A Maximum Weighted Likelihood Approach to Simultaneous Model Selection and Feature Weighting in Gaussian Mixture

Yiu-ming Cheung and Hong Zeng

Department of Computer Science, Hong Kong Baptist University, Hong Kong
{ymc,hzeng}@comp.hkbu.edu.hk

Abstract. This paper is to identify the clustering structure and the relevant features automatically and simultaneously in the context of Gaussian mixture model. We perform this task by introducing two sets of weight functions under the recently proposed *Maximum Weighted Likelihood* (MWL) learning framework. One set is to reward the significance of each component in the mixture, and the other one is to discriminate the relevance of each feature to the cluster structure. The experiments on both the synthetic and real-world data show the efficacy of the proposed algorithm.

1 Introduction

The finite mixture model has provided a formal approach to address the clustering problems. In this unsupervised domain, there are two key issues. One is the determination of an appropriate number of components (also called *number of clusters* or *model order* interchangeably) in a mixture model. In general, the true clustering structure may not be well described with too few components, whereas the estimated model may "over-fit" the data if it uses too many components. The other issue is how to identify the relevance of observation features with respect to the clustering structure. From the practical viewpoint, some features may not be so important, or even be irrelevant, to the clustering structure. Their participation in the clustering process will prevent a clustering algorithm from finding an appropriate partition. Hence, it is necessary to discriminate the relevance of each feature with respect to the clustering structure in the clustering analysis.

Clearly, the above two issues are closely related. However, most of the existing approaches deal with these two issues separately or sequentially. Some methods typically choose the most influential features prior to a clustering algorithm, e.g., see [1,2]. Although the success of their algorithms has been demonstrated in their application domains, these pre-selected features may not be necessarily suitable to the clustering algorithm that will be ultimately employed. Moreover, some approaches [3,4] wrap the clustering algorithms in an outer layer to evaluate the candidate feature subsets. The performance of such a method is superior to that of the previous approaches, but their search strategies of generating the feature subset candidates are prone to find a local maxima. Recently, it has been believed that the above two issues should be jointly optimized in a single learning paradigm. A typical example is the work of [5], which introduces the concept of *feature saliency* to measure the relevance of each feature to the clustering structure, as the *feature weight*. It then heuristically integrates the *Minimum Message Length* (MML) criterion into the likelihood, and optimizes this penalized likelihood using a modified Expectation-Maximization (EM) algorithm to obtain the feature weights and the clustering results. Nevertheless, the penalty terms given by the MML criterion are static and fixed for all components at each EM iteration. As a result, they may not be robust enough under a certain environment, in which it is more desirable to implement a dynamic and embedded scheme to control the model complexity.

In this paper, we propose such an approach to Gaussian mixture clustering by formulating the above two issues into a single *Maximum Weighted Likelihood* (MWL) optimization function [6]. We introduce two sets of weight functions to address these two issues, respectively. Consequently, both the model selection and feature weighting are performed automatically and simultaneously. The experiments on both the synthetic and real-world data have shown the efficacy of the proposed algorithm.

2 The MWL Learning Framework

Suppose N i.i.d. observations, denoted as $\mathbf{x_1}$, $\mathbf{x_2}$, ..., $\mathbf{x_N}$, come from the following mixture model:

$$p(\mathbf{x}_t|\Theta^*) = \sum_{j=1}^{k^*} \alpha_j^* p(\mathbf{x}_t|\theta_j^*) \tag{1}$$

with

$$\sum_{j=1}^{k^*} \alpha_j^* = 1 \quad \text{and} \quad \forall 1 \leq j \leq k^*, \quad \alpha_j^* > 0,$$

where each observation $\mathbf{x_t}$ $(1 \leq t \leq N)$ is a column vector of d-dimensional features, i.e. $\mathbf{x_t} = [x_{1t}, \ldots, x_{dt}]^T$, and $\Theta^* = \{\alpha_j^*, \theta_j^*\}_{j=1}^{k^*}$. Furthermore, θ_j^* denotes the parameter set of the jth probability density function (pdf) $p(\mathbf{x}_t|\theta_j^*)$ in the mixture model, k^* is the true cluster number, and α_j^* is the mixing proportion of the jth component in the mixture. Θ^* is estimated from these N observations by:

$$\hat{\Theta}_{ML} = \arg\max_{\Theta}\{\log p(\mathbf{X_N}|\Theta)\}. \tag{2}$$

where $\mathbf{X_N} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}\}$, and $\hat{\Theta}_{ML} = \{\alpha_j, \theta_j\}_{j=1}^{k}$ is a maximum likelihood (ML) estimate of Θ^*.

When the number of components k is known, the ML estimate in (2) could be obtained by the Expectation-Maximization (EM) algorithm. However, from the practical viewpoint, it is difficult or even impossible to know the number of components in advance. Recently, a promising approach called *Rival Penalized Expectation-Maximization* (RPEM for short) [6], where the model order is determined automatically and simultaneously with the parameter estimation, has been developed under the MWL learning framework. The main idea of the MWL framework is to introduce unequal weights in general into the conventional maximum likelihood, which actually provides a new promising way for regularization so that the weighted likelihood does not increase monotonically over the candidate model complexity. Specifically, the weighted likelihood is given bellow:

$$\begin{aligned} Q(\Theta, \mathbf{X}_N) &= \frac{1}{N}\sum_{t=1}^{N} \log p(\mathbf{x}_t|\Theta) = \frac{1}{N\zeta}\sum_{t=1}^{N}\sum_{j=1}^{k} g(j|\mathbf{x}_t, \Theta)\log p(\mathbf{x}_t|\Theta) \\ &= \frac{1}{N\zeta}\sum_{t=1}^{N} \mathcal{M}(\Theta, \mathbf{x}_t) \end{aligned} \tag{3}$$

$$\mathcal{M}(\Theta, \mathbf{x}_t) = \sum_{j=1}^{k} g(j|\mathbf{x}_t, \Theta)\log[\alpha_j p(\mathbf{x}_t|\theta_j)] - \sum_{j=1}^{k} g(j|\mathbf{x}_t, \Theta)\log h(j|\mathbf{x}_t, \Theta) \tag{4}$$

where $h(j|\mathbf{x}_t, \Theta) = \frac{\alpha_j p(\mathbf{x}_t|\theta_j)}{p(\mathbf{x}_t|\Theta)}$ is the posterior probability that $\mathbf{x}_t$ belongs to the jth component in the mixture, k is an estimate of k^* with $k \geq k^*$, and ζ is a constant. The $g(j|\mathbf{x}_t, \Theta)$'s are the weight functions, satisfying the constraints:

$$\forall t, j, \ \sum_{j=1}^{k} g(j|\mathbf{x}_t, \Theta) = \zeta; \lim_{h(j|\mathbf{x}_t,\Theta)\to 0} g(j|\mathbf{x}_t, \Theta)\log h(j|\mathbf{x}_t, \Theta) = 0.$$

In the RPEM algorithm of [6], the weight functions are constructed as:

$$g(j|\mathbf{x}_t, \Theta) = (1 + \varepsilon_t)I(j|\mathbf{x}_t, \Theta) - \varepsilon_t h(j|\mathbf{x}_t, \Theta) \tag{5}$$

with

$$I(j|\mathbf{x}, \Theta) = \begin{cases} 1 \ if \ j = c \equiv \arg\max_{1\leq i\leq k} h(i|\mathbf{x}, \Theta); \\ 0 \ otherwise. \end{cases} \tag{6}$$

where ε_t is a small positive quantity. Under this weight construction, given an observation $\mathbf{x}_t$, a positive weight $g(c|\mathbf{x}_t, \Theta)$ is assigned to the log-likelihood of the winning component, i.e., the component with the maximum value of $h(j|\mathbf{x}_t, \Theta)$, so that it is updated to adapt to $\mathbf{x}_t$, meanwhile all rival components are penalized with a negative weight. This intrinsic rival penalization mechanism of the RPEM makes the genuine clusters survive, whereas the "pseudo-clusters" gradually vanish. The updating details of $\hat{\Theta}_{MWL} = \arg\max_{\Theta}\{Q(\Theta, \mathbf{X}_N)\}$ can be found in [6].

The numerical results have shown its outstanding performance on both of synthetic and real-life data [6], where all features are equally useful in clustering process. Nevertheless, analogous to the most of the existing clustering algorithms, the performance of the RPEM may deteriorate provided that there exist some irrelevant features in feature vectors. In the following, we will therefore perform the feature relevancy analysis and further extend the RPEM accordingly within the MWL framework.

3 Simultaneous Clustering and Feature Weighting

To discriminate the importance of each feature in the cluster structure, we utilize the concept of *feature saliency* defined in [5] as our feature weight ($i.e., w_l, 0 \leq w_l \leq 1, \forall 1 \leq l \leq d$): the lth feature is relevant with a probability w_l that the feature's pdf is dependent of the pdf's of components in the mixture. For those features whose values are distributed among all clusters, we regard its distribution as a common one. We suppose the features are independent of each other, then the pdf of a more general mixture model can be written below as in [5]:

$$\begin{aligned} p(\mathbf{x}_t|\Theta) &= \sum_{j=1}^{k} \alpha_j \prod_{l=1}^{d} p(x_{lt}|\Phi) \\ &= \sum_{j=1}^{k} \alpha_j \prod_{l=1}^{d} [w_l p(x_{lt}|\theta_{lj}) + (1 - w_l) q(x_{lt}|\lambda_l)] \end{aligned} \tag{7}$$

where $p(x_{lt}|\theta_{lj}) = \mathcal{N}(x_{l,t}|m_{lj}, s_{lj}^2)$ denotes a Gaussian density function of the relevant feature x_{lt} with the mean m_{lj}, and the variance s_{lj}^2; $q(x_{lt}|\lambda_l)$ is the common distribution of the irrelevant feature. In this paper, we shall limit it to be a Gaussian as well for a general purpose, i.e., $q(x_{lt}|\lambda_l) = \mathcal{N}(x_{lt}|cm_l, cs_l^2)$. Subsequently, the full parameter set of the general Gaussian mixture model is redefined as $\Theta = \{\{\alpha_j\}_{j=1}^k, \Phi\}$ and $\Phi = \{\{\theta_{lj}\}_{l=1,j=1}^{d,k}, \{w_l\}_{l=1}^d, \{\lambda_l\}_{l=1}^d\}$. Note that

$$p(x_{lt}|\Phi) = w_l p(x_{lt}|\theta_{lj}) + (1 - w_l) q(x_{lt}|\lambda_l) \tag{8}$$

is a linear mixture of two possible densities for each feature, and the feature weight w_l acts as a regulator to determine which distribution is more appropriate to describe the feature. A new perspective is to regard this form as a *lower level* Gaussian mixture, which resembles the *higher level* Gaussian mixture on which the weights of the genuine clusters are estimated. Hence, the feature weight w_l can be considered as the counterpart of component weight α_j. Subsequently, a similar rewarding and penalizing scheme can be embedded into the likelihood function of (3) for this *lower level* mixture. To this end, we re-write (3) as:

$$\tilde{Q}(\Theta, \mathbf{X}_N) = \frac{1}{N} \sum_{t=1}^{N} \log p(\mathbf{x}_t|\Theta) = \frac{1}{N\zeta} \sum_{t=1}^{N} \tilde{\mathcal{M}}(\Theta, \mathbf{x}_t). \tag{9}$$

To control the complexity of the model to be estimated, we introduce two sets of weight functions, i.e. $\tilde{g}(.|\mathbf{x}_t, \Theta)$ and $\tilde{f}(.|x_{lt}, \Phi)$, into the log-likelihood for the components in the *higher level* and *lower level* mixtures, respectively. Altogether, by inserting the following formulas:

$$p(\mathbf{x}_t|\Theta) = \frac{\alpha_j p(\mathbf{x}_t|\Phi)}{\tilde{h}(j|\mathbf{x}_t, \Theta)};\ p(x_{lt}|\Phi) = \frac{w_l p(x_{lt}|\theta_{lj})}{h'(1|x_{lt}, \Phi)} = \frac{(1 - w_l) q(x_{lt}|\lambda_l)}{h'(0|x_{lt}, \Phi)},$$

into (9), we obtain the weighted log-likelihood for the mixture model as follows:

$$\tilde{\mathcal{M}}(\Theta, \mathbf{x}_t) = \sum_{j=1}^{k} \tilde{g}(j|\mathbf{x}_t, \Theta) \log \alpha_j +$$
$$\sum_{j=1}^{k}\sum_{l=1}^{d} \tilde{g}(j|\mathbf{x}_t, \Theta)\Big\{\tilde{f}(1|x_{lt}, \Phi) \log[w_l p(x_{lt}|\theta_{lj})] + \tilde{f}(0|x_{lt}, \Phi) \log[(1-w_l)q(x_{lt}|\lambda_l)]\Big\} -$$
$$\sum_{j=1}^{k}\sum_{l=1}^{d} \tilde{g}(j|\mathbf{x}_t, \Theta)\tilde{f}(1|x_{lt}, \Phi) \log h^{'}(1|x_{lt}, \Phi) - \sum_{j=1}^{k}\sum_{l=1}^{d} \tilde{g}(j|\mathbf{x}_t, \Theta)\tilde{f}(0|x_{lt}, \Phi) \log h^{'}(0|x_{lt}, \Phi)$$
$$- \sum_{j=1}^{k} \tilde{g}(j|\mathbf{x}_t, \Theta) \log \tilde{h}(j|\mathbf{x}_t, \Theta) \tag{10}$$

where

$$\tilde{h}(j|\mathbf{x}_t, \Theta) = \frac{\alpha_j p(\mathbf{x}_t|\Phi)}{p(\mathbf{x}_t|\Theta)} = \frac{\alpha_j \prod_{l=1}^{d} [w_l p(x_{lt}|\theta_{lj}) + (1-w_l)q(x_{lt}|\lambda_l)]}{\sum_{i=1}^{k} \alpha_i \prod_{l=1}^{d} [w_l p(x_{lt}|\theta_{li}) + (1-w_l)q(x_{lt}|\lambda_l)]},$$

$$h^{'}(1|x_{lt}, \Phi) = \frac{w_l p(x_{lt}|\theta_{lj})}{w_l p(x_{lt}|\theta_{lj}) + (1-w_l)q(x_{lt}|\lambda_l)}, \; h^{'}(0|x_{lt}, \Phi) = 1 - h^{'}(1|x_{lt}, \Phi).$$

$\tilde{h}(j|\mathbf{x}_t, \Theta)$ indicates the probability that some features in the data points come from the jth density component in the subspace. $h^{'}(1|x_{lt}, \Phi)$ represents the posterior probability that the lth feature conforms to the mixture model. That is, it reflects the prediction for the relevance of the lth feature to the clustering structure.

We design the weight functions for the *higher level* mixture as follows:

$$\tilde{g}(j|\mathbf{x}_t, \Theta^{old}) = I(j|\mathbf{x}_t, \Theta) + \tilde{h}(j|\mathbf{x}_t, \Theta), j = 1, \ldots, k_{max}, \tag{11}$$

where the $I(j|\mathbf{x}_t, \Theta)$ is the indicator function defined in (6). It is clear that this form meets the requirements of weight functions under the MWL framework. The rationale behind this form is that we give an award to the winning component, i.e., the cth one, by assigning a weight whose value is larger than the corresponding $\tilde{h}(c|\mathbf{x}_t, \Theta)$. In contrast, we keep the weights of those rival components exactly equal to their corresponding $\tilde{h}(j|\mathbf{x}_t, \Theta)$'s. That is, we give the winning component an award, but the rival ones not. Hence, this is actually another kind of award-penalization scheme. Such a scheme is able to make the genuine components survive in the learning process, whereas those "pseudo-clusters" will be faded out from the mixture gradually.

In (10), the new weight functions $\{\tilde{f}(1|x_{lt}, \Phi), \tilde{f}(0|x_{lt}, \Phi)\}$ should satisfy the following constraint:

$$\lim_{h^{'}(i|x_{lt}, \Phi) \to 0} \tilde{f}(i|\mathbf{x}_t, \Theta) \log h^{'}(i|x_{lt}, \Phi) = 0, i \in \{0, 1\}.$$

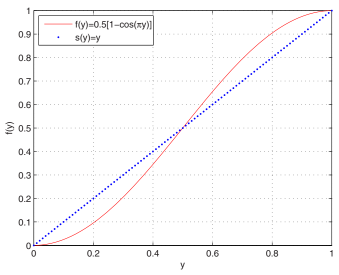

Fig. 1. f(y) vs. s(y). f(y) is plotted with "-"; s(y) is plotted with ".";

Accordingly, an applicable form is presented below:

$$\tilde{f}(1|x_{lt},\Phi) = f(h^{'}(1|x_{lt},\Phi)), \tilde{f}(0|x_{lt},\Phi) = 1 - f(h^{'}(1|x_{lt},\Phi)),$$

with

$$f(y) = 0.5[1 - \cos(\pi y)], y \in [0,1]. \tag{12}$$

$f(y)$ is plotted in Fig. 1. An interesting property of $f(y)$ can be observed from Fig. 1:

$$y > f(y) > 0,\ \ for\ 0 < y < 0.5;\ \ y < f(y) < 1,\ \ for\ 0.5 < y < 1.$$

Hence, if $h^{'}(1|x_{lt},\Phi) > h^{'}(0|x_{lt},\Phi)$, it implies that the feature seems useful in the clustering. Subsequently, its log-likelihood $\log[w_l p(x_{lt}|\theta_{lj})]$ is amplified by a larger coefficient $\tilde{f}(1|x_{lt},\Phi)$, meanwhile the log-likelihood of the "common" distribution is suppressed by a smaller coefficient $\tilde{f}(0|x_{lt},\Phi)$. A reverse assigning scheme is also held for the case when the feature seems less useful. In this sense, the function $f(y)$ rewards the important features that make the greater contribution to the likelihood, and penalizes those of insignificant features. Consequently, the estimation for the whole parameter set is given by:

$$\hat{\Theta}_{MWL} = \arg\max_{\Theta}\{\tilde{Q}(\Theta,\mathbf{X}_N)\}. \tag{13}$$

An EM-like iterative updating by gradient ascent technique is used to estimate the parameter set. Algorithm 1 shows the pseudo-code description of the proposed algorithm. In implementation of this algorithm, $\{\alpha_j\}_{j=1}^k$s must satisfy the constraint: $\sum_{j=1}^k \alpha_j = 1$ and $0 \leq \alpha_j < 1$. Hence, we also update $\{\beta_j\}_{j=1}^k$s instead of $\{\alpha_j\}_{j=1}^k$ as shown in [6], and $\{\alpha_j\}_{j=1}^k$ are obtained by: $\alpha_j = e^{\beta_j}/\sum_{i=1}^k e^{\beta_i}, 1 \leq j \leq k$. As for another parameter w_l with the constraint: $0 \leq w_l \leq 1$, we update γ_l instead of w_l, and w_l is obtained by the sigmoid function of γ_l: $w_l = \frac{1}{1+e^{-\tau\cdot\gamma_l}}, 1 \leq l \leq d$. The constant τ ($\tau > 0$) is used to tune the shape of the sigmoid function. To implement a moderate sensitivity for w_l, i.e., an approximately equal increasing or decreasing quantity in both γ_l and w_l, the slope of the sigmoid function around 0 with respect to (w.r.t.) γ_l (namely, around 0.5 w.r.t. w_l) should be roughly equal to 1. By rule of thumb,

Algorithm 1. The complete algorithm.

```
input  : $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, $k_{max}$, $\eta$, $epoch_{max}$, initial $\mathbf{\Theta}$
output: The converged $\hat{\mathbf{\Theta}}$
count ← 0;
while count ≤ $epoch_{max}$ do
    for $t \leftarrow 1$ to $N$ do
        step 1: Calculate $h^{'}$, $\tilde{h}$ to obtain $\tilde{f}$ and $\tilde{g}$;
        step 2:
            $\hat{\Theta}^{new} = \hat{\Theta}^{old} + \Delta\Theta = \hat{\Theta}^{old} + \eta \left.\frac{\partial \tilde{\mathcal{M}}(\mathbf{x}_t;\Theta)}{\partial\Theta}\right|_{\hat{\Theta}^{old}}$
    end
    count ← count + 1;
end
```

we find that a linear fitting of the two curves: $w_l = \frac{1}{1+e^{-\tau\cdot\gamma_l}}$ and $w_l = \gamma_l + 0.5$ around 0 w.r.t γ_l occurs around $\tau = 4.5$. In the following, we will therefore set τ at 4.5.

For each observation $\mathbf{x}_t$, the changes in the parameters set are calculated as:

$$\Delta\beta_j = \eta_\beta \left.\frac{\partial \tilde{\mathcal{M}}(\mathbf{x}_t;\Theta)}{\partial\beta_j}\right|_{\Theta^{old}} = \eta_\beta \sum_{i=1}^{k_{max}} \frac{\partial \tilde{\mathcal{M}}(\mathbf{x}_t;\Theta)}{\partial\alpha_i} \cdot \left.\frac{\partial\alpha_i}{\partial\beta_j}\right|_{\Theta^{old}}$$
$$= \eta_\beta(\tilde{g}(j|\mathbf{x}_t,\Theta) - \alpha_j^{old}), \tag{14}$$

$$\Delta m_{lj} = \eta \left.\frac{\partial \tilde{\mathcal{M}}(\mathbf{x}_t;\Theta)}{\partial m_{lj}}\right|_{\Theta^{old}}$$
$$= \eta\tilde{g}(j|\mathbf{x}_t,\Theta)\tilde{f}(1|x_{lt},\Phi)\frac{x_{lt} - m_{lj}^{old}}{(s_{lj}^{old})^2}, \tag{15}$$

$$\Delta s_{lj} = \eta \left.\frac{\partial \tilde{\mathcal{M}}(\mathbf{x}_t;\Theta)}{\partial s_{lj}}\right|_{\Theta^{old}}$$
$$= \eta\tilde{g}(j|\mathbf{x}_t,\Theta)\tilde{f}(1|x_{lt},\Phi)\left[\frac{(x_{lt} - m_{lj}^{old})^2}{(s_{lj}^{old})^3} - \frac{1}{s_{lj}^{old}}\right], \tag{16}$$

$$\Delta cm_l = \eta \left.\frac{\partial \tilde{\mathcal{M}}(\mathbf{x}_t;\Theta)}{\partial cm_l}\right|_{\Theta^{old}}$$
$$= \eta \sum_{j=1}^{k_{max}} \tilde{g}(j|\mathbf{x}_t,\Theta)\tilde{f}(0|x_{lt},\Phi)\frac{x_{lt} - cm_l^{old}}{(cs_l^{old})^2}, \tag{17}$$

$$\Delta cs_l = \eta \left.\frac{\partial \tilde{\mathcal{M}}(\mathbf{x}_t;\Theta)}{\partial cs_l}\right|_{\Theta^{old}}$$

$$= \eta \sum_{j=1}^{k_{max}} \tilde{g}(j|\mathbf{x}_t, \Theta)\tilde{f}(0|x_{lt}, \Phi)\left[\frac{(x_{lt} - cm_l^{old})^2}{(cs_l^{old})^3} - \frac{1}{cs_l^{old}}\right], \tag{18}$$

$$\Delta\gamma_l = \eta \left.\frac{\partial\tilde{\mathcal{M}}(\mathbf{x}_t;\Theta)}{\partial\gamma_l}\right|_{\Theta^{old}} = \eta \left.\frac{\partial\tilde{\mathcal{M}}(\mathbf{x}_t;\Theta)}{\partial w_l} \cdot \frac{\partial w_l}{\partial\gamma_l}\right|_{\Theta^{old}}$$

$$= \eta \cdot \tau \cdot \sum_{j=1}^{k_{max}} \tilde{g}(j|\mathbf{x}_t, \Theta)\left[\tilde{f}(1|x_{lt}, \Phi)(1 - w_l^{old}) - \tilde{f}(0|x_{lt}, \Phi)w_l^{old}\right]. \tag{19}$$

Generally speaking, the learning rate of β_js should be chosen as $\eta_\beta > \eta$ to help eliminate the much smaller α_j (we suggest $\eta = 0.1\eta_\beta$).

4 Experimental Results

4.1 Synthetic Data

We generated $1,000$ 2-dimensional data points from a Gaussian mixture of three components:

$$0.3*\mathcal{N}\left[\begin{pmatrix}1.0\\1.0\end{pmatrix}, \begin{pmatrix}0.15 & 0\\0 & 0.15\end{pmatrix}\right] + 0.4*\mathcal{N}\left[\begin{pmatrix}1.0\\2.5\end{pmatrix}, \begin{pmatrix}0.15 & 0\\0 & 0.15\end{pmatrix}\right] + 0.3*\mathcal{N}\left[\begin{pmatrix}2.5\\2.5\end{pmatrix}, \begin{pmatrix}0.15 & 0\\0 & 0.15\end{pmatrix}\right].$$

In order to illustrate the ability of the proposed algorithm to perform automatic model selection and feature weighting jointly, we appended two additional features to the original set to yield a 4-dimensional one. The last two features are sampled independently from $\mathcal{N}(2; 2.5^2)$ as the Gaussian noise covering the entire data set. Apparently, the last two dimensions do not hold the same mixture structure, thus are not as significant as the first two dimensions in the partitioning process.

We initialized k_{max} to 15, and all β_j's and γ_l's to 0, which is equivalent to setting each α_j to $1/15$ and w_l to 0.5, the remaining parameters were randomly initialized. The learning rates are $\eta = 10^{-5}$, $\eta_\beta = 10^{-4}$. After 500 epochs, the mixing coefficients and feature weights are obtained by the proposed algorithm:

$$\hat{\alpha}_6 = 0.4251 \; \hat{\alpha}_8 = 0.2893 \; \hat{\alpha}_{15} = 0.2856 \; \hat{\alpha}_j = 0, j \neq 6, 8, 15$$
$$\hat{w}_1 = 0.9968 \; \hat{w}_2 = 0.9964 \; \hat{w}_3 = 0.0033 \; \hat{w}_4 = 0.0036.$$

The feature weights of the first two dimensions converge close to 1, while those of the last two dimensions are assigned close to 0. It can be seen that the algorithm has accurately detected the underlying cluster structures in the first two dimensions, and meanwhile the appropriate model order and component parameters have been well estimated.

4.2 Real-World Data

We further investigated the performance of our proposed algorithm on serval benchmark databases [7] for data mining. The partitional accuracy of the algorithm without the prior knowledge of the underground class labels and the

Table 1. Results of the 30-fold runs on the test sets for each algorithm, where each data set has N data points with d features from k^* classes

Data Set	Method	Model Order $Mean \pm Std$	Error Rate $Mean \pm Std$
wine	RPEM	2.5 ± 0.7	0.0843 ± 0.0261
$d = 13$	EMFW	3.3 ± 1.4	0.0673 ± 0.0286
$N = 178$	NFW	2.8 ± 0.7	0.0955 ± 0.0186
$k^* = 3$	proposed method	3.3 ± 0.4	$\mathbf{0.0292 \pm 0.0145}$
heart	RPEM	1.7 ± 0.1	0.3167 ± 0.0526
$d = 13$	EMFW	2.5 ± 0.5	0.2958 ± 0.0936
$N = 270$	NFW	2.2 ± 0.4	0.2162 ± 0.0473
$k^* = 2$	proposed method	**fixed at 2**	$\mathbf{0.2042 \pm 0.0379}$
wdbc	RPEM	1.7 ± 0.4	0.2610 ± 0.0781
$d = 30$	EMFW	6.0 ± 0.7	0.0939 ± 0.0349
$N = 569$	NFW	3.0 ± 0.8	0.4871 ± 0.2312
$k^* = 2$	proposed method	2.6 ± 0.6	$\mathbf{0.0834 \pm 0.0386}$
ionosphere	RPEM	1.8 ± 0.5	0.4056 ± 0.0121
$d = 34$	EMFW	3.2 ± 0.6	0.1968 ± 0.0386
$N = 351$	NFW	2.2 ± 0.5	0.3201 ± 0.0375
$k^* = 2$	proposed method	2.9 ± 0.7	0.2029 ± 0.0667

relevancy of each features were measured by the *error rate* index. We randomly split those raw data sets into equal size for the training sets and the testing sets. The process was repeated 30 times, yielding 30 pairs of different training and test sets. For comparison, we conducted the proposed algorithm, the RPEM algorithm, as well as the approach in [5] (denoted as EMFW). To examine the efficacy of the feature weight function $f(.|x_{lt}, \Phi)$, we also conducted the algorithm (denoted as NFW) with the feature weight function in (12) setting to $s(y) = y, y \in [0, 1]$, i.e., no penalization on the *lower level* Gaussian mixture in feature relevancy estimation. The means and standard deviations of the results obtained on the four sets are summarized in Table 1, from which we have the three remarks as follows:

Table 2. The average weighting results of the 30 fold-runs on the real data, where the feature weights for *wdbc* and *ionosphere* are not included as the number of their features is too large to accommodate in this Table

Data set	Feature												
	1	2	3	4	5	6	7	8	9	10	11	12	13
wine	0.9990	0.8799	0.0256	0.2831	0.2354	0.9990	0.9990	0.0010	0.9900	0.9869	0.7613	0.9990	0.0451
heart	0.0010	0.6776	0.5872	0.0010	0.0010	0.9332	0.5059	0.0010	0.8405	0.3880	0.3437	0.4998	0.7856

Remark 1: In Table 1, it is noted that the proposed method has much lower error rates than the RPEM algorithm that is unable to perform the feature discrimination. Table 2 lists the mean feature weights of the sets obtained by the proposed algorithm in the 30 runs, indicating that only several features have good discriminating power. *Remark 2:* Without the feature weight functions to assign unequal emphases on the likelihood for the *lower level* Gaussian mixture, it is found that the performance of the NFW algorithm is unstable because of no enough penalization to the model complexity. It therefore validates the design of the proposed weight functions for weighting features. *Remark 3:* It is observed

that the method in [5] tends to use more components for the mixture, whereas the proposed algorithm not only gives general lower mismatch degrees, but also produces much more parsimonious models.

5 Conclusion

In this paper, a novel approach to tackle the two challenges for Gaussian mixture clustering, has been proposed under the Maximum Weighted Likelihood learning framework. The model order for the mixture model and the feature weighting are obtained simultaneously and automatically. Experimental results have shown the promising performance of the proposed algorithm on both the synthetic and the real-world data sets.

References

1. Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. IEEE Transactions on Knowledge and Data Engineering 17, 491–502 (2005)
2. Dash, M.C., Scheuermann, K., Liu, P.H.: Feature selection for clustering-A filter solution. In: Proceedings of 2nd IEEE International Conference on Data Mining, pp. 115–122. IEEE Computer Society Press, Los Alamitos (2002)
3. Dy, J., Brodley, C.: Feature selection for unsupervised learning. Joural of Machine Learning Research 5, 845–889 (2004)
4. Figueiredo, M.A.T., Jain, A.K., Law, M.H.C.: A feature selection wrapper for mixtures. LNCS, vol. 1642, pp. 229–237. Springer, Heidelberg (2003)
5. Law, M.H.C., Figueiredo, M.A.T., Jain, A.K.: Simultaneous feature selection and clustering using mixture models. IEEE Transactions on Pattern Analysis and Machine Intelligence 26, 1154–1166 (2004)
6. Cheung, Y.M.: Maximum weighted likelihood via rival penalized EM for density mixture clustering with automatic model selection. IEEE Transactions on Knowledge and Data Engineering 17, 750–761 (2005)
7. Blake, C.L., Merz, C.J.: UCI Repository of machine learning databases (1998), `http://www.ics.uci.edu/mlearn/MLRepository.html`