A New Technique for Adjusting the Learning Rate of RPEM Algorithm Automatically

Xing-Ming Zhao[1,2], Yiu-ming Cheung[3], Luonan Chen[1,2,4], Kazuyuki Aihara[1,2]
1. Aihara Complexity Modelling Project, ERATO, JST, Japan
2. Institute of Industrial Science, The University of Tokyo, Japan
3. Department of Computer Science, Hong Kong Baptist University, Hong Kong, China
4. Department of Electrical Engineering and Electronics, Osaka Sangyo University, Japan

Abstract

Recently, a new Rival Penalized Expectation Maximization (RPEM) algorithm has been proposed for estimating the parameters of the normal mixture model, meanwhile determining the number of classes automatically. The RPEM is an adaptive algorithm utilizing a small constant learning rate. To speed up its convergence speed, this paper proposes a new method to dynamically adjust the learning rate of the RPEM algorithm on line. The numerical results have shown the promising results of the proposed algorithm.

1 Introduction

Mixture models have been widely used in data mining [1], image processing [2], gene expression analysis [3], and so forth. In the literature, the Expectation-Maximization (EM) algorithm [4] has been widely used to estimate the parameters of the normal mixture model, which, however, needs to pre-assign class number. Generally, the EM algorithm almost always leads to a poor result if the class number is not appropriately pre-assigned. Recently, a new Rival Penalized Expectation-Maximization (RPEM) algorithm was proposed by Cheung [2] [5]. The RPEM algorithm can determine the number of classes automatically by gradually fading out the redundant components from the mixture during the parameter learning process. In [2] [5], the RPEM utilizes a small constant learning rate to ensure the algorithm's convergence, which, however, needs more iteration steps. Indeed, we can dynamically adjust the learning rates to speed up the performance convergence of the RPEM. In this paper, we present a new method accordingly for such a task. Hereinafter, we denote the RPEM algorithm with dynamic adjustment of learning rate as RPEM-DLR algorithm.

2 The RPEM-DLR Algorithm

The mixture model assumes that each group of data is generated by an underlying probability distribution. Suppose the number of classes is k, and the number of samples is N. In the RPEM algorithm, the likelihood function for a mixture model can be defined in a weighted form, i.e.,

$$l(\mathbf{\Theta};\mathbf{x}) = \int \sum_{j=1}^{k} g(j|\mathbf{x},\mathbf{\Theta})\ln p(\mathbf{x}|\mathbf{\Theta})dF(\mathbf{x}), \qquad (1)$$

with

$$p(\mathbf{x}|\mathbf{\Theta}) = \sum_{j=1}^{k} \alpha_j p(\mathbf{x}|\theta_j)\ , \qquad (2)$$

$$\sum_{j=1}^{k} \alpha_j = 1, \forall\, 1 \le j \le k,\ \alpha_j > 0\ , \qquad (3)$$

and

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

where $\mathbf{\Theta} = \{\alpha_j, \theta_j\}_{j=1}^{k}$ is the set of model parameters, $F(\mathbf{x})$ is the cumulative probability function of $\mathbf{x}$, $p(\mathbf{x}|\theta_j)$ is a multivariate probability density function (pdf) of $\mathbf{x}$, α_j is the proportion that $\mathbf{x}$ comes from Class j, and gs are designable weights. The details can be found in [5].

The RPEM algorithm in normal mixture models can be summarized as follows:

- **Initialization:** Given a specific k ($k \ge k^*$, k^* is the true class number), we initialize $\mathbf{\Theta}$. Then, at each time step t, we implement the following two steps:

- **Step 1:** Fixing $\mathbf{\Theta}^{(old)}$, calculate $h(j|\mathbf{x}_t, \mathbf{\Theta}^{(old)})$, $g(j|\mathbf{x}_t, \mathbf{\Theta}^{(old)})$ and $\alpha_j^{(old)}$, where

$$\alpha_j^{(old)} = \frac{\exp(\beta_j^{(old)})}{\sum_{i=1}^{k} \exp(\beta_i^{(old)})}. \qquad (5)$$

- **Step 2:** Fixing $h(j|\mathbf{x}_t, \mathbf{\Theta}^{(old)})$s, update $\mathbf{\Theta}$

$$\beta_j^{(new)} = \beta_j^{(old)} + \eta_1[g(j|\mathbf{x}_t, \mathbf{\Theta}^{(old)}) - \alpha_j^{(old)}]\,, \qquad (6)$$

$$\mu_j^{(new)} = \mu_j^{(old)} + \eta_2 g(j|\mathbf{x}_t, \mathbf{\Theta}^{(old)})\mathbf{\Sigma}_j^{-1(old)}(\mathbf{x}_t - \mu_j^{(old)}), \qquad (7)$$

$$\begin{aligned}\mathbf{\Sigma}_j^{-1(new)} &= [1 + \eta_2 g(j|\mathbf{x}_t, \mathbf{\Theta}^{(old)})]\mathbf{\Sigma}_j^{-1(old)} \\ &\quad - \eta_2 g(j|\mathbf{x}_t, \mathbf{\Theta}^{(old)})\mathbf{U}_{t,j}\,,\end{aligned}$$

where $\mathbf{U}_{t,j} = [\mathbf{\Sigma}_j^{-1(old)}(\mathbf{x}_t - \mu_j^{(old)})(\mathbf{x}_t - \mu_j^{(old)})^T\mathbf{\Sigma}_j^{-1(old)}]$, η_1 and η_2 are small positive learning rates with $\eta_1 \ll \eta_2 \leq 1$. The procedure repeat until $\mathbf{\Theta}$ converges.

In RPEM algorithm, it can be seen that the learning rate is generally a fixed small positive constant. Actually, the choice of the learning rate can affect the convergence performance of the RPEM. In general, there is a tradeoff between the residue deviation and rate of convergence [6]. When using a fixed learning rate, it should be small enough for the algorithm to converge. The smaller the learning rate, the smaller the residue deviation, but the slower the convergence speed. It is usually difficult to determine an optimal learning rate in advance because it is problem dependent.

According to the condition for the asymptotic convergence provided by a standard theorem [7] from stochastic approximation theory, the learning rate should satisfies:

$$\lim_{it\to\infty} \eta(it) = 0, \text{ and } \sum_{it=1}^{\infty} \eta(it) = \infty, \qquad (8)$$

where it is the it-th epoch. Under the circumstances, we propose a new method here for adjusting the learning rate dynamically. The learning rate is defined as:

$$\eta(j, it) = \eta(j, it-1) * \frac{1-\alpha(j)}{1+\alpha(j)},$$
$$1 \leq j \leq k, \text{and } 1 \leq it < \infty,$$

where $\eta(j, it)$ is the learning rate for the j-th component in the it-th epoch, and α_j is the proportion that $\mathbf{x}$ comes from Class j. The initial learning rate η_0 is set at a fixed small positive constant. Therefore, the learning rate will be adjusted dynamically according to α_j in each epoch.

3 Experimental Results and Discussions

In this section, two sets of data were used to investigate the performance of the RPEM-DLR algorithm. First, we generated 1,000 synthetic data points from a mixture of three bivariate Gaussian densities:

$$\begin{aligned} p1(\mathbf{x}|\mathbf{\Theta}) &= 0.3G[\mathbf{x}|\begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}0.10 & 0.05\\0.05 & 0.20\end{pmatrix}] \\ &\quad +0.4G[\mathbf{x}|\begin{pmatrix}1.0\\5.0\end{pmatrix}, \begin{pmatrix}0.10 & 0.0\\0.0 & 0.10\end{pmatrix}] \\ &\quad +0.3G[\mathbf{x}|\begin{pmatrix}5.0\\5.0\end{pmatrix}, \begin{pmatrix}0.10 & -0.05\\-0.05 & 0.10\end{pmatrix}].\end{aligned}$$

Suppose the number of seed points was set $k = 10$ and $k = 20$, respectively. We initialized each of $\mathbf{\Sigma}_j$s to be an identity matrix, and all β_js to be zero. The initial learning rate η_0 was set at 0.01. With the same initial parameters, the performance of the RPEM-DLR algorithm was compared to that of the RPEM algorithm. The results obtained by RPEM-DLR and RPEM are shown in Figure 1, where the points marked by '+' are the learned cluster centers via RPEM-DLR and EM, respectively. As shown in Figure 1, both the RPEM algorithm and RPEM-DLR can locate the cluster centers correctly by pushing away the redundant seed points when the pre-assigned class number is larger than the true mixture number, i.e., $k = 3$.

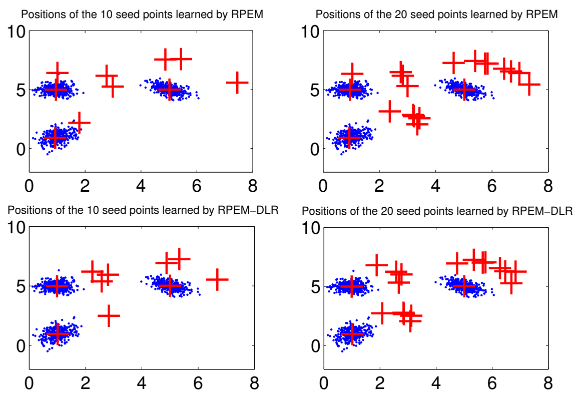

Figure 1: Convergent positions of the seed points learned via RPEM and RPEM-DLR for the data generated by $p1$.

Furthermore, we investigated the computation time taken by RPEM and RPEM-DLR algorithm during the learning procedure. The number of the seed points was set at 5, 10, 20 and 30, respectively. Table 1

Table 1: The comparison of computation time of RPEM and RPEM-DLR (seconds)

number of	methods	
seed points	RPEM	RPEM-DLR
5	101.5961	11.1861
10	239.0037	33.9889
20	402.2885	112.7221
30	1.0397e+003	225.6845

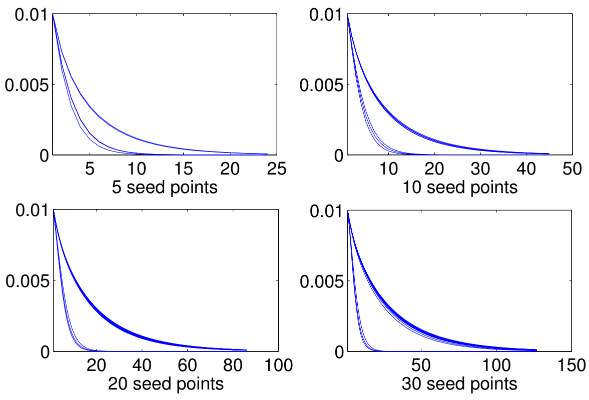

Figure 2: Learning curves of η_js via RPEM-DLR for the data generated by $p1$

shows the comparison of the computation time by RPEM and RPEM-DLR under the same conditions. As shown in Table 1, the computation time taken by RPEM-DLR is much less than that of the RPEM algorithm, i.e. our proposed RPEM-DLR algorithm largely reduces the calculation time. In a word, our proposed RPEM-DLR algorithm is really efficient, and speeds up the learning of the RPEM technique.

We further investigated the corresponding values of η_js learned via RPEM-DLR when the number of seed points was set at 5, 10, 20 and 30, respectively. As shown in Figure 2, the values of η_js corresponding to the extra seed points approached to zero slower than those of the true ones, which was reasonable because the learning rate was a monotonously dropping function of α_js. It can seen from Figure 2 that the learning rate was adjusted dynamically during the learning procedure, which speed up the learning of the RPEM.

Upon the data clusters well-separated above, we further investigated the performance of the RPEM-DLR algorithm on the data clusters that were considerably overlapped. We generated 1,000 synthetic data points from a mixture of three bivariate Gaussian densities:

$$\begin{aligned} p2(\mathbf{x}|\mathbf{\Theta}) &= 0.3G[\mathbf{x}|\begin{pmatrix}1\\1\end{pmatrix},\begin{pmatrix}0.15 & 0.05\\0.05 & 0.20\end{pmatrix}] \\ &\quad +0.4G[\mathbf{x}|\begin{pmatrix}1.0\\2.5\end{pmatrix},\begin{pmatrix}0.15 & 0.0\\0.0 & 0.15\end{pmatrix}] \\ &\quad +0.3G[\mathbf{x}|\begin{pmatrix}2.5\\2.5\end{pmatrix},\begin{pmatrix}0.15 & -0.1\\-0.1 & 0.15\end{pmatrix}] \end{aligned}$$

The number of seed points was set at 10 and 20, respectively. We initialized each of $\mathbf{\Sigma}_j$s to be an identity matrix, and all β_js to be zero. The initial learning rate η_0 was set at 0.01. Again, The performance of the RPEM-DLR algorithm was compared to that of the RPEM algorithm. The results obtained by RPEM-DLR and RPEM are shown in Figure 3. It can be seen from Figure 3 that both the RPEM and RPEM-DLR algorithm can stabilize at the corresponding cluster centers when the number of seed points is larger than the true mixture number.

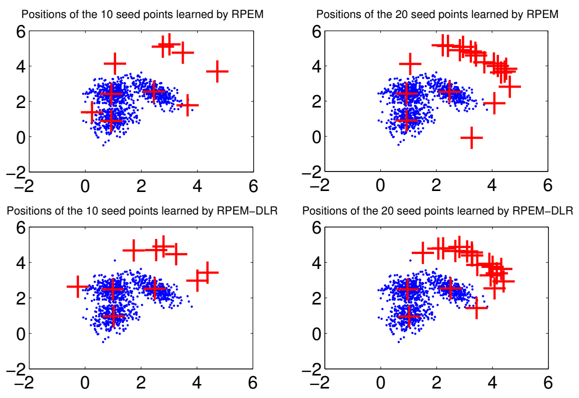

Figure 3: Convergent positions of the seed points learned via RPEM and RPEM-DLR for the data generated by $p2$.

The computation time taken by RPEM was compared to that of the RPEM-DLR algorithm during the learning procedure. The number of the seed points was set at 5, 10, 20 and 30, respectively. Table 2 shows the comparison of the computation time by RPEM and RPEM-DLR under the same conditions. As shown in Table 2, the computation time taken by RPEM-DLR is much less than that of the RPEM algorithm, which shows again that our proposed RPEM-DLR algorithm can adjust the learning rate dynamically and speed up the learning of RPEM.

Furthermore, Figure 4 shows the learning curves of η_js when the number of seed points was set at 5, 10, 20

Table 2: The comparison of computation time of RPEM and RPEM-DLR (seconds)

number of	methods	
seed points	RPEM	RPEM-DLR
5	124.5791	9.3735
10	152.4893	25.9974
20	486.4795	111.0196
30	1.0870e+003	266.6434

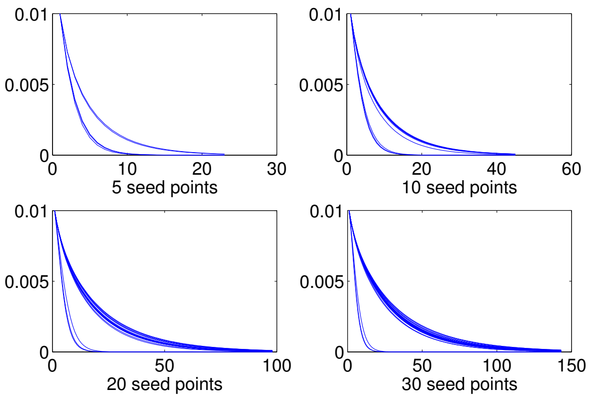

Figure 4: Learning curves of η_js via RPEM-DLR for the data generated by $p2$

and 30, respectively. As shown in Figure 4, the learning rates changed as we expected, and the RPEM-DLR technique can adjust the learning rate dynamically in the learning procedure. It can be concluded from the above experiments that the RPEM-DLR algorithm outperforms RPEM in terms of computation time, and adjusts its learning rate dynamically in the learning procedure.

4 Conclusions

This paper proposed a new method to dynamically adjust the learning rate of RPEM algorithm. Compared to the constant learning rate as used in the RPEM algorithm, our proposed method can efficiently speed up the performance convergence of RPEM algorithm. The numerical results have demonstrated its efficacy.

Acknowledgment

This work was partially supported by the Research Grant Council of Hong Kong SAR under Projects HKBU 2156/04E, HKBU 210306, and by Faculty Research Grant of Hong Kong Baptist University with the Project Code: FRG/05-06/II-42.

References

[1] X. F. Zhang, C. M. Lam, and W. K. Cheung, "Mining Local Data Sources for Learning Global Cluster Models via Local Model Exchange," *The IEEE Intelligent Informatics Bulletin*, vol. 4, pp. 16–22, 2004.

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

[3] X. F. Zhang, C. M. Lam, and W. K. Cheung, "Mining Local Data Sources for Learning Global Cluster Models via Local Model Exchange," *The IEEE Intelligent Informatics Bulletin*, vol. 4, pp. 16–22, 2004.

[4] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *J. Roy. Stat. Soc.*, vol. B39, pp. 1–38, 1977.

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

[6] C. Chinrungrueng and C. H. Sequin, "Optimal adaptive k-means algorithm with dynamic adjustment of learning rate," *IEEE Transaction on Neural Networks*, vol. 6, no. 1, pp. 157–169, 1995.

[7] A. Dvoretzky, "On stochastic approximation," in *Proc. 3rd Berkeley Sym. on Math. Stat. and Prob.*, J. Neyman, Ed.