, that is, each component $p(\mathbf{x} \mid \boldsymbol{\theta}_j)$ in (1) is a Gaussian density. We then have

$$p\left(j \mid \mathbf{x}_t, \boldsymbol{\theta}_j\right) = (2\pi)^{-d/2} \left|\mathbf{C}_j\right|^{-1/2} \exp\left[-\frac{1}{2}\left(\mathbf{x}_t - \boldsymbol{\mu}_j\right)^T \mathbf{C}_j^{-1}\left(\mathbf{x}_t - \boldsymbol{\mu}_j\right)\right], \tag{11}$$

where $\boldsymbol{\theta}_j = (\boldsymbol{\mu}_j, \mathbf{C}_j)$, $\boldsymbol{\mu}_j$ and $\mathbf{C}_j$ are the mean (also called *seed points* interchangeably) and the covariance of the jth density, respectively.

Through optimizing (10), we then finally obtain the batch RPEM algorithm as shown in Algorithm 2. If a covariance matrix $\mathbf{C}_j^{(n+1)}$ is singular, then it indicates that the corresponding jth density component is degenerated and can be simply discarded without being learned any more in the subsequent iterations. In this case, we have to normalize those remaining $\alpha_r^{(n+1)}$'s $(r \neq j)$ so that their sum is always kept to be 1.

In the above batch RPEM, its capability of automatic model selection is controlled by the weight functions $g(j \mid \mathbf{x}_t, \boldsymbol{\theta})$'s, which further rely on the parameter ε as shown in (7). Subsequently, a new question is arisen: how to assign an appropriate value of ε? The next section will answer this question.

4. How to Assign Parameter ε?

To deal with how to assign an appropriate value of ε, we rewrite (7) as the following form:

$$\begin{aligned} & g(j \mid \mathbf{x}_t, \boldsymbol{\Theta}) \\ &= \begin{cases} (1+\varepsilon) I(c \mid \mathbf{x}_t, \boldsymbol{\Theta}) - \varepsilon h(c \mid \mathbf{x}_t, \boldsymbol{\Theta}), & \text{if } j = c, \\ -\varepsilon h(j \mid \mathbf{x}_t), & \text{otherwise} \end{cases} \\ &= \begin{cases} h(c \mid \mathbf{x}_t, \boldsymbol{\Theta}) + (1+\varepsilon)(1 - h(c \mid \mathbf{x}_t, \boldsymbol{\Theta})), & \text{if } j = c, \\ h(j \mid \mathbf{x}_t, \boldsymbol{\Theta}) - (1+\varepsilon) h(j \mid \mathbf{x}_t, \boldsymbol{\Theta}), & \text{otherwise.} \end{cases} \end{aligned} \tag{12}$$

Intuitively, the term $(1+\varepsilon)(1 - h(c \mid \mathbf{x}_t, \boldsymbol{\Theta}))$ can be regarded as the award received by the winning density component (i.e., the cth density with $I(c \mid \mathbf{x}_t, \boldsymbol{\Theta}) = 1$), and meanwhile the term $-(1+\varepsilon)h(j \mid \mathbf{x}_t, \boldsymbol{\Theta})$ is the penalty of the rival components (i.e., those densities with $I(j \mid \mathbf{x}_t, \boldsymbol{\Theta}) = 0$). In general, it is expected that the award is positive and the penalty is negative. That is, ε should be greater than -1. Otherwise, as $\varepsilon < -1$, we will meet an awkward situation; that is, the amount of award is negative and the penalty one becomes positive. This implies that we will penalize the winner and award the rivals, which evidently violates our expectations. Furthermore, as $\varepsilon = -1$, both of the award and penalty amount become zero. In this special case, the batch RPEM is actually degenerated into the EM without the property of automatic model selection. As a result, ε is required to be greater than -1. In addition, ε in the batch RPEM should be a negative value. Otherwise, the weights of the rival components $g(j \mid \mathbf{x}_t, \boldsymbol{\Theta}) = -\varepsilon h(j \mid \mathbf{x}_t)$'s become negative, resulting in some α_j's to be negative finally. Hence, an appropriate selection of ε in the batch RPEM would be a negative value and greater than -1. That is, ε should be fallen into the range of $(-1, 0)$.

Furthermore, our empirical studies have found that a smaller ε will lead the batch RPEM algorithm to a more robust performance, especially when the value of k is large and the data are overlapped considerably. In other words, the algorithm has a poor capability of automatic model selection if ε is close to zero. To illustrate this scenario, we have utilized two synthetic data sets that are generated from the two bivariate three-Gaussian mixtures individually as shown in Figures 1(a) and 1(b), where each data set consists of 1,000 observations with the true mixture proportions: $\alpha_1^* = 0.4$, $\alpha_2^* = 0.3$, and $\alpha_3^* = 0.3$. Also, the true $\boldsymbol{\mu}_j^*$'s and $\mathbf{C}_j^*$'s of data set 1 in Figure 1(a) are

$$\begin{aligned} \boldsymbol{\mu}_1^* &= \begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix}, \quad \boldsymbol{\mu}_2^* = \begin{pmatrix} 1.0 \\ 5.0 \end{pmatrix}, \quad \boldsymbol{\mu}_3^* = \begin{pmatrix} 5.0 \\ 5.0 \end{pmatrix}, \\ \mathbf{C}_1^* &= \begin{pmatrix} 0.3 & 0.2 \\ 0.2 & 0.4 \end{pmatrix}, \quad \mathbf{C}_2^* = \begin{pmatrix} 0.2 & -0.1 \\ -0.1 & 0.3 \end{pmatrix}, \\ \mathbf{C}_3^* &= \begin{pmatrix} 0.30 & -0.20 \\ -0.20 & 0.25 \end{pmatrix}, \end{aligned} \tag{13}$$

Initialization: Given a specific k ($k \geq k^*$, k^* is the true number of clusters), initialize the parameter $\boldsymbol{\Theta}$.
Step 1: Given the current input $\mathbf{x}_t$ and the parameter estimate, written as $\boldsymbol{\Theta}^{(n)}$, compute $h(j \mid \mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$'s and $g(j \mid \mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$'s via(2) and (7), respectively.
Step 2: Given $h(j \mid \mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$'s and $g(j \mid \mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$'s, we update $\boldsymbol{\Theta}$ by
$\boldsymbol{\Theta}^{(n+1)} = \boldsymbol{\Theta}^{(n)} + \eta(\omega_t(\boldsymbol{\Theta}; \mathbf{x}_t)/\boldsymbol{\Theta})|_{\boldsymbol{\Theta}^{(n)}}$,
with
$\omega_t(\boldsymbol{\Theta}; \mathbf{x}_t) = \sum_{j=1}^{k} g(j\mathbf{x}_t, \boldsymbol{\Theta}) \ln[\alpha_j p(\mathbf{x}_t \boldsymbol{\theta}_j)]$,
where η is a small positive learning rate.
Step 3: Let $n = n + 1$, and go to *Step 1* for the next iteration until $\boldsymbol{\Theta}$ is converged.

Algorithm 1: Adaptive RPEM algorithm.

Initialization: Given a specific k ($k \geq k^*$, k^* is the true number of clusters), initialize the parameter $\boldsymbol{\Theta}$.
Step 1: Given $\boldsymbol{\Theta}^{(n)}$, we compute $h(j \mid \mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$'s and $g(j \mid \mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$'s for all $\mathbf{x}_t$'s via (2) and (7), respectively.
Step 2: Fixing $h(j \mid \mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$'s and $g(j \mid \mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$'s, we update $\boldsymbol{\Theta}$ by
$\alpha_j^{(n+1)} = \phi_j^{(n)} / \sum_{j=1}^{k} \phi_j^{(n)}$,
$\boldsymbol{\mu}_j^{(n+1)} = (1/\phi_j^{(n)}) \sum_{t=1}^{N} \mathbf{x}_t g(j \mid \mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$,
$\mathbf{C}_j^{(n+1)} = (1/\phi_j^{(n)}) \sum_{t=1}^{N} g(j \mid \mathbf{x}_t, \boldsymbol{\Theta}^{(n)})(\mathbf{x}_t - \boldsymbol{\mu}_j^{(n)})(\mathbf{x}_t - \boldsymbol{\mu}_j^{(n)})^T$,
where $\phi_j^{(n)} = \sum_{t=1}^{N} g(j \mid \mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$
Step 3: Let $n = n + 1$, and go to *Step 1* for the next iteration until $\boldsymbol{\Theta}$ is converged.

Algorithm 2: Batch RPEM algorithm.

while the true parameters of data set 2 in Figure 1(b) are

$$\boldsymbol{\mu}_1^* = \begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix}, \quad \boldsymbol{\mu}_2^* = \begin{pmatrix} 1.0 \\ 2.5 \end{pmatrix}, \quad \boldsymbol{\mu}_3^* = \begin{pmatrix} 2.5 \\ 2.5 \end{pmatrix},$$
$$\mathbf{C}_1^* = \begin{pmatrix} 0.3 & 0.1 \\ 0.1 & 0.4 \end{pmatrix}, \quad \mathbf{C}_2^* = \begin{pmatrix} 0.3 & 0.0 \\ 0.0 & 0.3 \end{pmatrix}, \quad \mathbf{C}_3^* = \begin{pmatrix} 0.30 & -0.05 \\ -0.05 & 0.25 \end{pmatrix}. \tag{14}$$

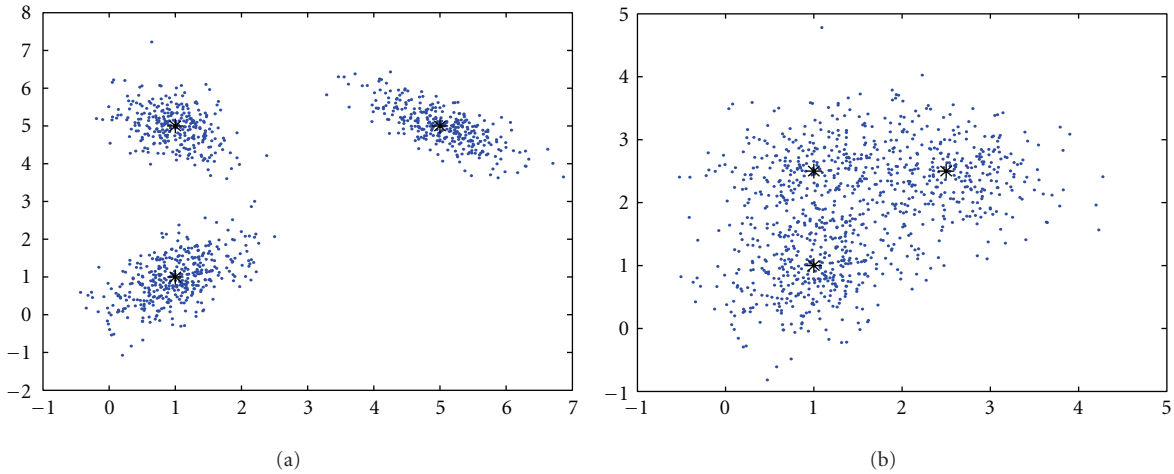
It can be seen that the clusters in data set 1 are well separated, whereas the clusters in data set 2 are overlapped considerably.

For each data set, we conducted the three experiments by setting $k = 3$, $k = 8$, and $k = 20$, respectively. Also, all α_j's and $\mathbf{C}_j$'s were initialized at $1/k$ and the identity matrix, respectively. During the learning process, we discarded those densities whose covariance matrices $\mathbf{C}_j$'s were singular. Table 1 shows the performance of the batch RPEM over the parameter ε. We found that, as $k = 3$ and $k = 8$, all ε's we have tried from -0.9 to -0.1 lead to the good performance of the algorithm when using the data set 1. For example, as $k = 8$ and $\varepsilon = -0.8$, we randomly initialized the eight seed points in the input space as shown in Figure 2(a). After all the parameters were converged, 2 out of 8 density components had been discarded and the mixture proportions of the remaining components were converged to $\alpha_1 = 0.2960$, $\alpha_2 = 0.0036$, $\alpha_3 = 0.2900$, $\alpha_4 = 0.0058$, $\alpha_5 = 0.0136$, and $\alpha_6 = 0.3910$. It can be seen that the three principal mixing proportions, α_1, α_3, and α_6, have well estimated the true ones, while the other proportions were inclined to zero. The corresponding three $\boldsymbol{\mu}_j$'s and $\mathbf{C}_j$'s were

$$\boldsymbol{\mu}_1 = \begin{pmatrix} 5.06 \\ 4.96 \end{pmatrix}, \quad \boldsymbol{\mu}_3 = \begin{pmatrix} 0.98 \\ 4.98 \end{pmatrix}, \quad \boldsymbol{\mu}_6 = \begin{pmatrix} 1.00 \\ 0.96 \end{pmatrix},$$
$$\mathbf{C}_1 = \begin{pmatrix} 0.29 & -0.17 \\ -0.17 & 0.22 \end{pmatrix}, \quad \mathbf{C}_3 = \begin{pmatrix} 0.18 & -0.08 \\ -0.08 & 0.25 \end{pmatrix}, \quad \mathbf{C}_6 = \begin{pmatrix} 0.29 & 0.19 \\ 0.19 & 0.39 \end{pmatrix}. \tag{15}$$

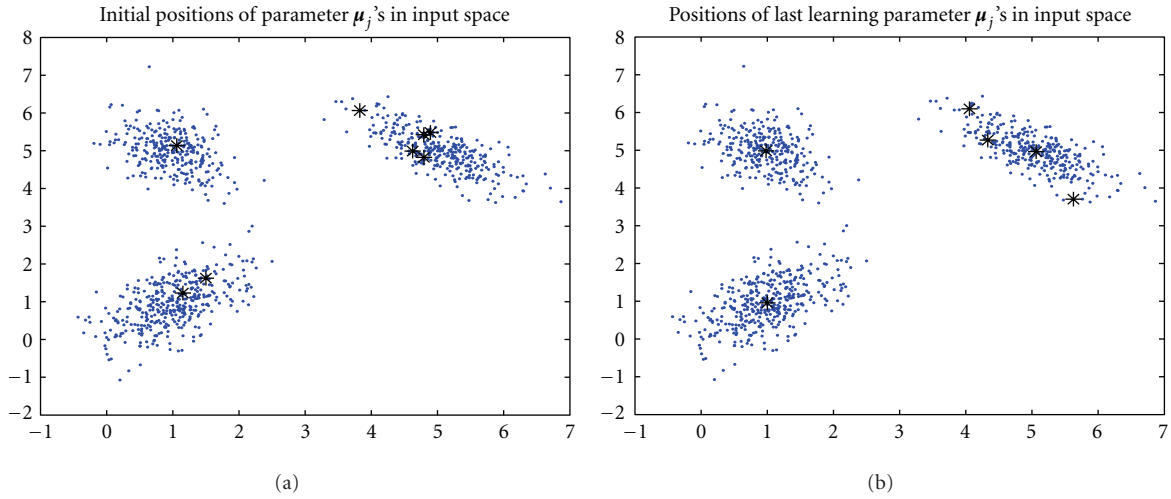
As shown in Figure 2(b), the three $\boldsymbol{\mu}_j$'s have successfully stabilized at the corresponding cluster centers, meanwhile the other three redundant seed points have been pushed away and stably located at the boundary of the clusters. That is, the redundant densities have been fade out through the learning, thus the batch RPEM can select the model automatically as well as the adaptive version.

Nevertheless, when k is set at a large value, for example, say $k = 20$, it is found that the proposed algorithm could not fade out the redundant density components from a mixture if ε is close to 0. Instead, we should set ε at a value close to -1. For example, as $k^* = 3$, $k = 20$, and $\varepsilon = -0.9$, we ran the proposed algorithm. It was found that



(a) (b)

Figure 1: (a) Synthetic data set 1 with the well-separated clusters, and (b) synthetic data set 2 with the clusters overlapped considerably.



(a) (b)

Figure 2: The performance of the batch RPEM as $k^* = 3$, $k = 8$, and $\varepsilon = -0.8$: (a) initial positions of seed points; (b) converged positions of seed points.

13 of 20 seed points were maintained by discarding those whose covariance matrices $\mathbf{C}_j$'s were singular. The mixture proportions of the remaining components were converged to $\alpha_1 = 0.0421$, $\alpha_2 = 0.0169$, $\alpha_3 = 0.0051$, $\alpha_4 = 0.2349$, $\alpha_5 = 0.0036$, $\alpha_6 = 0.0149$, $\alpha_7 = 0.3444$, $\alpha_8 = 0.0049$, $\alpha_9 = 0.0210$, $\alpha_{10} = 0.0029$, $\alpha_{11} = 0.0057$, $\alpha_{12} = 0.2944$, and $\alpha_{13} = 0.0091$. The three principal mixing proportions, α_4, α_7, and α_{12}, have also well estimated the true ones while the other proportions were tended to zero. Furthermore, the corresponding $\boldsymbol{\mu}_j$'s were $\boldsymbol{\mu}_1 = [1.0872, 4.9986]^T$, $\boldsymbol{\mu}_2 = [0.9897, 0.9640]^T$, and $\boldsymbol{\mu}_3 = [5.0754, 4.9552]^T$. As shown in Figure 3(a), the learned $\boldsymbol{\mu}_j$'s are correctly allocated at the center of the three clusters and the other redundant seed points were driven away to the boundaries of clusters. Hence, the batch algorithm performed a good model selection by assigning $\varepsilon = -0.9$. In contrast, if we assign ε to some value close to zero, the algorithm will lead to a poor model selection. We take $\varepsilon = -0.1$ for instance. The mixture proportions of the remaining 19 out of 20 components were converged to $\alpha_1 = 0.0461$, $\alpha_2 = 0.0121$, $\alpha_3 = 0.0439$, $\alpha_4 = 0.1404$, $\alpha_5 = 0.0070$, $\alpha_6 = 0.0258$, $\alpha_7 = 0.0178$, $\alpha_8 = 0.0348$, $\alpha_9 = 0.0659$, $\alpha_{10} = 0.0513$, $\alpha_{11} = 0.0493$, $\alpha_{12} = 0.0352$, $\alpha_{13} = 0.0362$, $\alpha_{14} = 0.0528$, $\alpha_{15} = 0.0587$, $\alpha_{16} = 0.0171$, $\alpha_{17} = 0.1916$, $\alpha_{18} = 0.0882$, and $\alpha_{19} = 0.0260$. It can be seen that none of α_j's tends to zero. As shown in Figure 3(b), all the converged positions have a bias from the cluster centers. In other words, the algorithm has a poor performance as ε get close to zero. Hence, if k is large, it would be better to choose a relative smaller value of ε between -1 and 0.

In addition, we also investigated the assignment of ε on data set 2, where the data are considerably overlapped. We take $k^* = 3$, $k = 20$, and $\varepsilon = -0.9$ for instance. The converged positions of the seed points are shown in Figure 4(a), where the learned positions converged to the cluster centers while driving the redundant seed points to the boundaries of the clusters. That is, the proposed batch algorithm can work quite well as $\varepsilon = -0.9$. Also, we let $k^* = 3$, $k = 20$, and $\varepsilon = -0.2$ to run the algorithm again

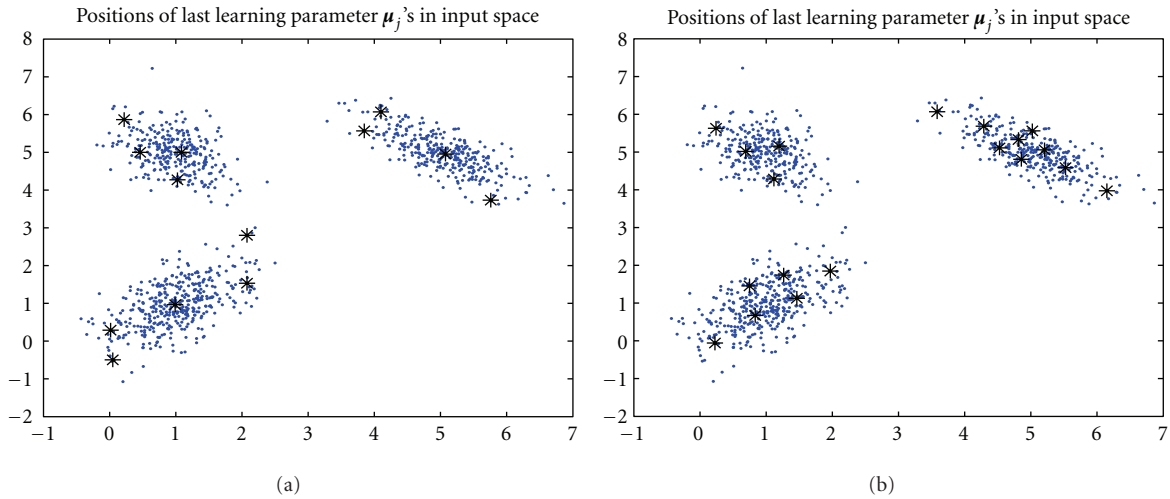


FIGURE 3: The converged positions of the seed points as $k^* = 3$ and $k = 20$: (a) $\varepsilon = -0.9$, (b) $\varepsilon = -0.1$.

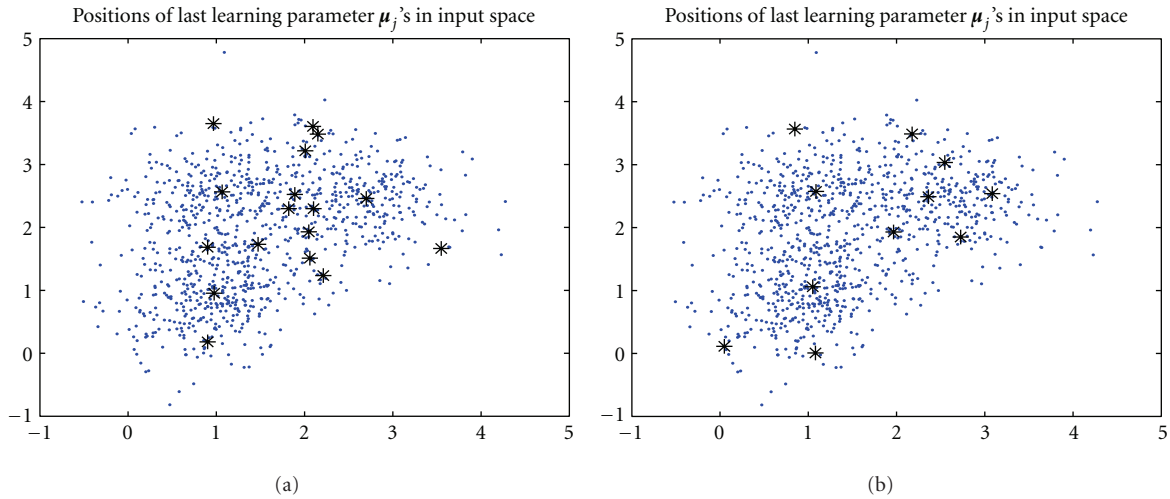


FIGURE 4: The converged positions of the seed points learned via the batch RPEM as $k^* = 3$ and $k = 20$: (a) $\varepsilon = -0.9$; (b) $\varepsilon = -0.2$.

for comparison. As shown in Figure 4(b), the converged positions of the seed points have a bias from the cluster centers. This implies that the values of ε that are close to zero cannot work well in this case. More examples can be found in Table 1. It can be seen that the feasible region of ε is shrunk over the overlap level of the data. For example, the appropriate values of ε are in the range of $[-0.9, -0.6]$ only when using the date set 2 with $k^* = 3$ and $k = 3$ or 8, respectively. In contrast, ε is feasible in the full range of $[-0.9, -0.1]$ where we have tried so far as data set 1 is used. Hence, by a rule of thumb, we should choose an appropriate value of ε close to -1. Nevertheless, we have also noted that it is not a good choice if ε is too close to -1. In fact, the proposed algorithm will gradually degenerate to the EM as ε tends to -1; that is, the capability of the proposed algorithm on model selection will be reduced as ε tends to -1. Hence, to sum up, empirical studies have found that $[-0.9, -0.8]$ is an appropriate feasible region of ε. In the next section, we therefore arbitrarily set ε at -0.8.

5. Experimental Results

To evaluate the performance of the batch RPEM algorithm, we have conducted the following three experiments.

5.1. Experiment 1: Batch RPEM on Synthetic Data with $K = K^*$. This experiment was to evaluate the convergence speed of the batch RPEM. We utilized 1,000 data points from a mixture of three bivariate Gaussian densities with the true parameters as follows:

$$\alpha_1^* = 0.3, \quad \alpha_2^* = 0.4, \quad \alpha_3^* = 0.3,$$
$$\boldsymbol{\mu}_1^* = [1.0, 1.0]^T, \quad \boldsymbol{\mu}_2^* = [1.0, 2.5]^T, \quad \boldsymbol{\mu}_3^* = [2.5, 2.5]^T,$$
$$\mathbf{C}_1^* = \begin{pmatrix} 0.20 & 0.05 \\ 0.05 & 0.30 \end{pmatrix}, \quad \mathbf{C}_2^* = \begin{pmatrix} 0.2 & 0.0 \\ 0.0 & 0.2 \end{pmatrix},$$
$$\mathbf{C}_3^* = \begin{pmatrix} 0.2 & -0.1 \\ -0.1 & 0.2 \end{pmatrix}. \tag{16}$$

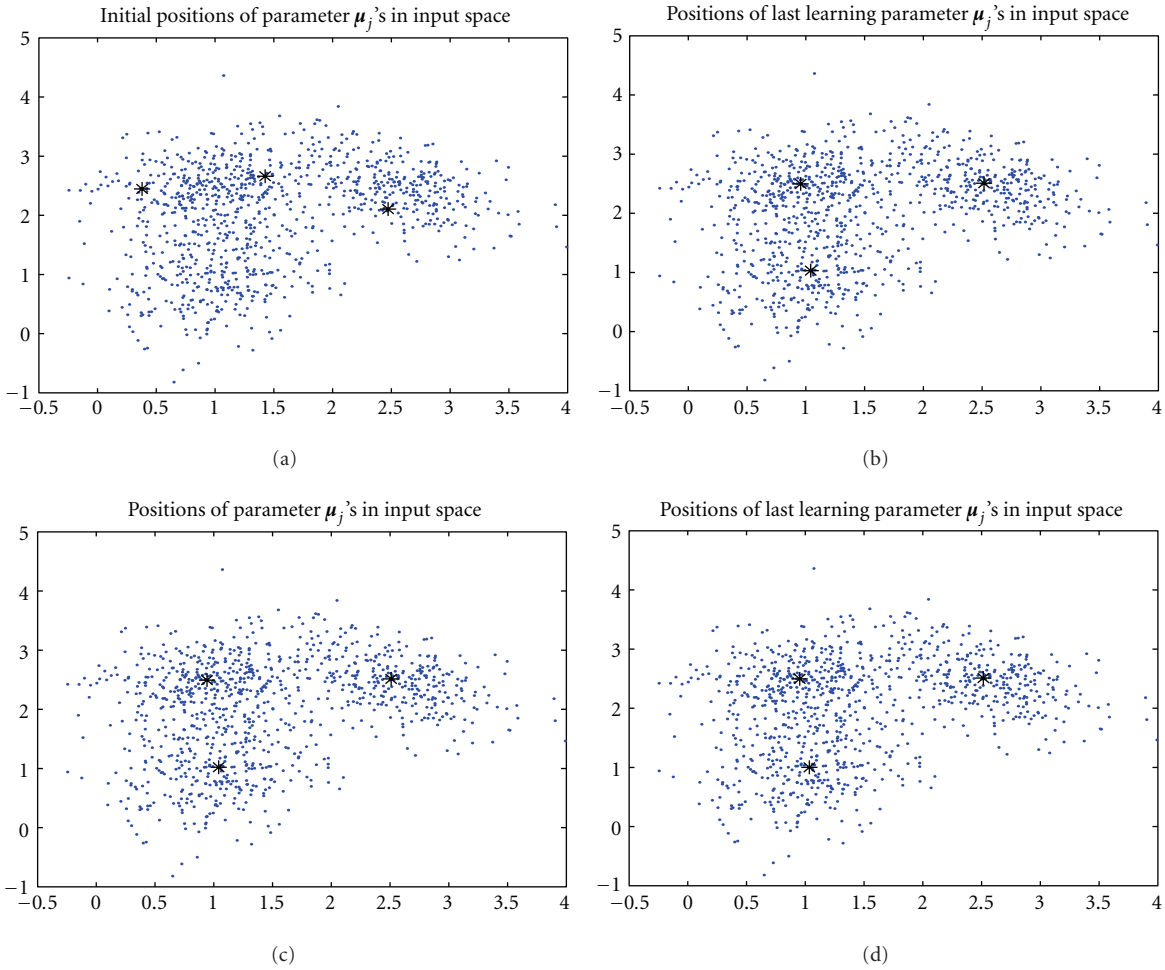


Figure 5: (a) The initial positions of the three seed points and their converged positions learned by (b) EM, (c) adaptive RPEM, and (d) batch RPEM, respectively.

Table 1: Performance of the Batch RPEM over the Parameter ε, where "G" stands for a good model selection capability of the algorithm, while "P" represents a poor model selection capability.

ε	$k^* = 3,\ k = 3$		$k^* = 3,\ k = 8$		$k^* = 3,\ k = 20$	
	Data set 1	Data set 2	Data set 1	Data set 2	Data set 1	Data set 2
−0.9	G	G	G	G	G	G
−0.8	G	G	G	G	G	G
−0.7	G	G	G	G	G	G
−0.6	G	G	G	G	G	G
−0.5	G	P	G	P	G	G
−0.4	G	P	G	P	P	P
−0.3	G	P	G	P	P	P
−0.2	G	P	G	P	P	P
−0.1	G	P	G	P	P	P

We let $k = 3$, which is equal to the true mixture number $k^* = 3$. The three seed points were randomly allocated in the observation space as shown in Figure 5(a), where the data are considerably overlapped. Moreover, all α_j's and $\mathbf{C}_j$'s were initialized at $1/k$ and the identity matrix, respectively. Figure 5(d) shows the positions of the three converged seed points, which are all stably located at the corresponding cluster centers. For comparison, we also implemented the EM under the same experimental environment. Figure 5(b) shows that the EM had successfully located the three seed points as well as the batch RPEM.

Nevertheless, as shown in Figures 6(c) and 7, the batch RPEM converges at 20 epochs, while the EM needs 60 epochs as shown in Figure 6(a). That is, the convergence speed of the batch RPEM is significantly faster than the EM. This indicates that the intrinsic rival-penalization scheme of the batch RPEM, analogous to the RPCL [11], RPCCL [12], and the adaptive RPEM [14], is able to drive away the rival seed points so that they can be more quickly towards the other cluster centers. As a result, the batch RPEM converges much faster than the EM. Moreover, we also compared it with the adaptive RPEM, in which we set the learning rate $\eta = 0.001$. Figure 5(c) shows the convergent results of the seed points. It can be seen that the adaptive RPEM works quite well in this case, but it needs around 70 epochs as shown in Figure 6(b). Actually, the adaptive RPEM can be further speed up if an

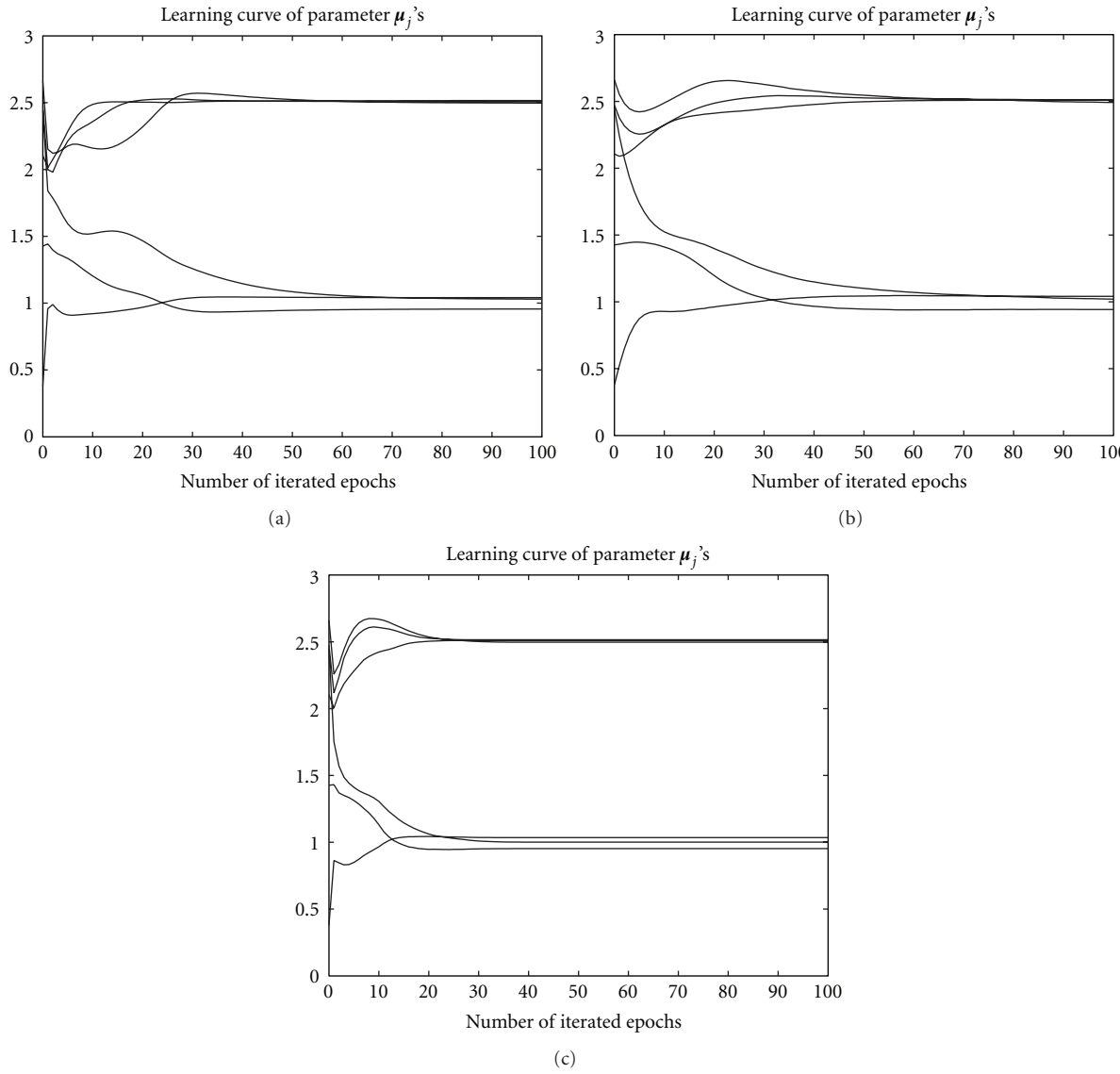


Figure 6: Learning curves of μ_j's by (a) EM, (b) adaptive RPEM, and (c) batch RPEM, respectively.

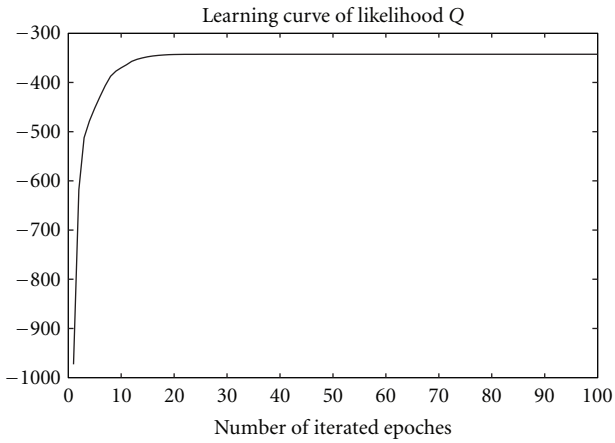


Figure 7: The value of the cost function Q over the epochs.

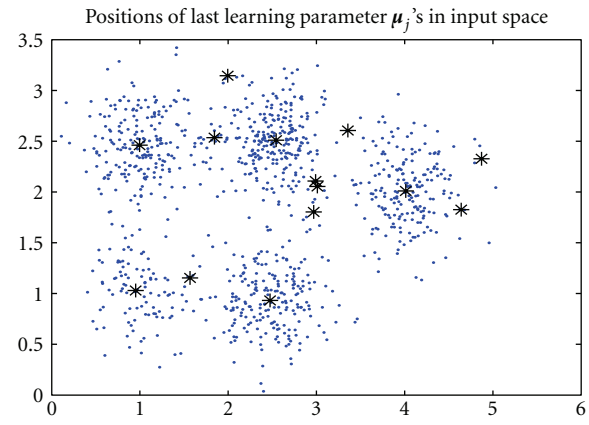


Figure 8: The converged positions of the seed points learned by the batch RPEM.

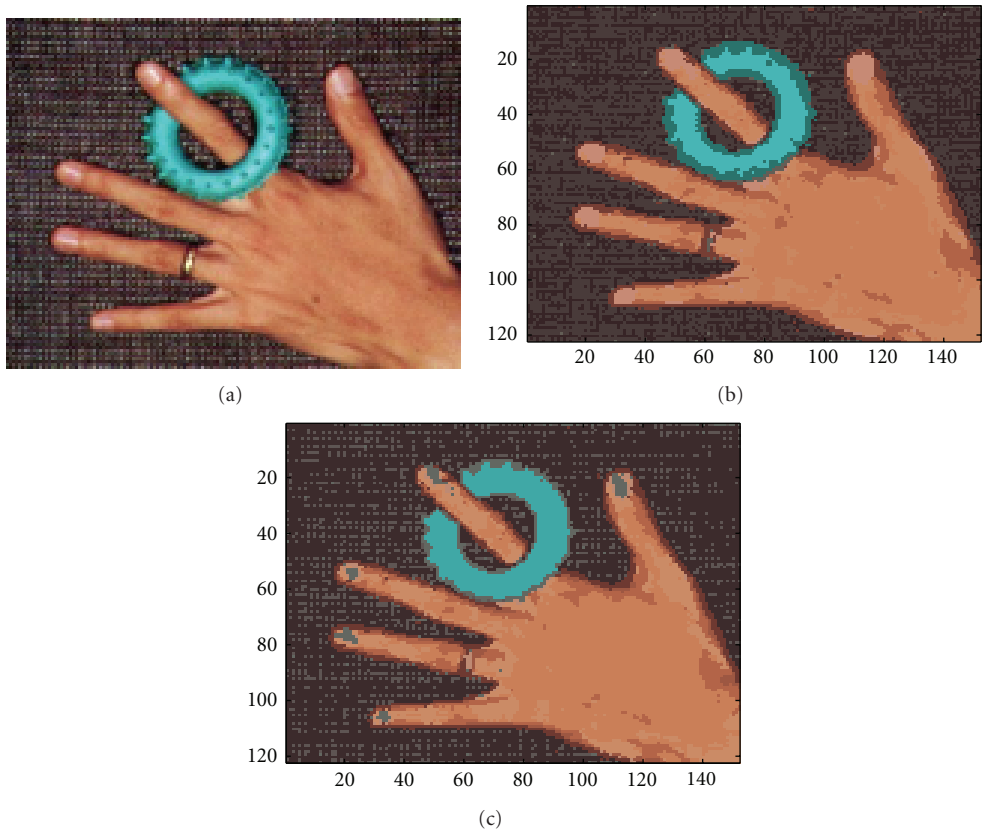


Figure 9: Segmentation of the hand image: (a) original image, (b) the result given by the EM, and (c) the result given by the batch RPEM.

Figure 10: The original house image.

appropriate learning rate is adopted, which, however, is not a trivial task.

5.2. Experiment 2: Batch RPEM on Synthetic Data with $K > K^*$. This experiment will investigate the performance of batch RPEM performance as $k > k^*$. We generated 1,000 observations from a mixture of five bivariate Gaussian density distributions with the mixing proportions:

$$\alpha_1^* = 0.1, \quad \alpha_2^* = 0.2, \quad \alpha_3^* = 0.3, \quad \alpha_4^* = 0.2, \quad \alpha_5^* = 0.2 \tag{17}$$

and the true cluster centers:

$$\boldsymbol{\mu}_1^* = [1.0, 1.0]^T, \quad \boldsymbol{\mu}_2^* = [1.0, 2.5]^T, \quad \boldsymbol{\mu}_3^* = [2.5, 2.5]^T, \quad \boldsymbol{\mu}_4^* = [2.5, 1.0]^T, \quad \boldsymbol{\mu}_5^* = [4.0, 2.0]^T. \tag{18}$$

15 seed points were initialized in the input space arbitrarily. During the learning, the three density components were discarded because their corresponding covariances became singular. As a result, the remaining 12 converged proportions were $\alpha_1 = 0.0065$, $\alpha_2 = 0.0113$, $\alpha_3 = 0.1929$, $\alpha_4 = 0.0030$, $\alpha_5 = 0.0068$, $\alpha_6 = 0.2013$, $\alpha_7 = 0.2084$, $\alpha_8 = 0.0074$, $\alpha_9 = 0.0083$, $\alpha_{10} = 0.0986$, $\alpha_{11} = 0.2531$, and $\alpha_{12} = 0.0022$. It can be seen that the five principal values α_3, α_6,α_7, α_{10}, and α_{11} were estimated well, while the others were learned towards zero. A snapshot of the corresponding $\boldsymbol{\mu}_j$'s were $\boldsymbol{\mu}_3 = [4.0348, 2.0075]^T$, $\boldsymbol{\mu}_6 = [0.9990, 2.4571]^T$, $\boldsymbol{\mu}_7 = [2.4725, 0.9220]^T$, $\boldsymbol{\mu}_{10} = [0.9553, 1.0277]^T$, and $\boldsymbol{\mu}_{11} = [2.5189, 2.5199]^T$. As shown in Figure 8, these five seed points have successfully allocated in the cluster centers, meanwhile the batch RPEM drove the redundant seed points to the boundaries of the clusters.

5.3. Experiment 3: Batch RPEM on Color Image Segmentation. This experiment further investigated the batch RPEM algorithm on color image segmentation in comparison to the EM algorithm. We implemented the image segmentation in

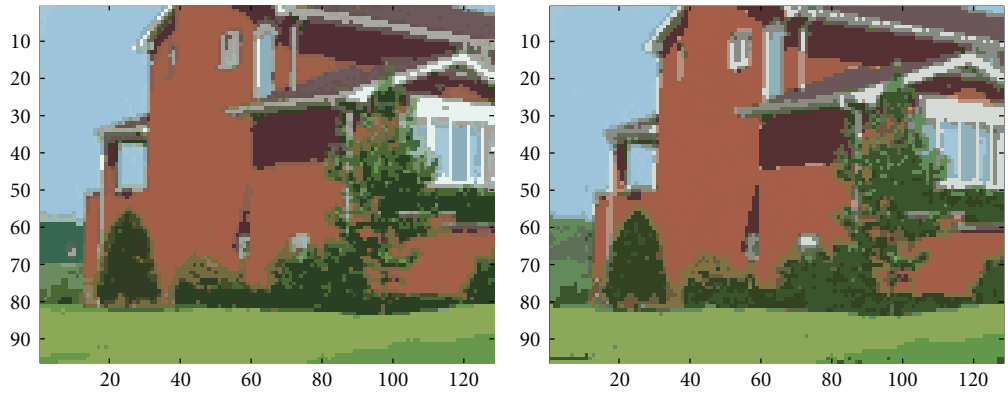


Figure 11: Segmentation of the house image by (a) EM; (b) batch RPEM.

the red-green-blue (RGB) color space model that represents each pixel in an image by a three-color vector. We conducted color image segmentation on a 122×152 hand image and a 96×128 house image as shown in Figures 9(a) and 10, respectively. For the former, we initially assigned 10 seed points randomly. After the convergence of the algorithms' performance, a snapshot of their segmentation results is shown in Figures 9(b) and 9(c). It can be seen that the blue tiny swim ring-shaped region after segmentation process by the batch RPEM is smoother than the EM. Further, the tiny nail regions have been partitioned by the batch RPEM but the EM is not. In other words, the batch RPEM algorithm performs better than the EM algorithm.

For the house image, we initially assigned the seed points to be 80. A snapshot of the converged segmentation results of the EM and the batch RPEM is shown in Figure 11. It can be seen that the texture on the red wall and the green lawn has no longer maintained after the segmentation process both by the EM and the RPEM. However, the small white regions of windows on red wall were disappeared by the EM as well as the triangle shadow area on the wall. In contrast, the batch RPEM algorithm partitioned these regions well as shown in Figure 11(b). Actually, the batch RPEM has drove out the redundant seed points far away and maintained some principal components correctly, which therefore leads to a better performance in color image segmentation.

6. Conclusion

In this paper, we have developed a batch RPEM algorithm based on MWL learning framework for Gaussian mixture clustering. Compared to the adaptive RPEM, this new one need not select the value of learning rate. As a result, it can learn faster in general and still preserve the capability of automatic model selection analogous to the adaptive one. We have evaluated the proposed batch RPEM algorithm on both synthetic data and color image segmentation. The numerical results have shown the efficacy of the proposed algorithm.

Acknowledgments

This work was jointly supported by the grants from the Research Grant Council of the Hong Kong SAR with the Project Code: HKBU 210309, the Natural Science Foundation of China (60974077), the Natural Science Foundation of Guangdong Province (s2011010005075), Guangzhou Technology Projects (11c42110781), the Grants 60403011 and 60973154 from the NSFC, and NCET-07-0338 from the Ministry of Education, China. This work was also partially supported by the Fundamental Research Funds for the Central Universities, HUST:2010ZD025, and Hubei Provincial Science Foundation under Grant 2010CDA006, China.

References

[1] U. Fayyad, G. Piatetsky-Shpiro, P. Smyth, and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*, MIT Press, 1996.

[2] B. Fritzke, "The LBG-U method for vector quantization—an improvement over LBG inspired from neural networks," *Neural Processing Letters*, vol. 5, no. 1, pp. 35–45, 1997.

[3] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, 1980.

[4] S. Y. Chen, H. Y. Tong, Z. J. Wang, S. Liu, M. Li, and B. W. Zhang, "Improved generalized belief propagation for vision processing," *Mathematical Problems in Engineering*, Article ID 416963, 12 pages, 2011.

[5] S. Y. Chen, H. Y. Tong, and C. Cattani, "Markov models for image labeling," *Mathematical Problems in Engineering*, vol. 2012, Article ID 814356, 18 pages, 2012.

[6] Y. W. Lim and S. U. Lee, "On the color image segmentation algorithm based on the thresholding and the fuzzy C-means techniques," *Pattern Recognition*, vol. 23, no. 9, pp. 935–952, 1990.

[7] T. Uchiyama and M. A. Arib, "Color image segmentation using competitive learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 12, pp. 1197–1206, 1994.

[8] J. M. Jolion, P. Meer, and S. Bataouche, "Robust clustering with applications in computer vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 8, pp. 791–802, 1991.

[9] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society B*, vol. 39, no. 1, pp. 1–38, 1977.

[10] Bilmes A. Jeff, *A Gentle Tutorial of the EM Algorithm and Its Application to Parameter Estimation for Gausssian Mixture and Hidden Markov Models*, International Computer Science

Institute, and Computer Science Division, Department of Electrical Engineering and Computer Science, Berkeley, Calif, USA, 1998, TR-97-021.

[11] L. Xu, A. Krzyzak, and E. Oja, "Rival penalized competive learning for clustering analysis, RBF Net, and curve detection," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 636–649, 1993.

[12] Y. M. Cheung, "Rival penalized controlled competitive learning for data clustering with unknown cluster number," in *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP '02)*, 2002.

[13] Y. M. Cheung, "A rival penalized EM algorithm towards maximizing weighted likelihood for density mixture clustering with automatic model selection," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR '04)*, vol. 4, pp. 633–636, Cambridge, UK, 2004.

[14] Y. M. Cheung, "Maximum weighted likelihood via rival penalized EM for density mixture clustering with automatic model selection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 750–761, 2005.

[15] X. M. Zhao, Y. M. Cheung, L. Chen, and K. Aihara, "A new technique for adjusting the learning rate of RPEM algorithm automatically," in *Proceedings of the 12th International Symposium on Artificial Life and Robotics*, pp. 597–600, Oita, Japan, January 2007.