A Batch Rival Penalized EM Algorithm for Gaussian Mixture Clustering with Automatic Model Selection

Dan Zhang[1] and Yiu-ming Cheung[2]

[1] Faculty of Mathematics and Computer Science
HuBei University, WuHan, China
mathzhang52@yahoo.com.cn
[2] Department of Computer Science
Hong Kong Baptist University, Hong Kong SAR, China
ymc@comp.hkbu.edu.hk

Abstract. Cheung [2] has recently proposed a general learning framework, namely Maximum Weighted Likelihood (MWL), in which an adaptive Rival Penalized EM (RPEM) algorithm has been successfully developed for density mixture clustering with automatic model selection. Nevertheless, its convergence speed relies on the value of learning rate. In general, selection of an appropriate learning rate is a nontrivial task. To circumvent such a selection, this paper further studies the MWL learning framework, and develops a batch RPEM algorithm accordingly provided that all observations are available before the learning process. Compared to the adaptive RPEM algorithm, this batch RPEM need not assign the learning rate analogous to the EM, but still preserve the capability of automatic model selection. Further, the convergence speed of this batch RPEM is faster than the EM and the adaptive RPEM. The experiments show the efficacy of the proposed algorithm.

Keywords: Maximum Weighted Likelihood, Rival Penalized Expectation-Maximization Algorithm, Learning Rate.

1 Introduction

As a typical statistical technique, clustering analysis has been widely applied to a variety of scientific areas such as data mining, vector quantization, image processing, and so forth. In general, one kind of clustering analysis can be formulated as a density mixture clustering problem, in which each mixture component represents the probability density distribution of a corresponding 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.

In general, the Expectation-Maximum (EM) algorithm [3] provides an efficient way to estimate the parameters in a density mixture model. Nevertheless, it needs to pre-assign a correct number of clusters. Otherwise, it will almost always lead to a poor estimate result. Unfortunately, from the practical viewpoint,

J.T. Yao et al. (Eds.): RSKT 2007, LNAI 4481, pp. 252–259, 2007.

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 pre-assigning the exact number of clusters. Such algorithms include the RPCL algorithm [4] and its improved version, namely RPCCL[1]. More recently, Cheung [2] 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. Subsequently, this intrinsic rival penalization mechanism enables the RPEM to automatically select an appropriate number of densities by fading out the redundant densities from a density mixture. The numerical results have shown its outstanding performance on both of synthetic and real-life data. Furthermore, a simplified version of RPEM has included RPCL and RPCCL as its special cases with some new extensions.

In the papers [2], the RPEM algorithm learns the parameters via a stochastic gradient ascending method, i.e., 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 diverge. 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 changed over time, e.g. see [5].

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 preserve the capability of automatic model selection. Further, the convergence speed of this batch RPEM is faster than the EM and the adaptive RPEM. The experiments have shown the efficacy of the proposed algorithm.

2 Overview of Maximum Weighted Likelihood (MWL) Learning Framework

Suppose an input $\mathbf{x}$ comes from the following density mixture model:

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

where $\boldsymbol{\Theta}$ 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 j^{th} component, and $p(\mathbf{x}|\boldsymbol{\theta}_j)$ is

a multivariate probability density function (pdf) of $\mathbf{x}$ parameterized by $\boldsymbol{\theta}_j$. In the MWL learning framework, the parameter set $\boldsymbol{\Theta}$ is learned via maximizing the following Weighted Likelihood (WL) cost function:

$$l(\boldsymbol{\theta}) = \int \sum_{j=1}^{k} g(j|\mathbf{x}, \boldsymbol{\Theta}) \ln[\alpha_j p(\mathbf{x}|\boldsymbol{\theta}_j)] dF(\mathbf{x}) - \int \sum_{j=1}^{k} g(j|\mathbf{x}, \boldsymbol{\Theta}) \ln h(j|\mathbf{x}, \boldsymbol{\Theta}) dF(\mathbf{x}), \tag{2}$$

with

$$h(j|\mathbf{x}, \boldsymbol{\Theta}) = \frac{\alpha_j p(\mathbf{x}|\boldsymbol{\theta}_j)}{P(\mathbf{x}|\boldsymbol{\Theta})} \tag{3}$$

to be the posterior probability of $\mathbf{x}$ coming from the j^{th} density as given $\mathbf{x}$, where $g(j|\mathbf{x}, \boldsymbol{\Theta})$s are the designable weights satisfying the two conditions:

(Condition 1) $\sum_{j=1}^{k} g(j|\mathbf{x}, \boldsymbol{\Theta}) = 1$, and
(Condition 2) $\forall j, g(j|\mathbf{x}, \boldsymbol{\Theta}) = 0$ if $h(j|\mathbf{x}, \boldsymbol{\Theta}) = 0$.

Suppose a set of N observations, denoted as $\chi = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, comes from the density mixture model in Eq.(1), the empirical WL function of Eq.(2), written as $\Upsilon(\boldsymbol{\Theta}; \chi)$, can be further expanded as:

$$\Upsilon(\boldsymbol{\Theta}; \chi) = \frac{1}{N} \sum_{t=1}^{N} \sum_{j=1}^{k} g(j|\mathbf{x}_t, \boldsymbol{\Theta}) \ln[\alpha_j p(\mathbf{x}_t|\boldsymbol{\theta}_j)] - \frac{1}{N} \sum_{t=1}^{N} \sum_{j=1}^{k} g(j|\mathbf{x}_t, \boldsymbol{\Theta}) \ln h(j|\mathbf{x}_t, \boldsymbol{\Theta}). \tag{4}$$

In [2], the weights $g(j|\mathbf{x}_t, \boldsymbol{\Theta})$ have been generally designed as:

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

where ε_t is a coefficient varying with the time step t. Please note that $g(j|\mathbf{x}_t, \boldsymbol{\Theta})$ in Eq.(5) can be negative as well as positive. For simplicity, we hereinafter set ε_t as a constant, denoted as ε. Furthermore, $I(j|x_t, \boldsymbol{\Theta})$ is an indicator function with

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

Subsequently, the earlier work [2] has presented the adaptive RPEM to learn $\boldsymbol{\Theta}$ via maximizing the WL function of Eq.(4) using a stochastic gradient ascent method. Interested readers may refer to the paper [2] for more details. In the following, we just summarize the main steps of the adaptive RPEM as follows:

Step 1. Given the current input $\mathbf{x}_t$ and the parameter estimate, written as $\boldsymbol{\Theta}^{(n)}$, we compute $h(j|\mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$ and $g(j|\mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$ via Eq.(3) and Eq.(5), respectively.

Step 2. Given $h(j|\mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$ and $g(j|\mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$, we update $\boldsymbol{\Theta}$ by

$$\boldsymbol{\Theta}^{(n+1)} = \boldsymbol{\Theta}^{(n)} + \eta \frac{q_t(\boldsymbol{\Theta}; \mathbf{x}_t)}{\boldsymbol{\Theta}}|_{\boldsymbol{\Theta}^{(n)}}, \tag{7}$$

with

$$q_t(\boldsymbol{\Theta}; \mathbf{x}_t) = \sum_{j=1}^{k} g(j|\mathbf{x}_t, \boldsymbol{\Theta}) \ln[\alpha_j p(\mathbf{x}_t|\boldsymbol{\theta}_j)], \tag{8}$$

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.

The experiments have shown the superior performance of the adaptive RPEM, in particular the capability of automatic model selection. Nevertheless, the convergence speed of this adaptive algorithm 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

3.1 Algorithm

As shown in Step 2 of Section 2, we actually update the parameters via maximizing the first term of Eq.(4), whereas the second term is just a conditional entropy of the densities and can be regarded as the constant when updating the parameters. In the following, we denote the first term of Eq.(4) as:

$$\zeta(\boldsymbol{\Theta}; \chi) = \frac{1}{N} \sum_{t=1}^{N} \sum_{j=1}^{k} g(j|\mathbf{x}_t, \boldsymbol{\Theta}) \ln[\alpha_j p(\mathbf{x}_t|\boldsymbol{\theta}_j)]. \tag{9}$$

Hence, we need to solve the following nonlinear optimization problem:

$$\widetilde{\boldsymbol{\Theta}} = \arg\max_{\boldsymbol{\Theta}} \{\zeta(\boldsymbol{\Theta}; \chi)\} \tag{10}$$

subject to the constraints as shown in Eq.(1). To solve this optimal problem with equality constraint, we adopt Lagrange method analogous to the EM by introducing a Lagrange multiplier λ into the Lagrange function. Subsequently, we have:

$$F(\boldsymbol{\Theta}, \lambda) = \zeta(\boldsymbol{\Theta}; \chi) + \lambda(\sum_{j=1}^{k} \alpha_j - 1) \tag{11}$$

with $\boldsymbol{\Theta} = (\alpha_j, \boldsymbol{\theta}_j)_{j=1}^{k}$.

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

$$p(j|\mathbf{x}_t, \boldsymbol{\theta}_j) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} exp[-\frac{1}{2}(\mathbf{x}_t - \mathbf{m}_j)^T \boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_t - \mathbf{m}_j)], \tag{12}$$

where $\boldsymbol{\theta}_j = (\mathbf{m}_j, \boldsymbol{\Sigma}_j)$, $\mathbf{m}_j$ and $\boldsymbol{\Sigma}_j$ are the mean (also called *seed points* interchangeably) and the covariance of j^{th} density, respectively.

Through optimizing Eq.(11), we then finally obtain the batch RPEM algorithm as follows:

Step 1. Given $\boldsymbol{\Theta}^{(n)}$, we compute $h(j|\mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$s and $g(j|\mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$s for all $\mathbf{x}_t$s via Eq.(3) and Eq.(5), respectively.

Step 2. Given $h(j|\mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$s and $g(j|\mathbf{x}_t, \boldsymbol{\Theta}^{(n)})$s computed in Step 1, we update $\boldsymbol{\Theta}$ by

$$\alpha_j^{(n+1)} = \aleph_j^{(\mathrm{n})} / \sum_{j=1}^{k} \aleph_j^{(\mathrm{n})}, \quad \mathbf{m}_j^{(n+1)} = \frac{1}{\aleph_j^{(\mathrm{n})}} \sum_{t=1}^{N} \mathbf{x}_t g(j|\mathbf{x}_t, \boldsymbol{\theta}^{(\mathrm{n})})$$

$$\boldsymbol{\Sigma}_j^{(n+1)} = \frac{1}{\aleph_j^{(\mathrm{n})}} \sum_{t=1}^{N} g(j|\mathbf{x}_t, \boldsymbol{\theta}^{(\mathrm{n})})(\mathbf{x}_t - \mathbf{m}_j^{(\mathrm{n})})(\mathbf{x}_t - \mathbf{m}_j^{(\mathrm{n})})^T, \qquad (13)$$

where $\aleph_j^{(\mathrm{n})} = \sum_{t=1}^{N} g(j|\mathbf{x}_t, \boldsymbol{\theta}^{(\mathrm{n})})$. If the covariance matrix $\boldsymbol{\Sigma}_j^{(n+1)}$ is singular, it indicates that the corresponding j^{th} 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 $\alpha_j^{(\mathrm{n}+1)}$s so that their sum is always kept to be 1.

Step 3. Let $n = n + 1$, and go to Step 1 for the next iteration until $\boldsymbol{\Theta}$ is converged.

In the above batch RPEM, we need to assign a value to ε in the weight design as shown in Eq.(5). A new question is how to assign an appropriate value of ε? The next sub-section will answer this question.

3.2 How to Assign Parameter ε?

We rewrite Eq.(5) as the following form:

$$g(j|\mathbf{x}_t, \boldsymbol{\Theta}) = \begin{cases} h(j|\mathbf{x}_t, \boldsymbol{\Theta}) + (1+\varepsilon)(1 - h(j|\mathbf{x}_t, \boldsymbol{\Theta})), & \text{if } j = c \\ h(j|\mathbf{x}_t, \boldsymbol{\Theta}) - (1+\varepsilon)h(j|\mathbf{x}_t, \boldsymbol{\Theta}), & \text{otherwise,} \end{cases} \qquad (14)$$

where the term $(1+\varepsilon)(1 - h(j|\mathbf{x}_t, \boldsymbol{\Theta}))$ is the award of the winning density component (i.e. the c^{th} density with $I(c|\mathbf{x}_t, \boldsymbol{\Theta}) = 1$), and meanwhile the term $-(1+\varepsilon)h(j|\mathbf{x}_t, \boldsymbol{\Theta})$ is the penalty of the rival components (i.e., those densities with $I(j|\mathbf{x}_t, \boldsymbol{\Theta}) = 0$). Intuitively, it is expected that the award is positive and the penalty is negative, i.e., ε should be greater than -1. Otherwise, as $\varepsilon < -1$, we will meet an awkward situation: 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 becomes zero. In this special case, the batch RPEM is actually degenerated into the EM without the property of automatic model selection.

In addition, it is noticed that the larger the ε, the stronger the award and penalty are. This property makes the algorithm converge faster with a larger value of ε, but the algorithm is more prone to a sub-optimal solution. Our empirical studies have shown that the covariance matrix of a rival density is prone to singular if ε is too large. Hence, an appropriate selection of ε in the batch

RPEM would be a negative value. Further, our empirical studies have found that the algorithm has a poor capability of automatic model selection if ε is close to zero. As we can see, the smaller the $|\varepsilon|$, the smaller difference among the rival densities is. For example, if we set $|\varepsilon| = 0$, we get $g(j|\mathbf{x}_t, \boldsymbol{\Theta}) = I(j|\mathbf{x}_t, \boldsymbol{\Theta})$. Subsequently, the batch RPEM degenerates to the hard-cut EM without the capability of automatic model selection. Hence, by a rule of thumb, an appropriate selection of ε should be within the range of $(-1, -0.4)$. In the next section, we will arbitrarily set ε at -0.5.

4 Experimental Results

Because of the space limitation, we will conduct two experiments only to demonstrate the performance of the batch RPEM. In these two experiments, we used the same $1,000$ observations that were generated from a mixture of three bivariate Gaussian distributions, whose true parameters were:

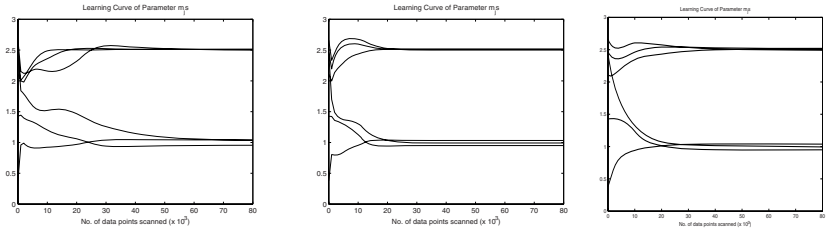
$$\begin{aligned}
&\alpha_1^* = 0.3, \quad \alpha_2^* = 0.4, \quad \alpha_3^* = 0.3 \\
&\mathbf{m}_1^* = [1.0, 1.0]^T, \quad \mathbf{m}_2^* = [1.0, 2.5]^T, \quad \mathbf{m}_3^* = [2.5, 2.5]^T \\
&\boldsymbol{\Sigma}_1^* = \begin{pmatrix} 0.20, 0.05 \\ 0.05, 0.30 \end{pmatrix}, \quad \boldsymbol{\Sigma}_2^* = \begin{pmatrix} 0.2\ 0.0 \\ 0.0\ 0.20 \end{pmatrix}, \quad \boldsymbol{\Sigma}_3^* = \begin{pmatrix} 0.2 & -0.1 \\ -0.1 & 0.2 \end{pmatrix}. \quad (15)
\end{aligned}$$

4.1 Experiment 1

This experiment was to investigate the convergence speed of the batch RPEM algorithm. We set $k = 3$, and the three seed points were randomly allocated in the observation space. Furthermore, all α_js and $\boldsymbol{\Sigma}_j$s were initialized at $\frac{1}{k}$ and the identity matrix, respectively. For comparison, we also implemented the EM under the same experimental environment. After all parameters were converged, both of the batch RPEM and EM gave the correct parameter estimates. Nevertheless, as shown in Fig. 1(a) and (b), the batch RPEM converges at 25 epochs while the EM needs 60 epochs. That is, the convergence speed of the former is significantly faster than the latter. This indicates that the intrinsic rival-penalization scheme of the batch RPEM, analogous to the RPCL [4], RPCCL [1] and the adaptive RPEM [2], 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. Furthermore, we also compared it with the adaptive RPEM, in which we set the learning rate $\eta = 0.001$. Fig. 1(c) shows that the adaptive RPEM converges at 40 epochs, slower than the proposed batch version.

4.2 Experiment 2

This experiment was to investigate the capability of the batch RPEM on model selection. We set $k = 10$, and randomly allocated the 10 seed points, $\mathbf{m}_1$, $\mathbf{m}_2$, ..., $\mathbf{m}_{10}$ into the observation space as shown in Fig. 2(a). During the learning process, we discarded those densities whose covariance matrices $\boldsymbol{\Sigma}_j$s were singular. After

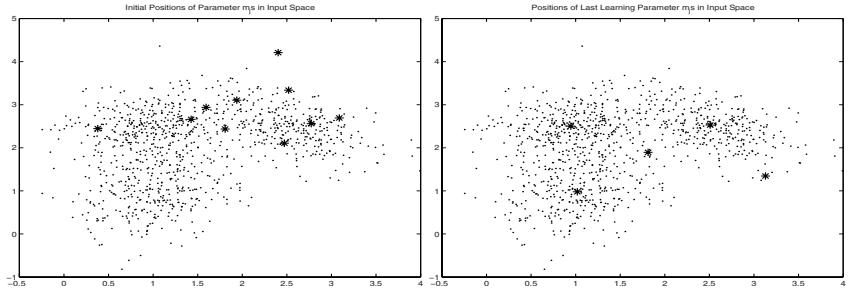


(a) (b) (c)

Fig. 1. Learning curves of $\mathbf{m}_j$s by (a) EM, (b) Batch RPEM, and (c) Adaptive RPEM, respectively

90 epochs, we found that 5 out of 10 density components have been discarded. The mixture proportions of the remaining components were converged to $\alpha_1 = 0.0061$, $\alpha_2 = 0.3508$, $\alpha_3 = 0.3222$, $\alpha_4 = 0.3161$, $\alpha_5 = 0.0048$. Furthermore, the corresponding $\mathbf{m}_j$s and $\boldsymbol{\Sigma}_j$s were:

$$\begin{aligned}
&\mathbf{m}_1 = [3.1292, 1.3460]^T, \mathbf{m}_2 = [0.9462, 2.5030]^T, \mathbf{m}_3 = [1.0190, 0.9788]^T, \\
&\mathbf{m}_4 = [2.5089, 2.5286]^T, \mathbf{m}_5 = [1.8122, 1.8955]^T \\
&\boldsymbol{\Sigma}_1 = \begin{pmatrix} 0.1920, 0.0310 \\ 0.0310, 0.0088 \end{pmatrix}, \boldsymbol{\Sigma}_2 = \begin{pmatrix} 0.1708, 0.0170 \\ 0.0170, 0.1489 \end{pmatrix}, \boldsymbol{\Sigma}_3 = \begin{pmatrix} 0.1894, 0.0461 \\ 0.0461, 0.2892 \end{pmatrix}, \\
&\boldsymbol{\Sigma}_4 = \begin{pmatrix} 0.2155, -0.1101 \\ -0.1101, \quad 0.2116 \end{pmatrix}, \boldsymbol{\Sigma}_5 = \begin{pmatrix} 0.0027, -0.0053 \\ -0.0053, \quad 0.0213 \end{pmatrix}. \qquad (16)
\end{aligned}$$



(a) The initial seed positions marked by '*', where seed points are those data points to be learned towards the cluster centers in the observation space.

(b) The converged seed positions learned by the batch RPEM algorithm

Fig. 2. The Results of the Batch RPEM Algorithm

It can be seen that the three seed points $\mathbf{m}_2, \mathbf{m}_3, \mathbf{m}_4$ together with the corresponding proportions $\alpha_2, \alpha_3, \alpha_4$ have provided a good estimate of the true parameters as shown in Fig. 2(b), where $\mathbf{m}_2$, $\mathbf{m}_3$, $\mathbf{m}_4$ are stabled at the center of the clusters. In contrast, $\mathbf{m}_1$ and $\mathbf{m}_5$ have been driven away and located at the boundary of the clusters with a very small mixture proportions: $\alpha_1 = 0.0061$ and $\alpha_5 = 0.0048$. Actually, the 1^{st} and 5^{th} density components have been gradually faded out from the mixture.

5 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. The numerical results have shown the efficacy of the proposed algorithm.

Acknowledgment

This work was fully supported by the grants from the Research Grant Council of the Hong Kong SAR with the Project Codes: HKBU 2156/04E and HKBU 210306.

References

1. Cheung Y.M.: Rival Penalized Controlled Competitive Learning for Data Clustering with Unknown Cluster Number. Proceedings of Ninth International Conference on Neural Information Processing (ICONIP'02), Paper ID: 1983 in CD-ROM Proceeding (2002)
2. 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, Vol. 17, No. 6 (2005) 750–761
3. Dempster A. Laird N. and Rubin D.: Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of Royal Statistical Society, Vol. 39, No. 1 (1977) 1–38
4. Xu L., Krzyżak A. and Oja E.: Rival Penalized Competive Learning for Clustering Analysis, RBF Net, and Curve Detection. IEEE Transactions on Neural Networks, Vol. 4, (1993) 636–648
5. Zhao X.M., Cheung Y.M., Chen L. and Aihara K.: A New Technique for Adjusting the Learning Rate of RPEM Algorithm Automatically. to appear in the Proceedings of The Twelfth International Symposium on Artificial Life and Robotics, Japan (2007)