A Rival Penalized EM Algorithm towards Maximizing Weighted Likelihood for Density Mixture Clustering with Automatic Model Selection*

Yiu-ming Cheung
Department of Computer Science
Hong Kong Baptist University, Hong Kong, China

Abstract

How to determine the number of clusters is an intractable problem in clustering analysis. In this paper, we propose a new learning paradigm named Maximum Weighted Likelihood (MwL), in which the weights are designable. Accordingly, we develop a novel Rival Penalized Expectation-Maximization (RPEM) algorithm, whose intrinsic rival penalization mechanism enables the redundant densities in the mixture to be gradually faded out during the learning. Hence, the RPEM can automatically select an appropriate number of densities in density mixture clustering. The experiments have shown the promising results.

1. Introduction

Clustering analysis has been widely applied in a variety of scientific areas such as vector quantization [4], data mining [3], image processing [5], and so forth. In the literature, a broad view of clustering problem has been formulated within the framework of density estimates [7, 6], in which the probability density of inputs is represented by a finite mixture model. Each mixture component represents the density distribution of a cluster of data. Consequently, clustering can therefore be viewed as identifying the dense regions of the input densities. In the past, the Expectation-Maximization (EM) algorithm [2] has provided a general solution for the parameter estimate in a density mixture model. Unfortunately, it needs to pre-assign a correct number of densities. Otherwise, the EM will almost always lead to a poor estimate result.

In this paper, we will propose a new learning paradigm named Maximum Weighted Likelihood (MwL), under which a novel Rival Penalized EM (RPEM) algorithm is developed accordingly. The RPEM has the intrinsic rival penalization mechanism that enables the algorithm to gradually fade out the redundant densities of a mixture during the learning process. In other words, the RPEM has the capability of automatically selecting an appropriate number of densities in density mixture clustering. The experiments have demonstrated its outstanding performance on Gaussian mixture clustering in comparison with the EM.

*This work was supported by the Faculty Research Grant of Hong Kong Baptist University with the Project Code: FRG/02-03/II-40.

2 Maximum Weighted Likelihood (MwL): A New Learning Paradigm

Suppose N i.i.d. observations: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ are from a mixture of k^* densities, written as $p(\mathbf{x}|\mathbf{\Theta}^*)$, where $\mathbf{\Theta}^*$ denotes the true model parameter set. The ML estimate of $\mathbf{\Theta}^*$ can be obtained via maximizing the following cost function

$$\ell(\mathbf{x};\mathbf{\Theta}) = \int \ln p(\mathbf{x}|\mathbf{\Theta}) dF(\mathbf{x}), \tag{1}$$

with

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

where $F(\mathbf{x}) = \int_{-\infty}^{x} p(\mathbf{x})d\mathbf{x}$ is the cumulative probability function of $\mathbf{x}$. Hereinafter, we suppose that k is no less than k^*, and $p(\mathbf{x}|\mathbf{\Theta})$ is an identifiable density with respect to $\mathbf{\Theta}$. It can be seen that Eq.(1) can be further represented as

$$\begin{aligned} \ell(\mathbf{x};\mathbf{\Theta}) &= \int \ln p(\mathbf{x}|\mathbf{\Theta}) dF(\mathbf{x}) \\ &= \int \sum_{j=1}^{k} g(j|\mathbf{x},\mathbf{\Theta}) \ln p(\mathbf{x}|\mathbf{\Theta}) dF(\mathbf{x}) \end{aligned} \tag{3}$$

where $g(j|\mathbf{x},\mathbf{\Theta})$s are the designable weights satisfying

$$\sum_{j=1}^{k} g(j|\mathbf{x},\mathbf{\Theta}) = 1. \tag{4}$$

In general, each of them is a deterministic function with respect to $\mathbf{x}$ and $\mathbf{\Theta}$. We name Eq.(3) *Weighted Likelihood*

function. By Baye's formula, we know that the posterior probability that $\mathbf{x}$ comes from the j^{th} density as given $\mathbf{x}$ is

$$h(j|\mathbf{x},\Theta) = \frac{\alpha_j p(\mathbf{x}|\boldsymbol{\theta}_j)}{p(\mathbf{x}|\Theta)}. \quad (5)$$

Subsequently, for any $1 \le j \le k$, we then have

$$p(\mathbf{x}|\Theta) = \frac{\alpha_j p(\mathbf{x}|\boldsymbol{\theta}_j)}{h(j|\mathbf{x},\Theta)} \quad (6)$$

as long as $h(j|\mathbf{x},\Theta)$ is not equal to zero. Putting Eq.(6) into Eq.(3), we therefore have

$$\begin{aligned}\ell(\mathbf{x};\Theta) &= \int [g(1|\mathbf{x},\Theta)\ln p(\mathbf{x}|\Theta) \quad (7)\\ &\quad + \ldots, + g(k|\mathbf{x},\Theta)\ln p(\mathbf{x}|\Theta)]dF(\mathbf{x})\\ &= \int \sum_{j=1}^{k} g(j|\mathbf{x},\Theta)\ln\frac{\alpha_j p(\mathbf{x}|\boldsymbol{\theta}_j)}{h(j|\mathbf{x},\Theta)}dF(\mathbf{x})\\ &= \int \sum_{j=1}^{k} g(j|\mathbf{x},\Theta)\ln[\alpha_j p(\mathbf{x}|\boldsymbol{\theta}_j)]dF(\mathbf{x})\\ &\quad - \int \sum_{j=1}^{k} g(j|\mathbf{x},\Theta)\ln h(j|\mathbf{x},\Theta)dF(\mathbf{x}).\end{aligned}$$

As N is large enough, the MwL cost function of Eq.(7) can be further approximated by:

$$\begin{aligned}Q(\mathbf{X}_N;\Theta) &= \frac{1}{N}\sum_{t=1}^{N}\sum_{j=1}^{k} g(j|\mathbf{x}_t,\Theta)\ln[\alpha_j p(\mathbf{x}_t|\boldsymbol{\theta}_j)]\\ &\quad \frac{1}{N}\sum_{t=1}^{N}\sum_{j=1}^{k} g(j|\mathbf{x}_t,\Theta)\ln h(j|\mathbf{x}_t,\Theta), \quad (8)\end{aligned}$$

in which the first term is a generalized version of EM cost function, and no longer adheres to the mean value to estimates the hidden label as given the corresponding input. Evidently, it degenerates to the latter when $g(j|\mathbf{x}_t,\Theta)$ is equal to $h(j|\mathbf{x},\Theta)$ for any j. The second term is actually a kind of measures to represent the uncertainty of the densities that the input $\mathbf{x}_t$ comes from. For instance, as $g(j|\mathbf{x},\Theta) = h(j|\mathbf{x},\Theta)$, the second term is exactly the conditional entropy of the densities. In general, the learning of Θ towards maximizing the first term of Eq.(8) is to reduce such an uncertainty, but the learning of maximizing Eq.(8) will also increase the value of second term. In other words, the second term is serving as a regularization term in the learning of Θ. In Eq.(7) and Eq.(8), we do not consider the case that $h(j|\mathbf{x},\Theta) = 0$ for some j. Clearly, if $h(j|\mathbf{x},\Theta) = 0$ holds for some j, the maximum function value of Eq.(8) may not exist. To avoid this awkward situation, we therefore further request

$$\forall\, j,\ g(j|\mathbf{x},\Theta) = 0 \text{ if and only if } h(j|\mathbf{x},\Theta) = 0 \quad (9)$$

in designing $g(j|\mathbf{x},\Theta)$, which has a variety of choices as long as the conditions stated in Eq.(4) and Eq.(9) are satisfied. For instance, we can let $g(j|\mathbf{x},\Theta)$ be some probability function, i.e.,$\sum_{j=1}^{k} g(j|\mathbf{x},\Theta) = 1$ and $g(j|\mathbf{x},\Theta) \ge 0$ for any $1 \le j \le k$. A typical example is to let $g(j|\mathbf{x},\Theta) = h(j|\mathbf{x},\Theta)$, or

$$I(j|\mathbf{x}_t,\Theta) = \begin{cases} 1, & \text{if } j = c = \arg\max_{1\le r\le k} h(j|\mathbf{x}_t,\Theta)\\ 0, & \text{otherwise.}\end{cases} \quad (10)$$

In the former, Eq.(8) degenerates to the Kullback-Leibler divergence function derived from Ying-Yang Machine with the backward architecture, e.g., see [8]. In contrast, the latter design leads Eq.(8) to be the cost function of hard-cut EM [8]. In the subsequent sections, we will prefer to investigate one specific $g(j|\mathbf{x}_t,\Theta)$ only with

$$g(j|\mathbf{x}_t,\Theta) = 2\varphi(j|\mathbf{x}_t,\Theta) - h(j|\mathbf{x}_t,\Theta), \quad (11)$$

where $\varphi(j|\mathbf{x}_t,\Theta)$ is a special probability function named *indicator function*, i.e., given any input $\mathbf{x}_t$, we have $\sum_{j=1}^{k}\varphi(j|\mathbf{x}_t,\Theta) = 1$, $\phi(j|\mathbf{x}_t,\Theta) \ge 0$ for $1 \le j \le k$, and there is one and only one, denoted as $\varphi(c|\mathbf{x}_t,\Theta)$, equal to 1. After designing the weights, the learning of Θ can then be accomplished towards maximizing Eq.(8). We therefore name such a learning as *Maximum Weighted Likelihood* (MwL) learning approach.

3 Rival Penalized EM Algorithm

By considering the specific weights in Eq.(11) and putting Eq.(11) into Eq.(8), the cost function of Eq.(8) then becomes

$$Q(\mathbf{X}_N;\Theta) = \frac{1}{N}\sum_{t=1}^{N} q_t(\mathbf{x}_t;\Theta) \quad (12)$$

with

$$\begin{aligned}&q_t(\mathbf{x}_t;\Theta) = \quad (13)\\ &\sum_{j=1}^{k}[2\varphi(j|\mathbf{x}_t,\Theta) - h(j|\mathbf{x}_t,\Theta)]\ln[\alpha_j p(\mathbf{x}_t|\boldsymbol{\theta}_j)]\\ &- \sum_{j=1}^{k}[2\varphi(j|\mathbf{x}_t,\Theta) - h(j|\mathbf{x}_t,\Theta)]\ln h(j|\mathbf{x}_t,\Theta),\end{aligned}$$

where $q_t(\mathbf{x}_t;\Theta)$ is called an instantaneous cost function at time step t because its value depends on Θ and the current input $\mathbf{x}_t$ only. Before estimating Θ via maximizing $Q(\mathbf{X}_N;\Theta)$ in Eq.(12), we need to specify $\varphi(j|\mathbf{x}_t,\Theta)$. One choice is simply let

$$\varphi(j|\mathbf{x}_t,\Theta) = I(j|\mathbf{x}_t,\Theta), \text{for } 1 \le j \le k \quad (14)$$

as given in Eq.(10). It should be note that, if the number of maximum values of $h(j|\mathbf{x},\Theta)$s is more than one, we

can randomly select one index among them as c and let $\varphi(c|\mathbf{x}_t, \mathbf{\Theta}) = 1$, meanwhile the others are equal to zero. Subsequently, we can always guarantee $\varphi(j|\mathbf{x}_t, \mathbf{\Theta})$ to be an indicator function. As a result, we can learn $\mathbf{\Theta}$ via maximizing Eq.(12) adaptively. That is, after assigning some initial value to $\mathbf{\Theta}$, we perform the following two steps as given an input $\mathbf{x}_t$:

Step 1 Fixing $\mathbf{\Theta}^{(\text{old})}$, we compute $h(j|\mathbf{x}, \mathbf{\Theta}^{(\text{old})})$ and $\varphi(j|\mathbf{x}_t, \mathbf{\Theta}^{(\text{old})})$ via Eq.(5) and Eq.(14), respectively.

Step 2 Fixing $h(j|\mathbf{x}_t, \mathbf{\Theta})$s calculated in **Step 1**, we update $\mathbf{\Theta}$ with a small step towards the direction of maximizing Eq.(13). To avoid the constraint on α_js during the optimization, we therefore let α_js be the soft-max function of k new free variables β_js with

$$\alpha_j = \frac{\exp(\beta_j)}{\sum_{r=1}^{k} \exp(\beta_r)}, \qquad \text{for} \quad 1 \le j \le k, \quad (15)$$

and update β_js directly instead of α_js. As a result, we update $\mathbf{\Theta}$ by

$$\begin{aligned}
\beta_c^{(\text{new})} &= \beta_c^{(\text{old})} + \eta \frac{\partial q_t(\mathbf{x}_t; \mathbf{\Theta})}{\partial \beta_c}|_{\mathbf{\Theta}^{(\text{old})}} \\
&= \beta_c^{(\text{old})} + \eta[2 - h(c|\mathbf{x}_t, \mathbf{\Theta}^{(\text{old})}) - \alpha_c^{(\text{old})}] \\
\boldsymbol{\theta}_c^{(\text{new})} &= \boldsymbol{\theta}_c^{(\text{old})} + \eta \frac{\partial q_t(\mathbf{x}_t; \mathbf{\Theta})}{\partial \boldsymbol{\theta}_c}|_{\mathbf{\Theta}^{(\text{old})}} \\
&= \boldsymbol{\theta}_c^{(\text{old})} + \eta[2 - h(c|\mathbf{x}_t, \mathbf{\Theta}^{(\text{old})})] \frac{\partial \ln p(\mathbf{x}_t|\boldsymbol{\theta}_c)}{\partial \boldsymbol{\theta}_c},
\end{aligned}$$

meanwhile

$$\begin{aligned}
\beta_r^{(\text{new})} &= \beta_r^{(\text{old})} + \eta \frac{\partial q_t(\mathbf{x}_t; \mathbf{\Theta})}{\partial \beta_r}|_{\mathbf{\Theta}^{(\text{old})}} \\
&= \beta_r^{(\text{old})} - \eta[h(r|\mathbf{x}_t, \mathbf{\Theta}^{(\text{old})}) + \alpha_r^{(\text{old})}] \\
\boldsymbol{\theta}_r^{(\text{new})} &= \boldsymbol{\theta}_r^{(\text{old})} + \eta \frac{\partial q_t(\mathbf{x}_t; \mathbf{\Theta})}{\partial \boldsymbol{\theta}_r}|_{\mathbf{\Theta}^{(\text{old})}} \\
&= \boldsymbol{\theta}_r^{(\text{old})} - \eta h(r|\mathbf{x}_t, \mathbf{\Theta}^{(\text{old})}) \frac{\partial \ln p(\mathbf{x}_t|\boldsymbol{\theta}_r)}{\partial \boldsymbol{\theta}_r},
\end{aligned}$$

where η is a small positive learning rate, $\alpha_j^{(\text{old})}$ is computed via Eq.(15) in terms of $\beta_j^{(\text{old})}$, c is given in Eq.(10), and $r = 1, 2, ..., k$ but $r \neq c$.

The above two steps are iteratively implemented for each input until $\mathbf{\Theta}$ converges. It can be seen that, at each time step t, **Step 2** not only updates the associated parameters of the winning mixture component to adapt to the input, but all those of rival components are also penalized towards minimizing the value of $p(\mathbf{x}_t|\mathbf{\Theta})$ with the force strength proportional to $h(r|\mathbf{x}_t, \mathbf{\Theta})$s, respectively. The larger the $h(r|\mathbf{x}_t, \mathbf{\Theta})$ is, the stronger the penalized force is. We therefore name this algorithm *Rival Penalized EM* (RPEM), whose intrinsic rival-penalized mechanism, as shown in the next section, enables the RPEM to gradually fade the redundant components out in a density mixture. In the following, we will further study RPEM in more details under the Gaussian density mixtures.

Suppose each mixture component $p(\mathbf{x}_t|\boldsymbol{\theta}_j)$ of Eq.(8) is Gaussian, denoted as $G(\mathbf{x}_t|\mathbf{m}_j, \mathbf{\Sigma}_j)$, where $\mathbf{m}_j$ and $\mathbf{\Sigma}_j$ are the means and covariance matrices, respectively. As a result, the details of the previous **Step 1** and **Step 2** can be given as follows:

Step 1 Given an input $\mathbf{x}_t$, we fix $\mathbf{\Theta}^{(\text{old})}$, and calculate

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

where $1 \le j \le k$, and α_js are calculated by Eq.(15). Furthermore, $\varphi(j|\mathbf{x}_t, \mathbf{\Theta}^{(\text{old})}) = I(j|\mathbf{x}_t, \mathbf{\Theta}^{(\text{old})})$ as given by Eq.(10).

Step 2 Fixing $h(j|\mathbf{x}_t, \mathbf{\Theta}^{(\text{old})})$s, we update $\mathbf{\Theta}$. We have noticed that all subsequent computations involve $\mathbf{\Sigma}_j^{-1}$s only rather than $\mathbf{\Sigma}_j$s. To save computing costs and ensure the learning of $\mathbf{\Sigma}_j$ stable, we therefore directly update $\mathbf{\Sigma}_j^{-1}$s rather than $\mathbf{\Sigma}_j$s. Consequently, the details of updating $\mathbf{\Theta}$ are given as follows:

$$\begin{aligned}
\beta_j^{(\text{new})} &= \beta_j^{(\text{old})} + \eta[g(j|\mathbf{x}_t, \mathbf{\Theta}^{(\text{old})}) - \alpha_j^{(\text{old})}] \\
\mathbf{m}_j^{(\text{new})} &= \mathbf{m}_j^{(\text{old})} + \eta g(j|\mathbf{x}_t, \mathbf{\Theta}^{(\text{old})}) \mathbf{\Sigma}_j^{-1(\text{old})} \\
&\quad (\mathbf{x}_t - \mathbf{m}_j^{(\text{old})}) \\
\mathbf{\Sigma}_j^{-1(\text{new})} &= [1 + \eta g(j|\mathbf{x}_t, \mathbf{\Theta}^{(\text{old})})] \mathbf{\Sigma}_j^{-1(\text{old})} \\
&\quad -\eta g(j|\mathbf{x}_t, \mathbf{\Theta}^{(\text{old})}) \mathbf{U}_{t,j} \qquad (16)
\end{aligned}$$

with

$$\begin{aligned}
\mathbf{U}_{t,j} &= [\mathbf{\Sigma}_j^{-1(\text{old})} (\mathbf{x}_t - \mathbf{m}_j^{(\text{old})})(\mathbf{x}_t - \mathbf{m}_j^{(\text{old})})^T \\
&\quad \mathbf{\Sigma}_j^{-1(\text{old})}], \quad 1 \le j \le k, \qquad (17)
\end{aligned}$$

where $g(j|\mathbf{x}_t, \mathbf{\Theta}^{(\text{old})})$ is given by Eq.(11).

Please note that, to simplify the computation of $\mathbf{\Sigma}_j^{-1}$s' update, Eq.(16) has updated $\mathbf{\Sigma}_j^{-1}$ along the direction of $\mathbf{\Sigma}_j^{-1} \frac{\partial q_t(\mathbf{x}_t; \mathbf{\Theta})}{\partial \mathbf{\Sigma}_j^{-1}} \mathbf{\Sigma}_j^{-1}$, i.e, along the direction with an acute angle of $\frac{\partial q_t(\mathbf{x}_t; \mathbf{\Theta})}{\partial \mathbf{\Sigma}_j^{-1}}$.

In the above algorithm, if we ignore the difference between $h(c|\mathbf{x}_t, \mathbf{\Theta})$ and $I(c|\mathbf{x}_t)$, the $g(j|\mathbf{x}_t, \mathbf{\Theta})$ in Eq.(11) then becomes

$$g(j|\mathbf{x}_t, \mathbf{\Theta}) \approx \begin{cases} 1, & \text{if } j = c, \\ -h(j|\mathbf{x}_t, \mathbf{\Theta}), & \text{otherwise.} \end{cases} \qquad (18)$$

Eventually, the RPEM algorithm will become a generalized version of the Rival Penalization Controlled Competitive Learning (RPCCL)[1]. Also, it will include the existing RPCL [9] and its Type A variant [8] as its special cases, but meanwhile providing a theoretical guidance to choose their awkward de-learning rate. We will go into the details elsewhere because of the space limitation.

4 Simulation Results

To demonstrate the performance of RPEM, we generated $1,000$ synthetic data points from a mixture of three bivariate Gaussian density distributions with the true mixture proportions $\alpha_1^* = 0.3$, $\alpha_2^* = 0.4$, and $\alpha_3^* = 0.3$. Furthermore, we set $\eta = 0.001$, and randomly assigned 7 seed points in the input space as shown in Fig. 1(a). After 250 epoches, Fig. 1(b) shows the stable positions of 7 seed points learned by RPEM, where three of them are located at the corresponding cluster centers, while the other four stay at the outside of the clusters. Furthermore, a snapshot of α_js is:

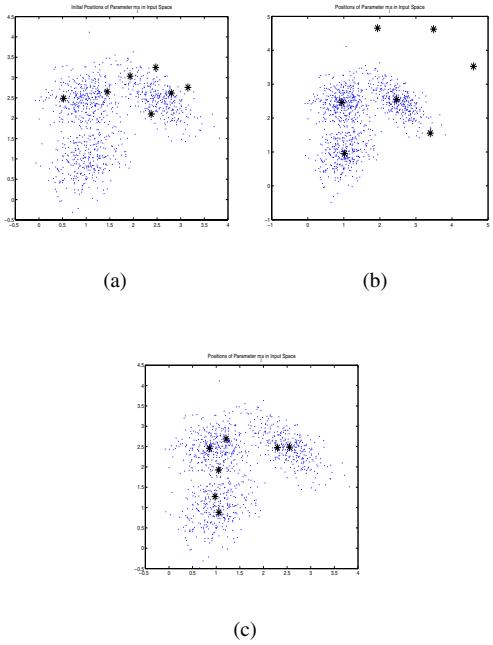
$$\begin{aligned} \alpha_1 &= 0.023, \quad \alpha_2 = 0.342, \quad \alpha_3 = 0.287, \quad \alpha_4 = 0.293, \\ \alpha_5 &= 0.019, \quad \alpha_6 = 0.018, \quad \alpha_7 = 0.018, \end{aligned}$$

we found that α_1, α_5, α_6 and α_7 are learned towards zero. In other words, the input data set is recognized from the mixture of the three densities: $2, 3, 4$. Hence, the RPEM has the robust performance without knowing the true mixture number.

For comparison, we also demonstrated the EM performance under the same experimental environment. Fig. 1(c) shows the final positions of 7 seed points in the input space, where they are all biased from the cluster centers. Also, none of α_js was approached to zero through the learning. Instead, the EM led 7 densities to compete each other without making extra densities die. That is, the EM cannot work at all in this case.

5 Conclusion

We have proposed a new MwL learning paradigm, under which the RPEM algorithm has been developed accordingly. Compared to the EM, the RPEM has the intrinsic rival penalization mechanism, which enables the algorithm to automatically select an appropriate number of densities by gradually fading the redundant densities out in a density mixture. The experiments have shown the outstanding performance of RPEM in Gaussian mixture clustering.



(a) (b)

(c)

Figure 1. The positions of 7 seed points marked by '*' in the input data space: (a) the initial random positions, (b) the final position obtained via the RPEM, (c) the final position obtained via the EM.

References

[1] Y. M. Cheung. Rival penalization controlled competitive learning for data clustering with unknown cluster number. In *Proceedings of 9th International Conference on Neural Information Processing (Paper ID:* 1983 *in CD-ROM Proceeding)*, November 18-22, 2002.

[2] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of Royal Statistical Society*, 39:1–38, 1977.

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

[4] B. Fritzke. The lbg-u method for vector quantization – an improvement over lbg inspired from neural networks. *Neural Processing Letters*, 5(1):35–45, 1997.

[5] Y. Lim and S. Lee. On the color image segmentation algorithm based on the thresholding and the fuzzy c-means techniques. *Pattern Recognition*, 23(9):935–952, 1990.

[6] G. McLachlan and K. Basford. *Mixture Models: Inference and Application to Clustering*. Dekker, 1988.

[7] B. Silverman. *Density Estimation for Statistics and Data Analysis*. London: Chapman & Hall, 1986.

[8] L. Xu. Bayesian ying-yang machine, clustering and number of clusters. *Pattern Recognition Letters*, 18(11-13):1167–1178, 1997.

[9] L. Xu, A. Krzyzak, and E. Oja. Rival penalized competitive learning for clustering analysis, rbf net, and curve detection. *IEEE Transaction on Neural Networks*, 4:636–648, 1993.