Concise Papers

On Weight Design of Maximum Weighted Likelihood and an Extended EM Algorithm

Zhenyue Zhang and
Yiu-ming Cheung, *Senior Member*, *IEEE*

Abstract—The recent Maximum Weighted Likelihood (MWL) [18], [19] has provided a general learning paradigm for density-mixture model selection and learning, in which weight design, however, is a key issue. This paper will therefore explore such a design, and through which a heuristic extended Expectation-Maximization (X-EM) algorithm is presented accordingly. Unlike the EM algorithm [1], the X-EM algorithm is able to perform model selection by fading the redundant components out from a density mixture, meanwhile estimating the model parameters appropriately. The numerical simulations demonstrate the efficacy of our algorithm.

Index Terms—Maximum weighted likelihood, weight design, extended expectation-maximization algorithm, model selection.

1 Introduction

As a general solution for the parameter estimate in a density mixture model, the Expectation-Maximization (EM) algorithm [1] and its variants, e.g., see [2], [3], [4], [5], [6] have been extensively applied to a variety of applications such as data clustering [7], Bayesian network [8], hidden Markov model [9], bioinformatics [10], and so forth. Nevertheless, the EM is unable to make a model selection, i.e., to determine the appropriate number of model components in a density mixture. If the number of components is not correctly assigned, the EM will generally lead to a poor estimate of the model parameters. In general, it is a nontrivial task to determine such a number.

In the literature, Bayes factors [11] have provided a general framework for model comparisons, which gives a systematic means of selecting not only the parameterization of the model, but also the number of components. From Bayes factors, there are some typical model selection criteria, e.g., AIC [12], [13], BIC [14], CAIC [15], and SIC [14]. Generally, the EM algorithm and these criteria are used separately for the parameter estimation and model selection. Alternatively, another interesting way is to learn the parameter estimation and model selection jointly in a single paradigm. One example is the Reversible Jump Markov Chain Monte Carlo (RJMCMC) method proposed by Green [16], which is essentially a random sweep Metropolis-Hastings method. The RJMCMC algorithm has been successfully applied to the Gaussian mixture model (GMM) [17]. Another example is the Competitive EM (CEM) [5]. Analogous to the SMEM algorithm [4], the CEM utilizes a heuristic split-and-merge mechanism to either split the model components in an underpopulated region or merge the components in an overpopulated region iteratively so that the algorithm is able to avoid local solutions. Furthermore, the CEM also exploits a heuristic component annihilation mechanism to determine the number of model components. The experiments in [5] have shown the promising results of this method. Nevertheless, the computations of the CEM are rather heavier than the EM. In particular, its split-and-merge mechanism introduces two new parameters, which have close relations with the performance of the CEM. To the best of our knowledge, how to determine their values is still an open problem from the theoretical viewpoint.

Recently, the second author of this paper has proposed a new novel learning framework, namely, Maximum Weighted Likelihood (MWL) [18], [19]. With a specific weight design, it has been shown that the MWL is able to select the models automatically during the parameter learning process. Nevertheless, the MWL leaves two open questions: 1) How to design the weights so that an MWL algorithm is able to perform model selection? 2) What are the general convergence properties of an MWL algorithm, although the convergence of a specific MWL algorithm proposed in [18], [19] has been guaranteed?

In this paper, we will concentrate on the first question only to explore the weight design, and through which a heuristic extended EM (X-EM) algorithm is presented accordingly. Keeping the number of model components unchanged, this new algorithm updates the parameters analogous to the EM [1], and is able to perform the model selection by fading the redundant components out from a density mixture during the parameter learning process. Experimental results have shown the efficacy of our algorithm.

2 The Framework of MWL Learning

Given a series of N observations: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, that are all independently and identically distributed from a mixture distribution of k^* probability densities, denoted as $p(\mathbf{x}|\Theta^*)$, the Maximum Likelihood (ML) estimate of true model parameter set Θ^*, denoted as Θ, can be obtained via maximizing the following cost function

$$\ell(\Theta) = \sum_{t=1}^{N} \ln p(\mathbf{x}_t|\Theta), \tag{1}$$

with

$$p(\mathbf{x}_t|\Theta) = \sum_{j=1}^{k} \alpha_j p(\mathbf{x}_t|\boldsymbol{\theta}_j), \sum_{j=1}^{k} \alpha_j = 1, \text{ and } \alpha_j \geq 0 \text{ for } \forall j, \tag{2}$$

where k is an estimate of the true mixture number k^*, and $\Theta = \{\alpha_j, \boldsymbol{\theta}_j\}_{j=1}^{k}$. Hereinafter, we suppose that $p(\mathbf{x}|\Theta)$ is an identifiable density with respect to Θ, i.e., for any two possible values of Θ, denoted as Θ_1 and Θ_2, $p(\mathbf{x}|\Theta_1) = p(\mathbf{x}|\Theta_2)$ if and only if $\Theta_1 = \Theta_2$. In general, after preassigning the value of k, the learning of Θ can be achieved via the EM algorithm [1] whose major steps can be summarized as follows:

E-Step: Let $\Theta^{(n)}$ be the estimate of Θ^* at the nth iteration. For each observation $\mathbf{x}_t$ with $t = 1, 2, \ldots, N$, calculate the posterior probability

$$h(j|\mathbf{x}_t, \Theta^{(n)}) = \frac{\alpha_j^{(n)} p(\mathbf{x}_t|\boldsymbol{\theta}_j^{(n)})}{p(\mathbf{x}_t|\Theta^{(n)})}, \quad \text{with} \quad j = 1, 2, \ldots, k. \tag{3}$$

M-Step: Fixing all $h(j|\mathbf{x}_t, \Theta^{(n)})$s, the new improved estimate of Θ^* is then:

$$\Theta^{(n+1)} = \arg\max_{\Theta} \sum_{t=1}^{N} \sum_{j=1}^{k} h(j|\mathbf{x}_t, \Theta^{(n)}) \ln[\alpha_j p(\mathbf{x}_t|\boldsymbol{\theta}_j)]. \tag{4}$$

However, if k^* is miss-estimated, the EM algorithm almost always leads to a poor estimate of Θ^*. For example, if $k > k^*$, the EM will regard a true density as a mixture of two or more densities, and has no mechanism to push those redundant α_js toward zero. As a result, the estimate of Θ^* is poor.

To address this problem, papers [18], [19] have recently proposed a new learning framework, which gives the ML estimate $\hat{\Theta}$ of Θ^* via maximizing

- *Z. Zhang is with the Department of Mathematics, Zhejiang University, China. E-mail: zyzhang@zju.edu.cn.*
- *Y.-m. Cheung is with the Department of Computer Science, Rm. 709, 7/f. Sir Run Run Shaw Building, Hong Kong Baptist University, Kowloon Tong, Kowloon, Hong Kong, P.R. China. E-mail: ymc@comp.hkbu.edu.hk.*

Manuscript received 21 Mar. 2005; revised 6 Sept. 2005; accepted 16 May 2006; published online 18 Aug. 2006.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0101-0305.

$$\begin{aligned}\ell(\Theta) &= \frac{1}{N}\sum_{t=1}^{N}\ln p(\mathbf{x}_t|\Theta)\\ &= \frac{1}{N}\sum_{t=1}^{N}\sum_{j=1}^{k} g(j|\mathbf{x}_t,\Theta)\ln p(\mathbf{x}_t|\Theta)\\ &= \frac{1}{N}\sum_{t=1}^{N}\sum_{j=1}^{k} g(j|\mathbf{x}_t,\Theta)\ln[\alpha_j p(\mathbf{x}_t|\boldsymbol{\theta}_j)]\\ &\quad - \frac{1}{N}\sum_{t=1}^{N}\sum_{j=1}^{k} g(j|\mathbf{x}_t,\Theta)\ln h(j|\mathbf{x}_t,\Theta),\end{aligned} \tag{5}$$

where k is a preassigned constant with $k \geq k^*$, and $g(j|\mathbf{x},\Theta)$s are designable weight functions, satisfying

$$\forall \mathbf{x}: \quad \sum_{j=1}^{k} g(j|\mathbf{x},\Theta) = 1, \tag{6}$$

where each of $g(j|\mathbf{x},\Theta)$s can be negative, zero, or positive.

Equation (5) is named the *Weighted Likelihood* function because the parameter learning of each model component, i.e., the updating of α_j and $\boldsymbol{\theta}_j$ in the jth density $p(\mathbf{x}_t|\boldsymbol{\theta}_j)$, are weighted by the associated weights as shown in the first term of (5). To guarantee the second term in (5) is bounded even if $h(j|\mathbf{x},\Theta)$ tends to zero, the constraints on $g(j|\mathbf{x},\Theta)$ are therefore imposed as follows:

$$\begin{aligned}&\forall\, j,\ g(j|\mathbf{x},\Theta) = 0 \text{ if } h(j|\mathbf{x},\Theta) = 0, \text{ and}\\ &\left\{\frac{g(j|\mathbf{x},\Theta)}{h(j|\mathbf{x},\Theta)},\ h(j|\mathbf{x},\Theta)\neq 0\right\} \text{ is bounded.}\end{aligned} \tag{7}$$

As soon as the function form of each $g(j|\mathbf{x},\Theta)$ is specified, we can learn Θ toward maximizing (5) analogous to the EM. That is, by putting $\Theta^{(n)}$ to be an estimate of Θ in $h(j|\mathbf{x},\Theta)$ and $g(j|\mathbf{x},\Theta)$ of (5), a new approximation $\Theta^{(n+1)}$ can be obtained with

$$\Theta^{(n+1)} = \arg\max_{\Theta}\sum_{t=1}^{N}\sum_{j=1}^{k} g(j|\mathbf{x}_t,\Theta^{(n)})\ln[\alpha_j p(\mathbf{x}_t|\boldsymbol{\theta}_j)]. \tag{8}$$

Such a learning via (5) and (8) is named the Maximum Weighted Likelihood (MWL) learning approach [18], [19]. It can be seen that (8) is generally different from (4) unless $g(j|\mathbf{x},\Theta) \equiv h(j|\mathbf{x}_t,\Theta)$ for any value of j and $\mathbf{x}_t$. It has been shown [18], [19] that the MWL learning is able to gradually decrease $k-k^*$ redundant α_js toward zero and push the corresponding redundant $\mathbf{m}_j$s away from the dense regions of observations as long as the weight functions $g(j|\mathbf{x},\Theta)$s are designed appropriately.

In the following, we will assume each of $g(j|\mathbf{x},\Theta)$s to be nonnegative and study their design, through which an extended EM (X-EM) algorithm is presented within the framework of MWL. For simplicity, this paper concentrates on the GMM only, i.e, $p(\mathbf{x}_t|\boldsymbol{\theta}_j) = G(\mathbf{x}_t|\mathbf{m}_j,\Sigma_j)$ for each $j = 1, 2, \ldots, k$, where $G(\mathbf{x}|\mathbf{m}_j,\Sigma_j)$ denotes the Gaussian probability density function of an observation $\mathbf{x}$ with the mean $\mathbf{m}_j$ (also called *seed point* hereinafter) and covariance matrix Σ_j.

3 Weight Design in the MWL Learning

Let $p(\mathbf{x}|\boldsymbol{\theta}_j)$ be a Gaussian density, denoted as $G(\mathbf{x}|\mathbf{m}_j,\Sigma_j)$, and $\Theta^{(n)} = \{\alpha_j^{(n)}, \mathbf{m}_j^{(n)}, \Sigma_j^{(n)}\}_{j=1}^{k}$ be an estimate of $\Theta^* = \{\alpha_j^*, \mathbf{m}_j^*, \Sigma_j^*\}_{j=1}^{k^*}$ at the nth iterative step. If the weight functions $g(j|\mathbf{x},\Theta)$s are well designed, it can be shown that the optimal solution $\Theta^{(n+1)} = \{\alpha_j^{(n+1)}, \mathbf{m}_j^{(n+1)}, \Sigma_j^{(n+1)}\}_{j=1}^{k}$ of (8) is given by

$$\begin{aligned}\alpha_j^{(n+1)} &= \frac{N_j^{(n)}}{N},\ \ \mathbf{m}_j^{(n+1)} = \frac{1}{N_j^{(n)}}\sum_{t=1}^{N} w_{j,t}^{(n)}\mathbf{x}_t,\\ \Sigma_j^{(n+1)} &= \frac{1}{N_j^{(n)}}\sum_{t=1}^{N} w_{j,t}^{(n)}(\mathbf{x}_t-\mathbf{m}_j^{(n)})(\mathbf{x}_t-\mathbf{m}_j^{(n)})^T,\end{aligned} \tag{9}$$

where $N_j^{(n)} = \sum_{t=1}^{N} w_{j,t}^{(n)}$, and $w_{j,t}^{(n)}$ is short for $g(j|\mathbf{x}_t,\Theta^{(n)})$.

To fix the problem of overestimating k^*, the weight function $g(j|\mathbf{x},\Theta)$ should be designed such that the $k-k^*$ redundant $\alpha_j^{(n+1)}$s are forced toward zero as n increases. By the definition of $\alpha_j^{(n+1)}$ in (9), it is equivalent to force $(k-k^*)$ weights toward zero for each $\mathbf{x}_t$ as $n\to\infty$, while the sum of the other k^* weights tends to one because of the weight condition in (6).

With the weight constraint in (7), a reasonable way to design $g(j|\mathbf{x},\Theta)$s is that

$$g(j|\mathbf{x}_t,\Theta) = c_t f(h(j|\mathbf{x}_t,\Theta)) \text{ with } h(j|\mathbf{x}_t,\Theta) = \frac{\alpha_j G(\mathbf{x}_t|\mathbf{m}_j,\Sigma_j)}{\sum_{i=1}^{k}\alpha_i G(\mathbf{x}_t|\mathbf{m}_i,\Sigma_i)}, \tag{10}$$

where f is a nonnegative function defined in the interval $[0,\ 1]$ such that $f(0) = 0$, and $c_t = 1/\sum_{j=1}^{k} f(h(j|\mathbf{x}_t,\Theta))$ is a normalization term such that $\sum_{j=1}^{k} g(j|\mathbf{x}_t,\Theta) = 1$.

To push the value of redundant weights toward zero, f should be chosen in such a way that $g(j|\mathbf{x}_t,\Theta)$ becomes much smaller as $h(j|\mathbf{x}_t,\Theta)$ is small, while keeping $g(j|\mathbf{x}_t,\Theta)$ not to be decreased when $h(j|\mathbf{x}_t,\Theta)$ is relatively large. That is, $f(s) < s$ for small s and $f(s) > s$ for relatively large s.

Many functions can be used as a required f. We suggest using

$$f(s\mid\beta) = \frac{s^\beta}{s^\beta + (1-s)^\beta} \tag{11}$$

defined in the unit interval $[0,\ 1]$ with a constant $\beta \geq 1$. This function has some interesting properties. It can be seen that, as $\beta > 1$, we have

$$\begin{aligned}&0 < f(s\mid\beta) < s, \text{ for } 0 < s < 1/2 \text{ and}\\ &s < f(s\mid\beta) < 1, \text{ for } 1/2 < s < 1.\end{aligned}$$

It means that f attracts $s \neq 1/2$ moving toward the two ends of the interval $[0,\ 1]$. The value of β determines the degree of the attraction: the larger the β is, the stronger the attraction is.

We denote by $g(j|\mathbf{x}_t,\Theta,\beta)$ the weight function defined in (10) with f in (11). The flexible choice of parameter β will lead to different iteration schemes. Obviously, if $\beta = 1$, we have $f(s\mid\beta) = s$ and, thus, $g(j|\mathbf{x}_t,\Theta,\beta) = h(j|\mathbf{x}_t,\Theta)$. Furthermore, as $\beta\to+\infty$, this specific weight design leads the MWL learning to the existing hard-cut EM [20] provided that the maximum value of $h(j|\mathbf{x}_t,\Theta)$s is unique.

4 The X-EM Algorithm

As $k > k^*$, we have noticed that the EM almost always makes the seed points of those $k-k^*$ redundant model components close to the other seed points and compete with them. As a result, there are seldom seed points to be stabilized at the desired values. To avoid this awkward situation, we therefore introduce a new modifying term in updating the value of each seed point, say $\mathbf{m}_j$, so that it is moved opposite to each of the other $\mathbf{m}_i$s. We let the modifying term with respect to $\mathbf{m}_i$ be $\alpha_i G(\mathbf{m}_j|\mathbf{m}_i,\Sigma_i)(\mathbf{m}_i-\mathbf{m}_j)$, where we assume that the distribution of $\mathbf{m}_j$ conditioned on each other seed point, say $\mathbf{m}_i$, is $G(\mathbf{m}_j|\mathbf{m}_i,\Sigma_i)$, i.e, the Gaussian distribution with the mean $\mathbf{m}_i$ and covariance matrix Σ_i. Eventually, taking into account all modifying terms on $\mathbf{m}_j$, we then have

$$\hat{\mathbf{m}}_j = \mathbf{m}_j - \sum_{i=1}^{k}\alpha_i G(\mathbf{m}_j|\mathbf{m}_i,\Sigma_i)(\mathbf{m}_i-\mathbf{m}_j). \tag{12}$$

This modification has three interesting properties as follows:

Property 1. *If a seed $\mathbf{m}_i$ is far from $\mathbf{m}_j$ or α_i is small, $\mathbf{m}_i$ will have a slight contribution to updating $\mathbf{m}_j$ because*

$$\lim_{\|\mathbf{m}_j-\mathbf{m}_i\|\to+\infty} G(\mathbf{m}_j|\mathbf{m}_i,\Sigma_i)(\mathbf{m}_i-\mathbf{m}_j) = 0.$$

Property 2. *If $\mathbf{m}_i$ and $\mathbf{m}_j$ both closely locate at the dense region of the same Gaussian density and have a relative long distance to other $\mathbf{m}_i$s, we have*

$$\hat{\mathbf{m}}_i \approx \mathbf{m}_i - \alpha_j G(\mathbf{m}_i|\mathbf{m}_j, \mathbf{\Sigma}_j)(\mathbf{m}_j - \mathbf{m}_i),$$
$$\hat{\mathbf{m}}_j \approx \mathbf{m}_j - \alpha_i G(\mathbf{m}_j|\mathbf{m}_i, \mathbf{\Sigma}_i)(\mathbf{m}_i - \mathbf{m}_j),$$

and

$$\|\hat{\mathbf{m}}_i - \hat{\mathbf{m}}_j\| \approx \left[1 + \alpha_i G(\mathbf{m}_j|\mathbf{m}_i, \mathbf{\Sigma}_i) + \alpha_j G(\mathbf{m}_i|\mathbf{m}_j, \mathbf{\Sigma}_j)\right] \cdot \|\mathbf{m}_i - \mathbf{m}_j\| > \|\mathbf{m}_i - \mathbf{m}_j\|.$$

That is, the distance of $\hat{\mathbf{m}}_j$ to $\hat{\mathbf{m}}_i$ is greater than the distance of $\mathbf{m}_j$ to $\mathbf{m}_i$.

Property 3. *If $\alpha_j G(\mathbf{m}_i|\mathbf{m}_j, \mathbf{\Sigma}_j) \gg \alpha_i G(\mathbf{m}_j|\mathbf{m}_i, \mathbf{\Sigma}_i)$, $\mathbf{m}_i$ will be pushed away from $\mathbf{m}_j$, whereas $\mathbf{m}_j$ is almost unchanged because*

$$\|\hat{\mathbf{m}}_i - \mathbf{m}_i\| \approx \alpha_j G(\mathbf{m}_i|\mathbf{m}_j, \mathbf{\Sigma}_j) > \alpha_i G(\mathbf{m}_j|\mathbf{m}_i, \mathbf{\Sigma}_i) \approx \|\hat{\mathbf{m}}_j - \mathbf{m}_j\|.$$

Following **Property 2** and **Property 3**, it can be seen that two or more seeds will not be stabilized at the same dense region of a Gaussian density any more. That is, such a modification can speed up the process of automatic model selection via maximizing (5). Further, even if $k = k^*$, this seed driving mechanism can let the seed points move faster to get into their desired positions compared to the EM, i.e., it can speed up the learning of model parameters as well.

As soon as all the seeds are updated, we modify the covariance matrices by

$$\hat{\mathbf{\Sigma}}_j = \frac{\sum_{t=1}^N h_{j,t}(\mathbf{x}_t - \hat{\mathbf{m}}_j)(\mathbf{x}_t - \hat{\mathbf{m}}_j)^T}{\sum_{t=1}^N h_{j,t}} \text{ with } h_{j,t} = h(j|\mathbf{x}_t, \mathbf{\Theta}).$$

Furthermore, the weight function (6) is changed to

$$g(j|\mathbf{x}_t, \mathbf{\Theta}, \beta) = c_t f(h(j|\mathbf{x}_t, \hat{\mathbf{\Theta}})|\beta) \text{with}$$
$$h(j|\mathbf{x}_t, \hat{\mathbf{\Theta}}) = \frac{\alpha_j G(\mathbf{x}_t|\hat{\mathbf{m}}_j, \hat{\mathbf{\Sigma}}_j)}{\sum_{i=1}^k \alpha_i G(\mathbf{x}_t|\hat{\mathbf{m}}_i, \hat{\mathbf{\Sigma}}_i)},$$

where $c_t = 1/\sum_{j=1}^k f(h(j|\mathbf{x}_t, \hat{\mathbf{\Theta}})|\beta)$. Subsequently, we give out the details of X-EM as follows:

Step 1: At the nth iterative step, we modify the seeds and covariance matrices by:

$$\gamma_{j,i}^{(n)} = \alpha_i^{(n)} G\left(\mathbf{m}_j^{(n)}|\mathbf{m}_i^{(n)}, \mathbf{\Sigma}_i^{(n)}\right),$$
$$\hat{\mathbf{m}}_j^{(n)} = \mathbf{m}_j^{(n)} - \sum_{i=1}^k \gamma_{j,i}^{(n)}\left(\mathbf{m}_i^{(n)} - \mathbf{m}_j^{(n)}\right),$$
$$h_{j,t}^{(n)} = \frac{\alpha_j^{(n)} G\left(\mathbf{x}_t|\mathbf{m}_j^{(n)}, \mathbf{\Sigma}_j^{(n)}\right)}{\sum_{i=1}^k \alpha_i^{(n)} G\left(\mathbf{x}_t|\mathbf{m}_i^{(n)}, \mathbf{\Sigma}_i^{(n)}\right)},$$
$$\hat{\mathbf{\Sigma}}_j^{(n)} = \frac{\sum_{t=1}^N h_{j,t}^{(n)}\left(\mathbf{x}_t - \hat{\mathbf{m}}_j^{(n)}\right)\left(\mathbf{x}_t - \hat{\mathbf{m}}_j^{(n)}\right)^T}{\sum_{t=1}^N h_{j,t}^{(n)}}.$$

Step 2: We construct the weights by

$$w_{j,t}^{(n)} = \frac{r_{j,t}^{(n)}}{\sum_{i=1}^k r_{i,t}^{(n)}},$$
$$\hat{h}_{j,t}^{(n)} = \frac{\alpha_j^{(n)} G(\mathbf{x}_t|\hat{\mathbf{m}}_j^{(n)}, \hat{\mathbf{\Sigma}}_j^{(n)})}{\sum_{i=1}^k \alpha_i^{(n)} G(\mathbf{x}_t|\hat{\mathbf{m}}_i^{(n)}, \hat{\mathbf{\Sigma}}_i^{(n)})},$$
$$r_{j,t}^{(n)} = \frac{(\hat{h}_{j,t}^{(n)})^\beta}{(\hat{h}_{j,t}^{(n)})^\beta + (1 - \hat{h}_{j,t}^{(n)})^\beta}.$$

Step 3: Let $N_j^{(n)} = \sum_{t=1}^N w_{j,t}^{(n)}$, we update

$$\alpha_j^{(n+1)} = \frac{N_j^{(n)}}{N}, \ \mathbf{m}_j^{(n+1)} = \frac{1}{N_j^{(n)}}\sum_{t=1}^N w_{j,t}^{(n)}\mathbf{x}_t,$$
$$\mathbf{\Sigma}_j^{(n+1)} = \frac{1}{N_j^{(n)}}\sum_{t=1}^N w_{j,t}^{(n)}\left(\mathbf{x}_t - \mathbf{m}_j^{(n)}\right)\left(\mathbf{x}_t - \mathbf{m}_j^{(n)}\right)^T.$$

The above three steps are iteratively implemented until the parameters α_j, $\mathbf{m}_j$, and $\mathbf{\Sigma}_j$ converge. It can be seen that as soon as one $\mathbf{m}_i$ moves away from the dense region of a Gaussian density, the value of $G(\mathbf{x}_t|\hat{\mathbf{m}}_i, \hat{\mathbf{\Sigma}}_i)$ will be reduced at the most time, and, thus, α_i will be decreased sharply. It finally leads to a relatively small α_i. Consequently, we have: 1) if $k > k^*$, the $k - k^*$ redundant α_i's will be reduced to small and 2) two or more converged seed points cannot stabilize at the same Gaussian density. To demonstrate this scenario, we show an example by using a mixture of three Gaussians. We set $k = 4 > k^* = 3$ and the parameters of the first two density components are initialized at the true values, i.e.,

$$\{\alpha_i^{(0)}, \mathbf{m}_i^{(0)}, \mathbf{\Sigma}_i^{(0)}\} = \{\alpha_i^*, \mathbf{m}_i^*, \mathbf{\Sigma}_i^*\}, \quad i = 1, 2.$$

The last true seed point $\mathbf{m}_3^*$ is copied with a small perturbation to form the other two initial guesses:

$$\mathbf{m}_3^{(0)} = \mathbf{m}_3^* + 0.001\mathbf{r}_1, \quad \mathbf{m}_4^{(0)} = \mathbf{m}_3^* + 0.001\mathbf{r}_2,$$

with

$$\alpha_3^{(0)} = \frac{\eta_1}{\eta_1 + \eta_2}\alpha_3^*, \quad \alpha_4^{(0)} = \frac{\eta_2}{\eta_1 + \eta_2}\alpha_3^*,$$

and $[\eta_1, \eta_2]^T = [0.5, 0.5]^T + 0.001\mathbf{r}_3$, where $\mathbf{r}_1$, $\mathbf{r}_2$, and $\mathbf{r}_3$ are the vectors whose components are randomly chosen from a Gaussian distribution in the interval $[-1, 1]$. We set $\mathbf{\Sigma}_3^{(0)} = \mathbf{\Sigma}_4^{(0)} = \mathbf{\Sigma}_3^*$. Fig. 1 shows a snapshot of four learned seed points and the corresponding α_is as the iteration number is 1, 6, 11, and 15, respectively, where the elliptical curves are the shapes of the covariance matrices. It can be seen from Fig. 1d that the redundant seed point, say $\mathbf{m}_i$, has been moved to the outside of the Gaussian dense region, and the corresponding α_i is 0.0059991 that is very close to zero. That is, the redundant ith component has been faded out from a mixture.

5 Numerical Simulations

5.1 Experiment 1

This experiment is to show the model selection capability of the X-EM and its robust performance of estimating the model parameters as $k > k^*$. We generated $N = 1{,}000$ data points from a mixture of three Gaussians whose parameters were:

$$\mathbf{m}_1^* = \begin{bmatrix}1.0\\0.5\end{bmatrix}, \ \mathbf{\Sigma}_1^* = \begin{bmatrix}0.15 & 0.05\\0.05 & 0.20\end{bmatrix}, \ \alpha_1^* = 0.45,$$
$$\mathbf{m}_2^* = \begin{bmatrix}-1.0\\2.5\end{bmatrix}, \ \mathbf{\Sigma}_2^* = \begin{bmatrix}0.25 & 0\\0 & 0.24\end{bmatrix}, \ \alpha_2^* = 0.35,$$
$$\mathbf{m}_3^* = \begin{bmatrix}2.0\\3.0\end{bmatrix}, \ \mathbf{\Sigma}_3^* = \begin{bmatrix}0.15 & -0.1\\-0.1 & 0.15\end{bmatrix}, \ \alpha_3^* = 0.2.$$

We set $k = 7$ that is greater than the true number of Gaussians. The initial $\alpha_i^{(0)}$, $i = 1, \ldots, k$ were taken to be equal, i.e., $\alpha_i^{(0)} = \frac{1}{k}$, $i = 1, \ldots, k$, and each initial seed point $\mathbf{m}_i^{(0)}$ was set at the mean of the all sample points. The seven covariances $\mathbf{\Sigma}_i^{(0)}$s were produced randomly as follows (in MATLAB form):

$$[\mathtt{Q}, \mathtt{R}] = \mathtt{qr}(1 - 2 * \mathtt{rand}(2));$$
$$\Sigma_j = Q * diag(rand(2, 1) + 0.1) * Q';$$

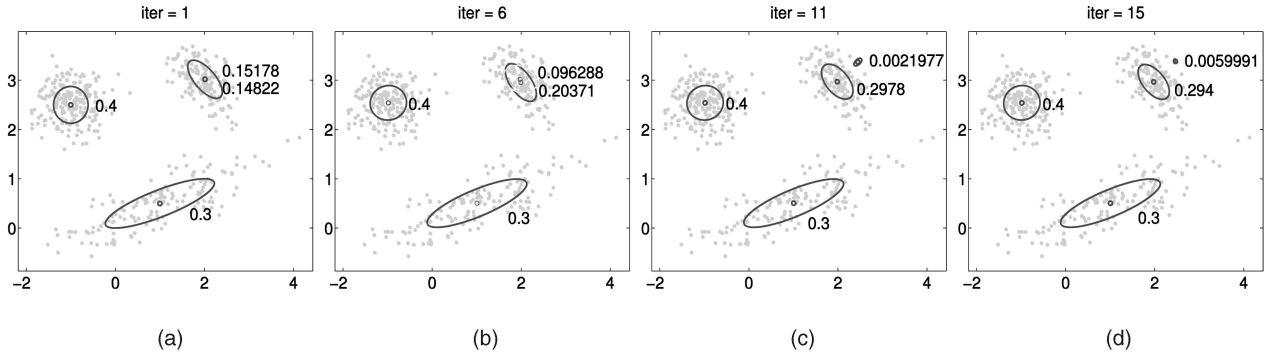

Fig. 1. The learned seed points and corresponding α_is and covariances by the X-EM with (a) 1, (b) 6, (c) 11, and (d) 15 iterations. The shapes of covariance matrices are plotted with the elliptical curves. The numbers marked near the curves are the values of corresponding α_js.

Fig. 2 plots the curves of the learned $\alpha_j^{(n)}$s. It shows that four out of seven α_js are quickly reduced toward zeros while the other three ones are convergent approximately to the true ones. Furthermore, Fig. 3 shows the positions of the learned seed points and the shapes of the covariance matrices corresponding to these three α_js. It can be seen that they are tightly close to the true ones. This implies that the X-EM has the robust performance of estimating the model parameters even though $k > k^*$ and, in particular, it can automatically determine the number of components by fading the redundant components out from the density mixture.

In this experiment, we set β at 2. Actually, we have noticed that the iteration number needed for the parameter convergence has a close relation with β as shown in Fig. 4. By rule of thumb, we empirically found that an appropriate choice of β is either $\beta = 2$ or $\beta = 3$. In the following, we will therefore set β at 2.

5.2 Experiment 2

To experimentally demonstrate the convergence speed of the model parameters learned by the X-EM in comparison to the EM, we generated $N = 1,000$ data points from a mixture of three Gaussians, whose parameters were:

$$\mathbf{m}_1^* = \begin{bmatrix} 1.0 \\ 0.5 \end{bmatrix}, \ \boldsymbol{\Sigma}_1^* = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 0.25 \end{bmatrix}, \ \alpha_1^* = 0.20,$$
$$\mathbf{m}_2^* = \begin{bmatrix} -0.5 - 2t \\ 2.5 - 0.5t \end{bmatrix}, \ \boldsymbol{\Sigma}_2^* = \begin{bmatrix} 0.25 & 0 \\ 0 & 0.24 \end{bmatrix}, \ \alpha_2^* = 0.35,$$
$$\mathbf{m}_3^* = \begin{bmatrix} 1.5 + 2t \\ 3.0 + 0.5t \end{bmatrix}, \ \boldsymbol{\Sigma}_3^* = \begin{bmatrix} 0.15 & -0.1 \\ -0.1 & 0.15 \end{bmatrix}, \ \alpha_3^* = 0.45,$$

where a preassigned parameter t controls the distance between $\mathbf{m}_2^*$ and $\mathbf{m}_3^*$. For instance, if $t = -0.5$, we have $\mathbf{m}_2^* = \mathbf{m}_3^*$. In general, the seed points $\mathbf{m}_2^*$ and $\mathbf{m}_3^*$ will gradually separate each other as t increases, while the shortest distance between $\mathbf{m}_i^*$ and $\mathbf{m}_1^*$, $i = 2, 3$ is no less than $\frac{1}{\sqrt{17}}$. We used six different values of t: $t = 0, 0.1, 0.2, 0.3, 0.4, 0.5$ to generate six input data sets. In each data set, we set $k = 3$, and initialized α_js, $\mathbf{m}_j$s and $\boldsymbol{\Sigma}_j$s in the same way as Experiment 1. We ran the X-EM and EM under the same experimental setting, including the same stopping criterion:

$$\|[\mathbf{m}_1^{(n+1)}, \mathbf{m}_2^{(n+1)}, \mathbf{m}_3^{(n+1)}] - [\mathbf{m}_1^{(n)}, \mathbf{m}_2^{(n)}, \mathbf{m}_3^{(n)}]\|_2 < \tau,$$

where τ is a preassigned threshold value. Table 1 lists the values of iteration numbers that are needed for the X-EM and EM to achieve the accuracy with $\tau = 10^{-6}$. It can be seen that the X-EM can significantly save the iteration times around 58.7 percent on average in comparison to the EM.

5.3 Experiment 3

To further investigate the sensitivity of the new algorithm to the parameter initialization, we ran the algorithm for 100 trials, in each of which the data were randomly generated from the common mixture of three Gaussians with the model parameters:

$$\mathbf{m}_1^* = \begin{bmatrix} 1.0 \\ 0.5 \end{bmatrix}, \ \boldsymbol{\Sigma}_1^* = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 0.25 \end{bmatrix}, \ \alpha_1^* = 0.45,$$
$$\mathbf{m}_2^* = \begin{bmatrix} -0.5 \\ 2.5 \end{bmatrix}, \ \boldsymbol{\Sigma}_2^* = \begin{bmatrix} 0.25 & 0 \\ 0 & 0.24 \end{bmatrix}, \ \alpha_2^* = 0.2,$$
$$\mathbf{m}_3^* = \begin{bmatrix} 1.5 \\ 3.0 \end{bmatrix}, \ \boldsymbol{\Sigma}_3^* = \begin{bmatrix} 0.15 & -0.1 \\ -0.1 & 0.15 \end{bmatrix}, \ \alpha_3^* = 0.35.$$

Similar to Experiment 1, we randomly initialized the covariances for each trial, and used the same stopping criterion as Experiment 2, i.e.,

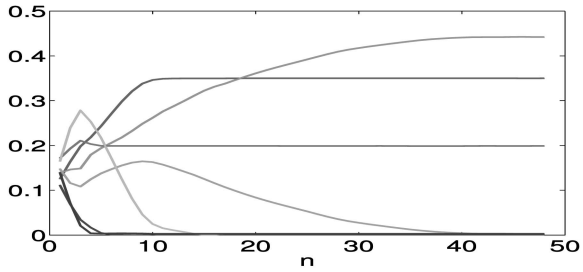

Fig. 2. The curves of all $\alpha_j^{(n)}$s learned by the X-EM with $\beta = 2$.

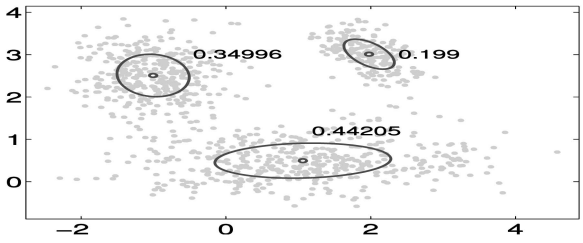

Fig. 3. The learned seed points (marked by $\circ$) of the three largest α_js and the shape curves of the corresponding covariance matrices by the X-EM with $\beta = 2$.

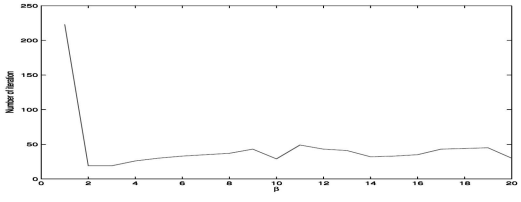

Fig. 4. The relation curve between iteration number needed for parameter convergence and β.

TABLE 1
The Iteration Number of the Parameter Learning via the X-EM and EM, Respectively

t	0	0.1	0.2	0.3	0.4	0.5
EM	57	66	33	26	24	24
X-EM	20	15	12	14	13	11

$$\|[\mathbf{m}_1^{(n+1)},,\mathbf{m}_k^{(n+1)}]-[\mathbf{m}_1^{(n)},,\mathbf{m}_k^{(n)}]\|_2<10^{-6}.$$

After the 100 trials, the minimum, mean, and maximum number of iterations needed are listed below, respectively:

$$n_{\min}=25,\quad n_{\text{mean}}=61,\quad n_{\max}=326.$$

Fig. 5 plots the date set we used and the three absolute errors for the learned parameters to the true ones, which are:

$$\epsilon_\alpha=\|[\alpha_{\pi_1},\alpha_{\pi_2},\alpha_{\pi_3}]-[\alpha_1^*,\alpha_2^*,\alpha_3^*]\|_2,$$
$$\epsilon_{\mathbf{m}}=\|[\mathbf{m}_{\pi_1},\mathbf{m}_{\pi_2},\mathbf{m}_{\pi_3}]-[\mathbf{m}_1^*,\mathbf{m}_2^*,\mathbf{m}_3^*]\|_2,$$
$$\epsilon_{\Sigma}=\|[\Sigma_{\pi_1},\Sigma_{\pi_2},\Sigma_{\pi_3}]-[\Sigma_1^*,\Sigma_2^*,\Sigma_3^*]\|_2,$$

where π_1,π_2,π_3 are three suitable indices corresponding to the three largest components of α_js. Since there is a slight difference between the true model parameters and the sample ones, the errors $\epsilon_\alpha,\epsilon_{\mathbf{m}},\epsilon_{\Sigma}$ cannot reach zero and have a lower bound as shown in Fig. 5. Nevertheless, it can be seen that these errors are small in the most cases. That is, the X-EM is insensitive to the parameter initialization to a certain degree.

5.4 Experiment 4

We also investigated the performance of X-EM on the input data with different overlapping levels. To save space, we ran three trials only as shown in the first row of Fig. 6. The second row shows the corresponding convergent curves of α_js. It can be seen that the X-EM can successfully work at all cases we have tried so far.

5.5 Experiment 5

In the previous experiments, we have demonstrated the outstanding performance of X-EM using synthetic data. In this experiment, we will further apply it to analyze the real-world microarray data—the yeast cell cycle data published by Cho et al. [21]. The data set we used (which can be downloaded from: http://www.cs.washington.edu/homes/kayee/model) consists of 384 genes, whose expression levels peak at different time points corresponding to the five phases of cell cycles (the five-phases criterion). Hence, it is expected that each of 384 genes can be assigned to one of the five clusters [21], [22]. In the literature, Yeung et al. [22] has successfully performed the microarray data analysis on this data set using a finite density mixture model, in which the BIC is utilized for model selection and the EM is applied for the parameter estimation. We call this method *EM algorithm + BIC* (Method II) for short, and compare it with the X-EM algorithm (called *Method I* hereinafter) as well as the other two existing methods: the supervised clustering method (Method III) [23] and the support vector machines (SVM) algorithm (Method IV) [24].

Suppose the true number of model components is unknown, we arbitrarily set the number of the seed points at 8 in the running of the X-EM. To measure the performance of these four methods, we utilized four indices: false positive (FP), false negative (FN), true positive (TP), and true negative (TN). The total error rate was defined as FP + FN [23]. Table 2 summarizes the results of these four methods, where the results of Method II-IV are obtained from [23]. In addition, Table 3 summarizes the total error rates of the four methods. It can be seen that the X-EM algorithm outperforms the other three methods. Further, Fig. 7 shows the five groups formed by the X-EM algorithm, in which it is indeed that the genes with the similar patterns have been classified as a group together.

6 Concluding Remarks

In this paper, we have further studied the weight design within the framework of MWL, through which the X-EM algorithm has been developed. In the X-EM, we always keep the number k of model components unchanged and greater than or equal to the true k^*. Compared to the EM, this new algorithm not only learns the model parameter faster, but is also able to perform model selection by automatically fading the redundant components from a mixture. The numerical simulations on synthetic and real-life data have shown the efficacy of this algorithm. In the future, we will further investigate and analyze the convergence properties of the X-EM.

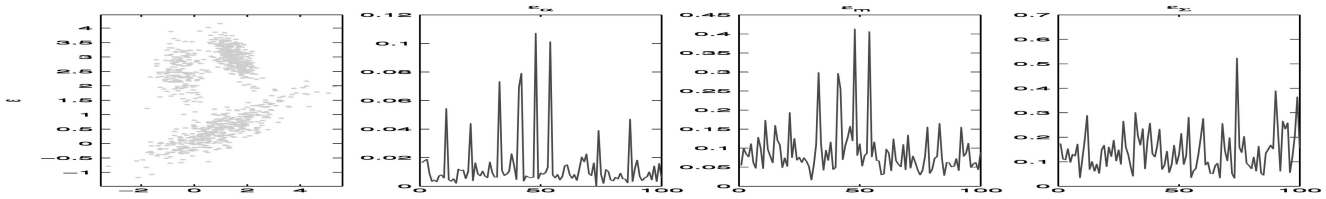

Fig. 5. A sample of the data set and absolute errors for 100 tests.

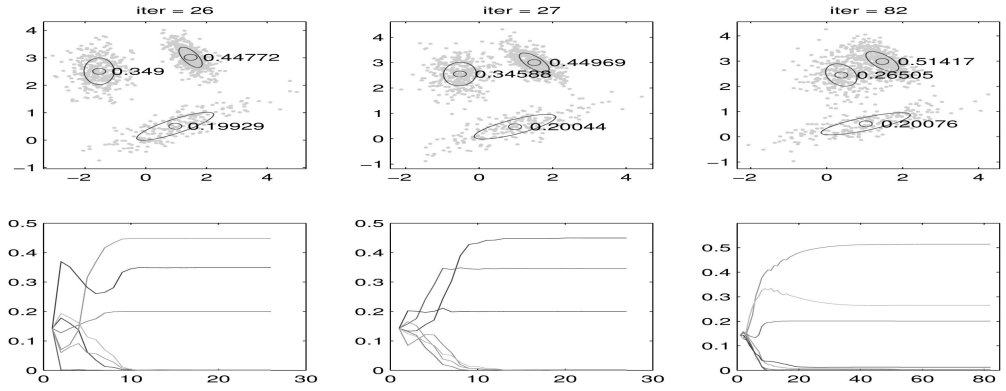

Fig. 6. The learning curves of α_js via the X-EM on three trials with the different overlapping levels.

TABLE 2
Performance of the Four Methods on Yeast Cell Cycle Microarray Data

Cell division phase	Methods	FP	FN	TP	TN
Early G1	Method I	11	24	43	306
(67 genes)	Method II	50	12	55	267
	Method III	21	21	46	296
	Method IV	38	10	57	279
Late G1	Method I	13	54	81	236
(135 genes)	Method II	28	40	95	221
	Method III	24	35	100	225
	Method IV	43	10	125	206
S	Method I	10	47	28	299
(75 genes)	Method II	33	49	26	276
	Method III	37	36	39	272
	Method IV	72	18	57	237
G2	Method I	13	22	30	319
(52 genes)	Method II	28	41	11	304
	Method III	18	29	23	314
	Method IV	46	5	47	286
M	Method I	12	26	29	317
(55 genes)	Method II	38	42	13	291
	Method III	19	8	47	310
	Method IV	47	2	53	283

TABLE 3
Comparison of the Total Error Rates of the Four Methods on Yeast Cell Cycle Microarray Data

Method	FP	FN	FP+FN
Method I	59	173	232
Method II	177	184	361
Method III	119	129	248
Method IV	246	45	291

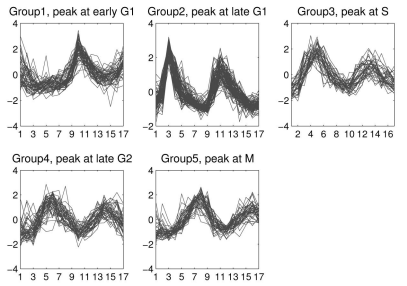

Fig. 7. The genes classified to the five classes by the X-EM algorithm.

ACKNOWLEDGMENTS

The authors would like to sincerely thank the editor and the anonymous reviewers for their valuable comments and insightful suggestions. They would also like to thank Mr. Xing-ming Zhao for conducting Experiment 5. The work described in this paper was supported by Faculty Research Grant of Hong Kong Baptist University with the Project Code: FRG/05-06/II-42, by the Research Grant Council of Hong Kong SAR under Project HKBU 2156/04E, and in part by the Special Funds for Major State Basic Research Projects (project G19990328) and NSFC (project 60372033).

REFERENCES

[1] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc.*, vol. 39, pp. 1-38, 1977.
[2] X.L. Meng and D.B. Rubin, "Maximum Likelihood Estimation via the ECM Algorithm: A General Framework," *Biometrika*, vol. 80, no. 2, pp. 267-278, 1993.
[3] X.L. Meng and D.A. van Dyk, "The EM Algorithm—An Old Folk Song Sung to a Fast New Tune," *J. Royal Statistical Soc. B*, vol. 59, pp. 511-567, 1997.
[4] N. Ueda, R. Nakano, Z. Ghahramani, and G.E. Hinton, "Smem Algorithm for Mixture Models," *Neural Computation*, vol. 12, pp. 2109-2128, 2000.
[5] B. Zhang, C. Zhang, and X. Yi, "Competitive EM Algorithm for Finite Mixture Models," *Pattern Recognition*, vol. 37, pp. 131-144, 2004.
[6] Y. Wu, Q. Tian, and T.S. Huang, "Discriminant-EM Algorithm with Application to Image Retrieval," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition*, pp. 222-227, 2000.
[7] G.J. McLachlan and K.E. Basford, *Mixture Models: Inference and Applications to Clustering*,. Marcel Dekker, 1988.
[8] F. Sparacino, "Sto(ry)chastics: A Bayesian Network Architecture for User Modeling and Computational Storytelling for Interactive Spaces," *Proc. Fifth Int'l Conf. Ubiquitous Computing*, pp. 54-72, 2003.
[9] V. Krishnamurthy and J.B. Moore, "On-Line Estimation of Hidden Markov Model Parameters Based on the Kullback-Leibler Information Measure," *IEEE Trans. Signal Processing*, vol. 41, pp. 2557-2573, Aug. 1993.
[10] I. Holmes and G.M. Rubin, "An Expectation Maximization Algorithm for Training Hidden Substitution Models," *J. Molecular Biology*, vol. 317, no. 5, pp. 753-764, 2002.
[11] R. Kass and A.E. Raftery, "Bayes Factors and Model Uncertainty," *J. Am. Statistical Assoc.*, vol. 90, pp. 773-795, 1995.
[12] H. Akaike, "Information Theory and an Extension of the Maximum Likelihood Principle," *Proc. Second Int'l Symp. Information Theory*, pp. 267-281, 1973.
[13] H. Akaike, "A New Look at the Statistical Model Identfication," *IEEE Trans. Automatic Control AC-19*, pp. 716-723, 1974.
[14] G. Schwarz, "Estimating the Dimension of a Model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461-464, 1978.
[15] H. Bozdogan, "Model Selection and Akaike's Information Criterion: The General Theory and Its Analytical Extensions," *Psychometrika*, vol. 52, no. 3, pp. 345-370, 1987.
[16] P.J. Green, "Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination," *Biometrika*, vol. 82, no. 4, pp. 711-732, 1995.
[17] S. Richardson and P.J. Green, "On Bayesian Analysis of Mixtures with an Unknown Number of Components (with Discussion)," *J. Royal Statistical Soc. Series B*, vol. 59, pp. 731-792, 1997.
[18] Y.M. Cheung, "A Rival Penalized EM Algorithm towards Maximizing Weighted Likelihood for Density Mixture Clustering with Automatic Model Selection," *Proc. Int'l Conf. Pattern Recognition*, vol. 4, pp. 633-636, 2004.
[19] Y.M. Cheung, "Maximum Weighted Likelihood via Rival Penalized EM for Density Mixture Clustering with Automatic Model Selection," *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 6, pp. 750-761, June 2005.
[20] L. Xu, "Bayesian Ying-Yang Machine, Clustering, and Number of Clusters," *Pattern Recognition Letters*, vol. 18, nos. 11-13, pp. 1167-1178, 1997.
[21] R.J. Cho, M.J. Campbell, E.A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T.G. Wolfsberg, A.E. Gabrielian, D. Landsman, D.J. Lockhart, and R.W. Davis, "A Genome-Wide Transcriptional Analysis of the Mitotic Cell Cycle," *Molecular Cell*, vol. 2, pp. 65-73, 1998.
[22] K.Y. Yeung, C. Fraley, A. Murua, A.E. Raftery, and W.L. Ruzzo, "Model-Based Clustering and Data Transformations for Gene Expression Data," *Bioinformatics*, vol. 17, pp. 977-987, 2001.
[23] Y. Qu and S. Xu, "Supervised Cluster Analysis for Microarray Data Based on Multivariate Gaussian Mixture," *Bioinformatics*, vol. 20, pp. 1905-1913, 2004.
[24] M.P.S. Brown, W.N. Grundy, D. Lin, N. Cristianini, C.W. Sugnet, T.S. Furey, M. Ares, and D. Haussler, "Knowledge-Based Analysis of Micro-Array Gene Expression Data by Using Support Vector Machines," *Proc. Nat'l Academy of Sciences of the USA*, vol. 97, pp. 262-267, 2000.