Neural Processing Letters (2006) 24:19–40
DOI 10.1007/s11063-006-9008-7

BYY Harmony Learning on Finite Mixture: Adaptive Gradient Implementation and A Floating RPCL Mechanism

JINWEN MA* and LE WANG
Department of Information Science, School of Mathematical Sciences and LMAM, Peking University, Beijing, 100871, China. e-mail: jwma@math.pku.edu.cn

Abstract. In tackling the learning problem on a set of finite samples, Bayesian Ying-Yang (BYY) harmony learning has developed a new learning mechanism that makes model selection implemented either automatically during parameter learning or in help of evaluating a new class of model selection criteria. In this paper, parameter learning with automated model selection has been studied for finite mixture model via an adaptive gradient learning algorithm for BYY harmony learning on a specific bidirectional architecture (BI-architecture). Via theoretical analysis, it has shown that the adaptive gradient learning implements a mechanism of floating rival penalized competitive learning (RPCL) among the components in the mixture. Also, the simulation results are demonstrated well for the adaptive gradient algorithm on the sample data sets from Gaussian mixtures with certain degree of overlap. Moreover, the adaptive gradient algorithm is applied to classification of the Iris data and unsupervised color image segmentation.

Key words. Bayesian Ying-Yang system, harmony learning, finite mixture, rival penalized competitive learning, clustering analysis

1. Introduction

Finite mixture distributions have been used extensively as models in a wide variety of practical situations where data can be viewed as arising from two or more populations mixed in certain proportions. Many studies have been made on mixture modelling as well as clustering analysis on such a sample data set [1]. Usually, it is assumed that the number of component densities in the mixture is pre-known and there have been several statistical methods to do such a task, e.g., the k-means algorithm [2] and the EM algorithm [3]. However, in many cases this key information is not available and the selection of an appropriate number of component densities must be made jointly with the estimation of the parameters, which becomes a rather difficult task [4].

With the Akaike's information criterion [5] or its extensions(e.g. [6, 7]), we can solve such a mixture modelling problem by choosing a best number k^* of component densities as the clusters in the sample data set. However, the process of

*Corresponding author

evaluating a criterion incurs a large computational cost since we need to repeat the entire parameter estimating process at a number of different value of k, even though such a process is attempted to be organized in a more efficient way, e.g., embedding the checking of the criterion value within clustering as did in ISODATA [8].

In the field of artificial neural networks, competitive learning (CL) has been developed for clustering analysis and vector quantization [9]. However, the conventional competitive learning algorithms such as the classical competitive learning algorithm [10] and the frequency sensitive competitive learning algorithm [11], can only be considered as adaptive versions of k-means algorithm and thus are unable to solve the problem of selecting k during clustering analysis. Xu et al. [12] proposed a new kind of competitive learning algorithm, called rival penalized competitive learning (RPCL) algorithm, via such a mechanism that for each input, the winner of the units (i.e., weight vectors) is rewarded to adapt to the input, but the rival (the second winner) is penalized (or delearned) by a smaller learning rate. It was demonstrated that the RPCL algorithm has the ability of automatically allocating an appropriate number of units for a sample data set, with the other extra units being pushed far away from the sample data. Therefore, the RPCL algorithm can be used on a clustering analysis problem with an unknown k, as long as an initial number of the units is given to be larger than the number of actual component densities in the sample data set. In some extended versions of the RPCL algorithm [13], the units have been generalized to be the component densities of a finite mixture, which makes it possible for a mixture modeling problem too.

Alternatively, Bayesian Ying-Yang (BYY) harmony learning system and theory, proposed in 1995 [14] and systematically summarized in [15–17], has provided a theoretical foundation to solve this mixture modeling problem. The BYY harmony learning acts as a general statistical learning framework not only for understanding several existing major learning approaches but also for tackling the learning problem on a set of finite samples with a new learning mechanism that makes model selection implemented either automatically during parameter learning or in help of evaluating a new class of model selection criteria. Applied to the mixture modeling based problem, selection of k is made via either evaluating a criterion derived from the harmony measure or directly maximizing the harmony measure. In the latter case, selection of k can be made automatically during parameter learning via the least complexity nature of BYY harmony learning.

However, the least complexity nature also yields a winner-take-all (WTA) effect that creates a local maximum solution to the harmony maximization. The problem can be taken by either certain internal mechanisms of harmony learning or some external techniques imposed together with learning [15–17]. Internally, either a Tikhonov regularization is introduced via a Parzen window based data smoothing [15, 17, 18] with various experimental supports [19], or a conscience type delearning is resulted from normalization regularization [15] with the experimental support too [20]. Moreover, it has been pointed out firstly in [15] and then further

elaborated in [17] that the latter case actually implements a RPCL mechanism that has its penalizing per sample during learning. Externally, the local maximum problem may also be solved by combining a Maximum-Likelihood(ML)-equivalent learning with the harmony via a weighted sum [15] or introducing simulated annealing like techniques that make learning gradually switched from a ML-equivalent learning to the harmony learning [15, 21]. In implementation, both cases are equivalent to replacing the WTA competition in the harmony learning and the Bayesian posterior allocation in the ML learning by some average of the both [15, 21]. Another internal mechanism, that is suggested but without further exploration in [15], is using a bidirectional architecture (BI-architecture) with its Ying path for the finite mixture and with its Yang path for regularization. In [22, 23], some batch way gradient implementations of BYY harmony learning with a BI-architecture were made on Gaussian mixture and experiments have shown that a RPCL like mechanism also occurs during parameter learning with automated model selection.

In the current paper, we study the implementation of an adaptive version of the gradient learning algorithm on finite mixture for this purpose, and analyze the learning mechanism on how it links to the RPCL learning with a floating penalizing. The simulation experiments have shown that the adaptive gradient algorithm in the Gaussian mixture setting is efficient in comparison with the batch gradient learning algorithms. Moreover, it is successfully applied to classification of the Iris data and unsupervised color image segmentation.

In the sequel, the adaptive gradient learning algorithm is derived in Section 2. Its learning mechanism is further analyzed in Section 3. Several simulation and practical experiments are conducted in Section 4 to demonstrate the algorithm for Gaussian mixture. Finally, we conclude in Section 5.

2. Adaptive Gradient Learning Algorithm

2.1. BYY HARMONY FUNCTION ON FINITE MIXTURE

We begin with a brief description of a BI-architecture of the BYY system on which the harmony learning is equivalent to the parameter learning on the finite mixture model, and leave the details to Xu [15].

A BYY system describes each observation $x \in \mathcal{X} \subset R^d$ and its corresponding inner representation $y \in \mathcal{Y} \subset R^m$ via the two types of Bayesian decomposition of the joint density $p(x, y) = p(x)p(y|x)$ and $q(x, y) = q(x|y)q(y)$, being called Yang machine and Ying machine, respectively. Given a data set $D_x = \{x_t\}_{t=1}^N$, the task of learning on a BYY system consists of specifying all the aspects of $p(y|x), p(x), q(x|y), q(y)$ with a harmony learning principle implemented by maximizing the functional

$$H(p||q) = \int p(y|x)p(x)ln[q(x|y)q(y)]\mathrm{d}x\mathrm{d}y - lnz_q, \tag{1}$$

where z_q is a regularization term that implements either data smoothing or normalization as mentioned previously in the introduction section.

If $p(y|x)$ and $q(x|y)$ are both parametric, i.e., from a family of probability densities with a parameter θ, the BYY system is called to have a BI-architecture for short. For the finite mixture modeling, we use the following BI-architecture of the BYY system. The inner representation y is discrete, i.e., $y \in \mathcal{Y} = \{1, 2, \ldots, k\} \subset R$ with $m = 1$. Then, we let $q(y = j) = \alpha_j \geqslant 0$ with $\sum_{j=1}^{k} \alpha_j = 1$. Also, we ignore the regularization term z_q (i.e., set $z_q = 1$) and let $p(x)$ be directly given by the empirical density $p_0(x)$:

$$p_0(x) = \frac{1}{N} \sum_{t=1}^{N} \delta(x - x_t), \tag{2}$$

where $x \in \mathcal{X} = R^d$ and $\delta(\cdot)$ is a kind of kernel function (e.g., Gaussian function). Moreover, the Yang path is constructed with the following parametric form:

$$p(y = j|x) = \frac{\alpha_j q(x|\theta_j)}{q(x|\Theta_k)}, \qquad q(x|\Theta_k) = \sum_{j=1}^{k} \alpha_j q(x|\theta_j), \tag{3}$$

where $q(x|\theta_j)$ denotes $q(x|y = j)$ with θ_j consisting of all its parameters and $\Theta_k = \{\alpha_j, \theta_j\}_{j=1}^{k}$. In this case, the BYY system implements a finite mixture model $q(x|\Theta_k)$, with regularization imposed via Equation (3).

Using a specific design of the Yang path to replace the regularization role of the term $\ln z_q$ that actually should be computed on the entire set of samples, BYY harmony learning on a BI-architecture is more suitable for being implemented adaptively, in comparison with BYY harmony learning with regularization by either data smoothing or normalization.

With all these component densities into Equation (1) and letting the kernel functions converge to the delta functions, $H(p||q)$ becomes the following harmony function on the parameters Θ_k:

$$J(\Theta_k) = \frac{1}{N} \sum_{t=1}^{N} \sum_{j=1}^{k} \frac{\alpha_j q(x_t|\theta_j)}{\sum_{i=1}^{k} \alpha_i q(x_t|\theta_i)} ln[\alpha_j q(x_t|\theta_j)], \tag{4}$$

which was originally introduced in [14] as a criterion for selection of k and then further suggested for parameter learning with automated selection of k [15].

2.2. ADAPTIVE GRADIENT LEARNING ALGORITHM

Denoting $U_j(x) = \alpha_j q(x|\theta_j)$, for $j = 1, 2, \ldots, k$, $J(\Theta_k)$ has the following expression:

$$J(\Theta_k) = \frac{1}{N} \sum_{t=1}^{N} J_t(\Theta_k), \qquad J_t(\Theta_k) = \sum_{j=1}^{k} \frac{U_j(x_t)}{\sum_{i=1}^{k} U_i(x_t)} ln U_j(x_t). \tag{5}$$

Considering the constraints on α_j, we use the following so-called softmax representation:

$$\alpha_j = \frac{e^{\beta_j}}{\sum_{i=1}^{k} e^{\beta_i}}, \quad j = 1, \ldots, k, \tag{6}$$

where $-\infty < \beta_1, \ldots, \beta_k < +\infty$.

Then, we can get the following derivatives of $J(\Theta_k)$ per sample x_t with respect to β_j and θ_j:

$$\begin{aligned}\frac{\partial J_t(\Theta_k)}{\partial \beta_j} &= \sum_{i=1}^{k} \frac{\partial J_t(\Theta_k)}{\partial U_i(x_t)} \frac{\partial U_i(x_t)}{\partial \beta_j} \\ &= \frac{1}{q(x_t|\Theta_k)} \sum_{i=1}^{k} [1 - \sum_{l=1}^{k} (p(l|x_t) - \delta_{il}) ln U_l(x_t)](\delta_{ij} - \alpha_j) U_i(x_t), \end{aligned} \tag{7}$$

$$\begin{aligned}\frac{\partial J_t(\Theta_k)}{\partial \theta_j} &= \sum_{i=1}^{k} \frac{\partial J_t(\Theta_k)}{\partial U_i(x_t)} \frac{\partial U_i(x_t)}{\partial \theta_j} \\ &= \frac{1}{q(x_t|\Theta_k)} [1 - \sum_{l=1}^{k} (p(l|x_t) - \delta_{jl}) ln U_l(x_t)] \alpha_j \frac{\partial q(x_t|\theta_j)}{\partial \theta_j}, \end{aligned} \tag{8}$$

where δ_{ij} is the Kronecker function.

Letting

$$\lambda_i(t) = 1 - \sum_{l=1}^{k} (p(l|x_t) - \delta_{il}) ln U_l(x_t), \quad i = 1, \ldots, k, \tag{9}$$

it follows from Equations (7) and (8) that we have the adaptive gradient learning rule:

$$\Delta \beta_j = \frac{\eta}{q(x_t|\Theta_k)} \sum_{i=1}^{k} \lambda_i(t)(\delta_{ij} - \alpha_j) U_i(x_t), \tag{10}$$

$$\Delta \theta_j = \frac{\eta \lambda_j(t) \alpha_j}{q(x_t|\Theta_k)} \frac{\partial q(x_t|\theta_j)}{\partial \theta_j} = \eta p(j|x_t) \lambda_j(t) \frac{\partial ln q(x_t|\theta_j)}{\partial \theta_j}, \tag{11}$$

where η denotes the learning rate that starts from a reasonable initial value and then reduces to zero with the iteration number n in such a way that $0 \leqslant \eta(n) \leqslant 1$, and

$$\lim_{n \to \infty} \eta(n) = 0, \qquad \sum_{n=1}^{\infty} \eta(n) = \infty, \tag{12}$$

i.e., in the way used in the Robbin–Monro stochastic approximation procedure [24]. The typical example of the learning rate satisfying Equation (12) is $\eta(n) = \eta_0/n$, where η_0 is a positive constant.

For the following experiments, we detail the adaptive gradient learning algorithm for Gaussian mixture. In this case, $q(x|\theta_j)$ is given by

$$q(x|\theta_j) = q(x|m_j, \Sigma_j) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma_j|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-m_j)^T \Sigma_j^{-1}(x-m_j)}, \tag{13}$$

where m_j is the mean vector and Σ_j is the covariance matrix which is positive definite. Its derivatives take the following form:

$$\frac{\partial q(x_t|m_j, \Sigma_j)}{\partial m_j} = q(x_t|m_j, \Sigma_j)\Sigma_j^{-1}(x_t - m_j), \tag{14}$$

$$\frac{\partial q(x_t|m_j, \Sigma_j)}{\partial \Sigma_j} = \frac{1}{2} q(x_t|m_j, \Sigma_j)[\Sigma_j^{-1}(x_t - m_j)(x_t - m_j)^T - I]\Sigma_j^{-1}, \tag{15}$$

where I is the d-dim identity matrix. Substituting the derivative of $q(x_t|m_j, \Sigma_j)$ with respect m_j into Equation (11), we then have the following adaptive update rule of m_j:

$$\Delta m_j = \eta p(j|x_t)\lambda_j(t)\Sigma_j^{-1}(x_t - m_j). \tag{16}$$

However, since Σ_j is constrained to be positive definite, we cannot guarantee its positive definite in the adaptive iteration if the derivative of $q(x_t|m_j, \Sigma_j)$ with respect to Σ_j is substituted into Equation (11) directly. Instead, we use the decomposition technique suggested in [23], that is, letting $\Sigma_j = B_j B_j^T$, where B_j is a nonsingular square matrix, we have the following adaptive update rule of B_j:

$$\Delta \mathrm{vec} B_j = \frac{\eta}{2} p(j|x_t)\lambda_j(t) \frac{\partial (B_j B_j^T)}{\partial B_j} \mathrm{vec}[\Sigma_j^{-1}(x_t - m_j)(x_t - m_j)^T \Sigma_j^{-1} - \Sigma_j^{-1}], \tag{17}$$

where vec$[A]$ denotes the vector obtained by stacking the column vectors of the matrix A, and

$$\frac{\partial (BB^T)}{\partial B} = I_{d\times d} \bigotimes B_{d\times d}^T + E_{d^2\times d^2} \cdot B_{d\times d}^T \bigotimes I_{d\times d},$$

where $\bigotimes$ denotes the Kronecker product (or tensor product), and

$$E_{d^2\times d^2} = \frac{\partial B^T}{\partial B} = (\Gamma_{ij})_{d^2\times d^2} = \begin{pmatrix} \Gamma_{11} & \Gamma_{12} & \cdots & \Gamma_{1d} \\ \Gamma_{21} & \Gamma_{22} & \cdots & \Gamma_{2d} \\ \cdots & \cdots & \cdots & \cdots \\ \Gamma_{d1} & \Gamma_{d2} & \cdots & \Gamma_{dd} \end{pmatrix}_{d^2\times d^2},$$

where Γ_{ij} is a $d \times d$ matrix whose (j, i)th element is just 1, with all the other elements being zero. Combining Equations (10), (16), and (17) together, we get the specific adaptive gradient learning algorithm for Gaussian mixture in which α_j and Σ_j are adaptively updated accordingly.

3. Analysis of the Adaptive Gradient Rule

In this section, we analyze the learning mechanism implied in the adaptive gradient learning rule given by Equations (10) and (11). First, we give some properties of $\lambda_j(t)$ via the following theorem:

THEOREM 1. *For each sample x_t, we have*

(i) *If $p(j_1|x_t) \leqslant p(j_2|x_t) \leqslant \cdots \leqslant p(j_k|x_t)$ under the permutation $\{j_1, j_2, \ldots, j_k\}$ of $\{1, 2, \ldots, k\}$, then $\lambda_{j_1}(t) \leqslant \lambda_{j_2}(t) \leqslant \cdots \leqslant \lambda_{j_k}(t)$. Moreover, there exists a threshold value $T(t) = e^{-(1+H(x_t))}$ such that if $p(j|x_t) > (\leqslant) T(t)$, $\lambda_j(t) > (\leqslant) 0$, where*

$$H(x_t) = H(p(1|x_t), \ldots, p(k|x_t)) = -\sum_{i=1}^{k} p(i|x_t) lnp(i|x_t); \tag{18}$$

(ii) *If $j_c = argmax p(j|x_t)$, then $\lambda_{j_c}(t) > 0$;*

(iii) *If $p(j|x_t) > \frac{1}{e}$, then $\lambda_j(t) > 0$; if $p(j|x_t) \leqslant \frac{1}{ke}$, $\lambda_j(t) < 0$, where e is the natural number.*

See the Appendix for the proof.

That is, $\lambda_j(t)$ is floating with $p(j|x_t)$ in the same descent order at each sample x_t and there exists a threshold value $T(t)$ such that if $p(j|x_t) > (\leqslant) T(t)$, $\lambda_j(t) > (\leqslant) 0$. Obviously, $T(t)$ varies with the sample x_t via the Shannon entropy of the posterior probabilities $p(j|x_t)$ of the components in the finite mixture. As $H(x_t)$ is high, i.e., the belonging component of x_t is obscure, $T(t)$ becomes low; otherwise, as $H(x_t)$ is low, i.e., the belonging component of x_t is clear, $T(t)$ becomes high.

Moreover, when $T(t)$ is low, there are generally several components with $\lambda_j(t) > 0$; otherwise, when $T(t)$ is high, there are a few components or just a single component with $\lambda_j(t) > 0$. If component j is the maximum one at the sample x_t, $\lambda_j(t)$ must be positive. On the other hand, if component j is relatively very small, $\lambda_j(t)$ becomes negative.

We consider the incremental of component $U_j(x_t)$ by the update of the adaptive gradient leaning rule and have

$$\Delta U_j(x_t) = q(x_t|\theta_j) \sum_{i=1}^{k} \frac{\partial \alpha_j}{\partial \beta_i} \Delta\beta_i + \alpha_j \left[\frac{\partial q(x_t|\theta_j)}{\partial \theta_j}\right]^T \Delta\theta_j. \tag{19}$$

It follows from Equations (10) and (11) that

$$\Delta U_j(x_t) = \eta U_j(x_t)\Big[(\lambda_j(t)p(j|x_t) - \sum_{i=1}^{k} \alpha_i \lambda_i(t) p(i|x_t)) + \Big(\sum_{i=1}^{k} \alpha_i^2 - \alpha_j\Big) + \frac{\alpha_j \lambda_j(t)}{q(x_t|\Theta_k) q(x_t|\theta_j)} \Big|\Big| \frac{\partial q(x_t|\theta_j)}{\partial \theta_j} \Big|\Big|^2\Big]. \tag{20}$$

For analysis, we let

$$\gamma_j(t) = \gamma_{j1}(t) + \gamma_{j2}(t), \tag{21}$$

where

$$\gamma_{j1}(t) = \lambda_j(t)\, p(j|x_t) - \sum_{i=1}^{k} \alpha_i \lambda_i(t)\, p(i|x_t); \tag{22}$$

$$\gamma_{j2}(t) = \sum_{i=1}^{k} \alpha_i^2 - \alpha_j. \tag{23}$$

We then have

$$\Delta U_j(x_t) = \eta U_j(x_t)\gamma_{j1}(t) + \eta U_j(x_t)\gamma_{j2}(t) + \eta \frac{\alpha_j^2 \lambda_j(t)}{q(x_t|\Theta_k)} \left|\left| \frac{\partial q(x_t|\theta_j)}{\partial \theta_j} \right|\right|^2. \tag{24}$$

According to Equation (22) and (i), it can be found that $\gamma_{j1}(t)$ also varies with $p(i|x_t)$ and there is another threshold value $T'(t)$ such that if $p(i|x_t) > (\leqslant) T'(t)$, $\gamma_{j1}(t) > (\leqslant) 0$. Since $\sum_{i=1}^{k} \alpha_i \lambda_i(t) p(i|x_t)$ is a weighted sum of these $\lambda_i(t) p(i|x_t)$, $\gamma_{j1}(t)$ becomes positive if $p(j|x_t)$ is large enough. Therefore, $\gamma_{j1}(t)$ behaves similarly as $\lambda_i(t)$, but their threshold values for $p(j|x_t)$ may be different.

On the other hand, $\gamma_{j2}(t)$ is only related to the mixing proportions $\alpha_1, \ldots, \alpha_k$. It follows from Equation (23) that if α_j is larger than a threshold value $\hat{\alpha} = \sum_{i=1}^{k} \alpha_i^2$, $\gamma_{j2}(t)$ will be negative; otherwise, it will be positive.

With the above preparations, we can uncover the competitive learning mechanism among the mixture components $U_1, \ldots, U_k$ by the adaptive gradient learning rule. When a sample x_t is inputted, the component parameters $\beta_j, \alpha_j, \theta_j$ are modified according to Equations (6), (10), and (11), respectively. Accordingly, component U_j is modified and its gain[1] consists of three parts: $\eta U_j(x_t)\gamma_{j1}(t)$, $\eta U_j(x_t)\gamma_{j2}(t)$, and $\eta \frac{\alpha_j^2 \lambda_j(t)}{q(x_t|\Theta_k)} || \frac{\partial q(x_t|\theta_j)}{\partial \theta_j} ||^2$.

Since both the descent queuing orders of $\gamma_{j1}(t)$ in $\{\gamma_{j1}(t)\}_{j=1}^{k}$ and $\lambda_j(t)$ in $\{\lambda_j(t)\}_{j=1}^{k}$ are that of $p(j|x_t)$ in $\{p(j|x_t)\}_{j=1}^{k}$, the first and last parts of the gain can be both considered as a result of the competition among the components $U_1, \ldots, U_k$ at the sample x_t through the posterior probabilities $p(1|x_t), \ldots, p(k|x_t)$. By each of these two parts, all the posterior probabilities can be put in a descent order and there is a threshold value $T(t)$ such that if $p(j|x_t) > T(t)$, the part of the gain is positive and U_j becomes a winner and is rewarded to increase; otherwise, if $p(j|x_t) < T(t)$, the part of the gain is negative and U_j becomes a loser and is penalized to decreased. Because $U_j(x_t) = q(x_t|\Theta_k) p(j|x_t)$, the competition of the posterior probabilities is equivalent to that of the components themselves.

[1]Since η is very small, we neglect its quadratic or higher order terms in the incremental of U_j at the sample x_t.

By the middle part of the gain, we further find a proportion conscience mechanism implied in the adaptive gradient learning rule. Clearly, $\hat{\alpha} = \sum_{i=1}^{k} \alpha_i^2 \in (\min_i \alpha_i, \max_i \alpha_i)$. Then, if $\alpha_j > \hat{\alpha}$, $\gamma_{j2}(t)$ becomes negative, with U_j being penalized to decrease. Otherwise, if $\alpha_j < \hat{\alpha}$, $\gamma_{j2}(t)$ becomes positive, with U_i being rewarded to increase. This is just a proportion conscience mechanism that the components with large proportions tends to be penalized, while the components with small proportions tends to be rewarded.

As a whole, we have found that the leaning mechanism implied in the adaptive gradient learning rule is a kind of floating RPCL mechanism among the components in the mixture. That is, the number of winners (or losers) is floating with the sample x_t and the gains of the components are also regulated by the mixing proportions via a conscience mechanism.

By the simulation experiments conducted in Section 4, we have also shown the floating RPCL mechanism in the adaptive gradient algorithm. In the experiments, the largest component U_{j_c} is always a winner and is mostly rewarded. At the beginning of the competition, there may be several winners or even all the components become the winners in a special case that each posterior probability is considerable and all the mixing proportions are equal. However, as the competitive learning continues and the extra components attenuates to zero, the number of the winners at a sample x_t gets less and finally becomes one. In this way, each component dominantly occupying the samples from one of the actual component densities will be strongly rewarded to conform to this true density multiplied by its mixing proportion, while the mixing proportions of the extra densities are finally penalized to reduce to zero.

Being different from the original RPCL rule [12], the adaptive gradient learning rule floats the numbers of winners and losers in each competition with the sample x_t. Such a floating characteristic may be more reasonable for automated model selection. In fact, at the beginning of the competitive learning, since we get less information from the sample data, $T(t)$ tends to be low such that all the possible candidates to match some actual component, will not be penalized. As the competitive learning continues, we get more information from the sample data and $T(t)$ tends to be high. In this way, the number of winners gets less and the convergence of the learning is speeded up. Finally, we get enough information to make clear that each sample comes from a unique component, i.e, there is just a unique winner for each sample.

4. Experimental Results

In this section, several simulation experiments are carried out to demonstrate the performance of the adaptive gradient learning algorithm for both model selection and parameter estimation on a sample data set from a Gaussian mixture, also being compared with those of the batch gradient learning algorithms as well as some other existing algorithms. Moreover, the adaptive gradient learning

algorithm is applied to classification of the Iris data and unsupervised color image segmentation.

4.1. SIMULATION EXPERIMENTS

4.1.1. *Sample Data Sets*

We begin with a brief description of the four sets of sample data used for our simulation experiments. We conducted four Monte Carlo experiments in which samples were drawn from a mixture of four or three bivariate Gaussians distributions on the plane coordinate system (i.e., $d = 2$). As shown in Figure 1, each data set of samples are generated with a certain degree of overlap among the clusters (Gaussians) in the mixture. They are four typical sets of sample data from Gaussian mixtures. The clusters in $\mathcal{S}_1$ are sphere-shaped, with the equal number of samples. But those in $\mathcal{S}_2$ are ellipse-shaped, with different numbers of samples.

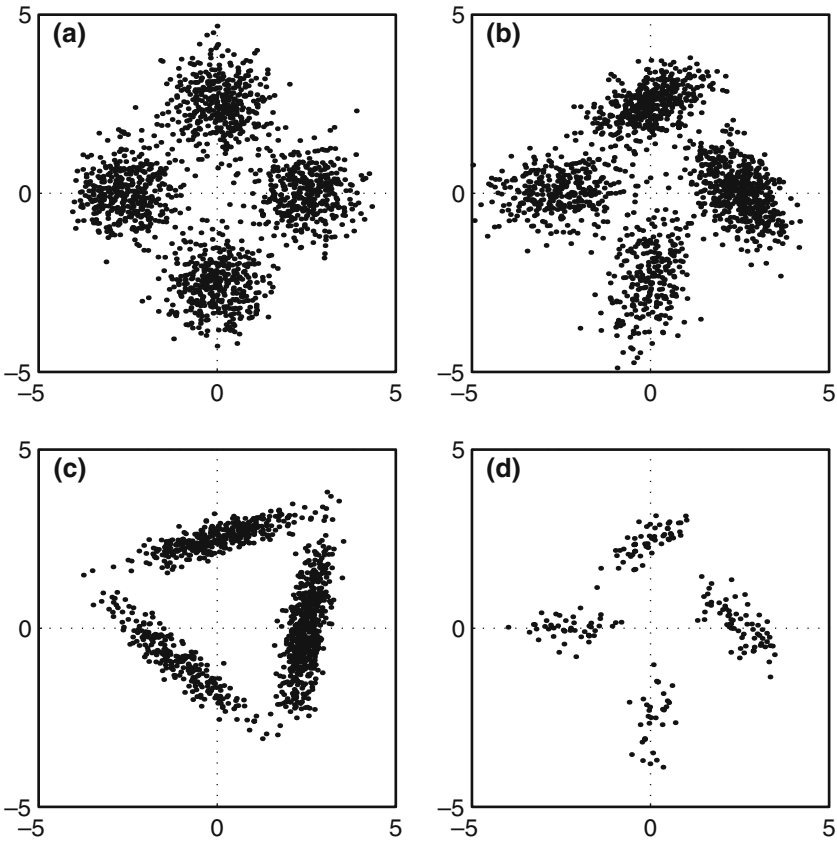

Figure 1. Four sets of sample data of Gaussian mixtures used in the experiments: (a) The first set of sample data $\mathcal{S}_1$; (b) The second set of sample data $\mathcal{S}_2$; (c) The third set of sample data $\mathcal{S}_3$; (d) The fourth set of sample data $\mathcal{S}_4$.

Moreover, $\mathcal{S}_3$ consists of three very flat clusters and $\mathcal{S}_4$ has a small number of samples, with the same structure of $\mathcal{S}_2$. The detailed parameters for these four sets of sample data are given in Table I where $m_i, \Sigma_i = [\sigma^i_{jk}]_{2\times 2}, \alpha_i$ and N_i denote the mean vector, covariance matrix, mixing proportion, and the number of samples of the ith Gaussian(cluster), respectively.

4.1.2. *Section of Initial Parameters*

In our simulation experiments and application experiments in the sequential subsections, k is always selected to be larger than k^*, i.e., the number of actual Gaussians in a sample or real data set. As mentioned in Section 1, we do not know the true number k^* of actual Gaussians in a data set. Generally, we can overestimate it with some additional information in the data set. If this can be done, we can let the overestimate of k^* be k so that $k > k^*$. Certainly, we can always choose a large number as k such that it is surely larger than k^* of the data set. But this may not be desirable since a large $k(>>k^*)$ will increase the training time and the risk of model selection error (see the following simulation analysis). Fortunately, an efficient procedure via the RPCL algorithm was suggested in [12] to get an appropriate k for a data set. We can start with a small number or a guess for k and perform the RPCL algorithm on the data set. By checking whether there are spare units (i.e., far-away weight vectors) left, we would know if $k > k^*$. If yes,

Table I. The parameters in the actual Gaussian mixtures to the three sets of sample data.

The sample set	Gaussian	m_i	σ^i_{11}	σ^i_{12}	σ^i_{22}	α_i	N_i
$\mathcal{S}_1$	Gaussian 1	(2.50, 0)	0.50	0	0.50	0.25	400
($N = 1600$)	Gaussian 2	(0, 2.50)	0.50	0	0.50	0.25	400
	Gaussian 3	(−2.50, 0)	0.50	0	0.50	0.25	400
	Gaussian 4	(0, −2.50)	0.50	0	0.50	0.25	400
$\mathcal{S}_2$	Gaussian 1	(2.50, 0)	0.45	−0.25	0.55	0.34	544
($N = 1600$)	Gaussian 2	(0, 2.50)	0.65	0.20	0.25	0.28	448
	Gaussian 3	(−2.50, 0)	1	0.10	0.35	0.22	352
	Gaussian 4	(0, −2.50)	0.30	0.15	0.80	0.16	256
$\mathcal{S}_3$	Gaussian 1	(2.50, 0)	0.10	−0.20	1.25	0.50	600
($N = 1200$)	Gaussian 2	(0, 2.50)	1.25	0.35	0.15	0.30	360
	Gaussian 3	(−1, −1)	1	−0.80	0.75	0.20	240
$\mathcal{S}_4$	Gaussian 1	(2.50, 0)	0.28	−0.20	0.32	0.34	68
($N = 200$)	Gaussian 2	(0, 2.50)	0.34	0.20	0.22	0.28	56
	Gaussian 3	(−2.50, 0)	0.50	0.04	0.12	0.22	44
	Gaussian 4	(0, −2.50)	0.10	0.05	0.50	0.16	32

then take this value; if not, increase k by adding a positive increment and perform the RPCL algorithm again. The same procedure can be repeated until $k > k^*$. In this way, we can get a moderately large number as k of the algorithm for the data set.

The other parameters in the adaptive gradient learning algorithm can be initialized randomly within certain intervals under the constraints. Actually, the initial value of β_j can be freely selected from some interval. But it has been found by the experiments that when the initial values of these β_j are equal or close and thus the initial values of the corresponding α_j are equal or close, the adaptive gradient learning algorithm converges more efficiently. Thus, we give the same initial value or close values to these β_j. As for m_j, we can select its initial value in the field of the sample data or some sample point. However, the initial matrix of B_j should be nonsingular. For convenience, we can let the initial matrix of each B_j be the identity matrix or around it. The learning rate η is given by $\eta(n) = \eta_0/n$ with $\eta_0 = 0.1$ for the simulation experiments. The algorithm is stopped when $|J(\Theta_k^{\text{new}}) - J(\Theta_k^{\text{old}})| < 10^{-7}$.

4.1.3. *Performance of Model Selection*

We implemented the adaptive gradient learning algorithm on these four typical sets of sample data. The experimental results on $\mathcal{S}_1$ and $\mathcal{S}_2$ are given in Figures 2 and 3, respectively, with case $k = 8$ and $k^* = 4$. We observe that four Gaussians are finally located accurately in either the sphere-shaped and symmetric structure case or the ellipse-shaped and asymmetric structure case, while the mixing proportions of the other four Gaussians were reduced to below 0.0001, i.e, these Gaussians are extra and can be discarded. Therefore, the correct number of the clusters have been detected on these data sets. Moreover, the experiment has been made on $\mathcal{S}_3$ with $k = 8, k^* = 3$. As shown in Figure 4, clusters are far from spherical shapes (actually they are very flat). Again, three Gaussians are located accurately, while the mixing proportions of the other five extra Gaussians become less than 0.001. That is, the correct number of the clusters can still be detected on such a special data set. Furthermore, the adaptive gradient learning algorithm was also implemented on $\mathcal{S}_4$ with $k = 8, k^* = 4$. As shown in Figure 5, even each cluster has a small number of samples, the correct number of clusters can still be detected, with the mixing proportions of other four extra Gaussians reduced below 0.0002. Therefore, the adaptive gradient learning algorithm can make model selection automatically during the parameter learning in each of these sample data sets.

The further experiments of the adaptive gradient algorithm had been also made on these sets of sample data in the similar cases. Actually, a failure on the correct number detection rarely happen when we initially set $k^* \leqslant k \leqslant 3k^*$. However, the adaptive gradient learning may lead to a wrong detection when $k > 3k^*$. On the other hand, an extra Gaussian may be stable with any shape as its mixing

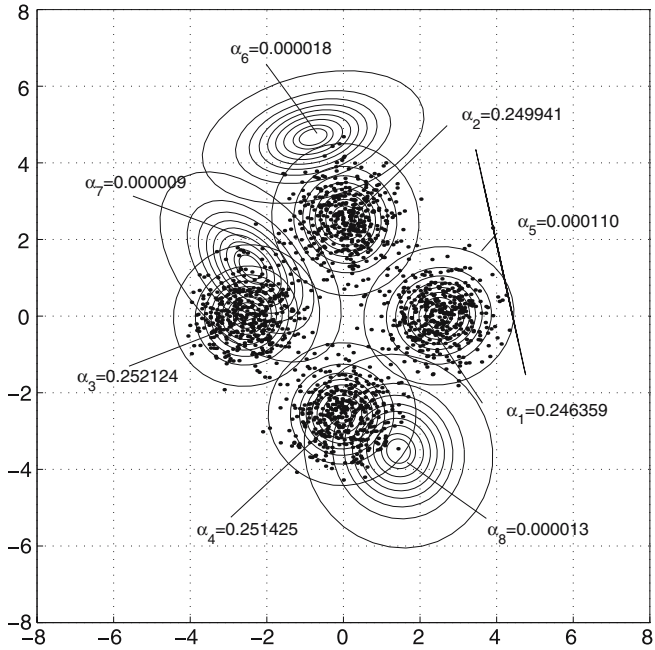

Figure 2. The experimental result of the adaptive gradient algorithm on the sample set $\mathcal{S}_1$ (after 4787×1600 iterations). In this and the following three figures, the contour lines of each Gaussian are retained unless its density is less than e^{-3}(peak).

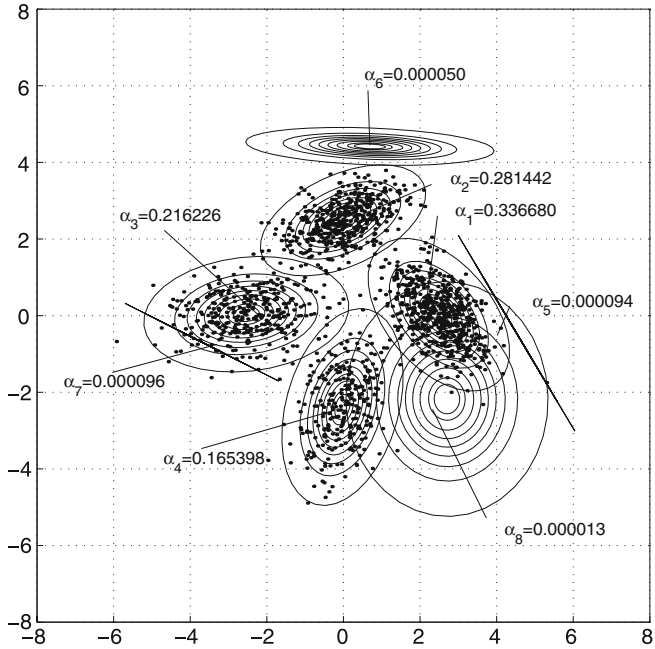

Figure 3. The experimental result of the adaptive gradient learning algorithm on the sample set $\mathcal{S}_2$ (after 4111×1600 iterations).

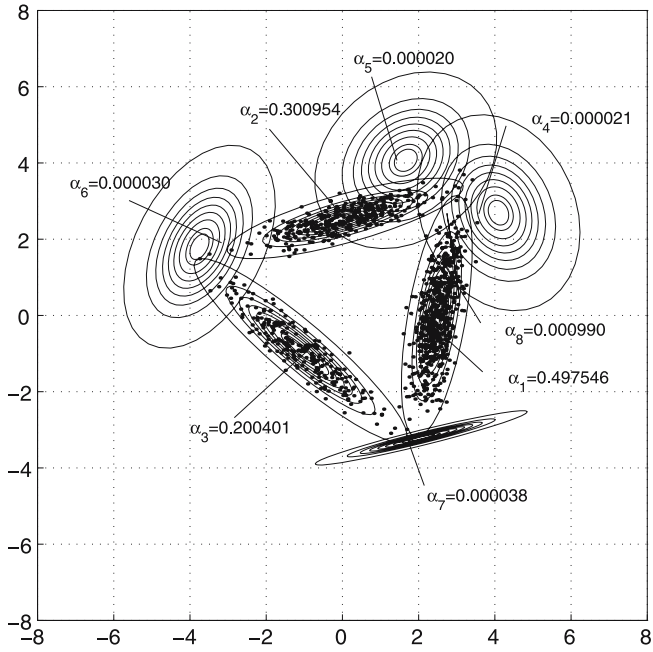

Figure 4. The experimental result of the adaptive gradient learning algorithm on the sample set $\mathcal{S}_3$ (after 3044×1200 iterations).

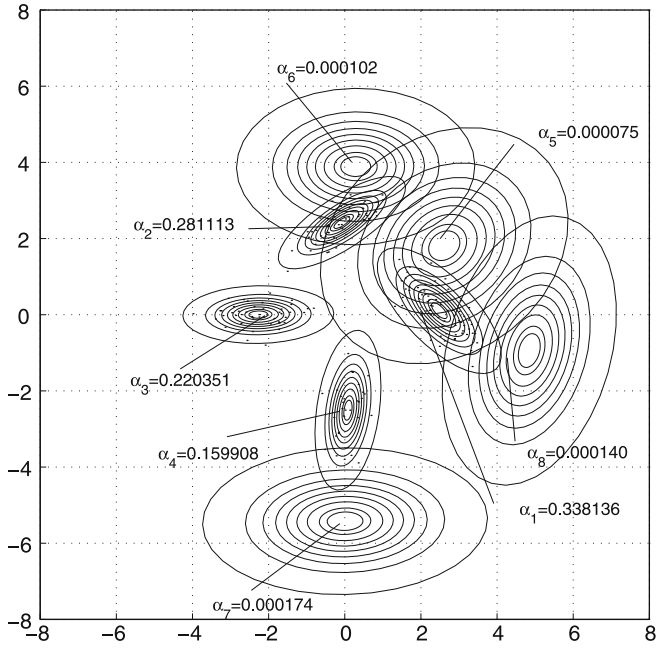

Figure 5. The experimental result of the adaptive gradient learning algorithm on the sample set $\mathcal{S}_4$ (after 10111×200 iterations).

Table II. The average errors of the estimated parameters by the adaptive gradient learning and EM algorithms on the four data sets.

$\mathcal{S}$	$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$
AGL algorithm	0.027539	0.025610	0.034658	0.016092
EM algorithm	0.019434	0.015928	0.029762	0.008623

proportion attenuates to zero. Sometimes, it degenerates into a 1-dim Gaussian taking a shape of a line.

4.1.4. *Parameter Estimation*

In addition to the automated model selection, we further compared the converged values of parameters (discarding the extra Gaussians) with those parameters in the true Gaussian mixture where the samples come from. The performance metric of average error is adopted as the absolute deviation between the estimated parameters and the true parameters in each case. In Table II, we give the average errors of the adaptive gradient learning (AGL) algorithm. For comparison, we also give the average errors of the EM algorithms on these four sets of sample data under $k = k^*$ in Table II.

From Table II, we can find that the adaptive gradient learning algorithm usually converges to a good estimate of the true parameters with a low average error. However, the EM algorithm slightly outperform the adaptive gradient learning algorithm on the parameter estimation. That is, there is a slight deviation between the BYY harmony learning estimate and the maximum likelihood estimate (obtained from the EM algorithm). It is clear that this deviation is caused by the floating RPCL mechanism of the adaptive gradient learning algorithm that each Gaussian may be pushed or penalized with a number of samples.

4.1.5. *Comparison with the Other Approaches*

As compared with the batch gradient learning algorithms [22, 23] on the same data set, we have found the following facts from the experiments:

(1) The convergence speed of the adaptive gradient learning algorithm is generally much faster than that of the traditional batch gradient algorithm [22]. However, it may be slower than those of the conjugate and natural gradient learning algorithms [23] in the cases where the overlap among Gaussians is weak, i.e., the actual Gaussians are well-separated. Oppositely, as the overlap among Gaussians in the sample data set becomes strong, it is considerably faster that those of the conjugate and natural gradient algorithms.

(2) The adaptive gradient learning algorithm converges to the correct solution more frequently than the batch gradient algorithms do in the cases that the overlap among the Gaussians in a sample data set becomes strong or k is much greater than k^*. That is, the adaptive gradient algorithm can search the global maximum of $J(\Theta_k)$ more efficiently.

As a result, the adaptive gradient learning algorithm is more efficient that the bath gradient learning algorithms for the practical applications, which is really demonstrated by the practical applications in the following two subsections. As compared with two typical methods of the maximum certainty partitioning using a number of kernel functions [25] and the variational Bayesian learning [26] for parameter learning and model selection on Gaussian mixture, our adaptive learning algorithm can lead to a similar or better result, but have much less computation.

4.2. CLASSIFICATION OF THE IRIS DATA

We further applied the adaptive gradient learning algorithm to classification (or recognition) of the Iris data [27] which is a typical real data set for testing a classification algorithm. Actually, it consists of 150 samples of three classes where each class contains 50 samples and each sample or datum is four-dimensional and consists of measures of the plants morphology. Although there is the class index for each sample, we could not use the class indexes of these samples in the adaptive gradient learning algorithm since it learns in an unsupervised way. However, the class indexes were used to check the classification result of the adaptive gradient learning algorithm on the Iris data.

The adaptive gradient learning algorithm was applied to the Iris data unsupervised classification problem by setting $k=6$, i.e., the two times of the real class number 3. For the other initial parameters, we set $\beta_j = 1 + 0.2 \times \text{rand}(1)$, $B_j = (1 + 0.2 \times \text{rand}(1)) \times I_4$, $\eta_n = 0.006/n^{0.8}$, where rand(1) is a random number in [0,1], and I_4 is the four-order identity matrix. Note that the learning rate here is slightly different from the one given in Section 4.1B for slow-down in the learning rate reduction. For quick convergence of the algorithm, we also set a low threshold value $T=0.04$ such that when some $\alpha_j < T$, we cancel the jth Gaussian in the mixture for the following learning iterations. The algorithm was also stopped when $|J(\Theta_k^{\text{new}}) - J(\Theta_k^{\text{old}})| < 10^{-7}$. It was shown by the experiments that the adaptive gradient learning algorithm can detect the three classes in the Iris data with an optimal classification accuracy of 96.7% (there are only five errors in the second class), which is slightly less than the classification accuracy 98% (there are three errors) of the maximum certainty partitioning method with a large number of linear mixing kernels (Gaussian functions) [25]. However, the Iris data can be classified completely, i.e., the classification accuracy 100%, via a recently proposed probabilistic ensemble simplified fuzzy ARTMAP (SFAM) classifier that employs a probabilistic plurality voting approach on a number of supervised learning classifiers SFAMs [28].

4.3. UNSUPERVISED COLOR IMAGE SEGMENTATION

We finally applied the adaptive gradient learning algorithm to unsupervised color image segmentation that has been considered as a promising and challenging area in image processing [29]. Here, we utilized each Gaussian in the adaptive gradient learning algorithm to represent an object (including the background) in a color image; however, the number of actual objects is not given in advance. We set k to be larger than the true number k^* of the actual objects and the pixels in the image are partitioned according to the maximum posteriori probabilities among $p(j|x_t)$.

We used three typical color images of house, jellies, and peppers in the segmentation experiments. Each pixel in the image was represented by a three-dimensional real point. In our experiments, we first regularized all the three coordinates of the pixels in each color image via dividing them by 32 so that the regularized coordinates were within an interval of [0,8]. We then ran the adaptive gradient learning algorithm on the data sets of these three color images, respectively, by letting $k = 6$ (We could guess that the number of objects in a color image is less than 6) with the other initial parameters set by $\beta_j = 1 + 0.1 \times \text{rand}(1)$, $B_j = (1.1 + 0.1 \times \text{rand}(1)) \times I_3$, $\eta_n = 0.0008/n$. Again, we set a low threshold value $T = 0.04$ such that when some $\alpha_j < T$, we cancel the jth Gaussian in the mixture for the following learning iterations. The algorithm was also stopped when $|J(\Theta_k^{\text{new}}) - J(\Theta_k^{\text{old}})| < 10^{-7}$.

The first experiment was made on the color image of house which is shown in Figure 6(a). We implemented the adaptive gradient learning algorithm on this color image with the above initial settings and leaded to the segmentation results shown in Figure 6(b). It can be found that two objects (the house and the background) are finally located accurately, while the mixing proportions of the other four Gaussians (objects) are reduced to below 0.04, i.e, these objects are extra and discarded from the figure. That is, the correct number of the actual objects had been detected on the color image with an accurate segmentation. Moreover, the second experiment had been made on the color image of jellies, which is shown in Figure 7(a), with $k = 6$. As shown in Figure 7(b), two objects (the jellies and the background) are located accurately, while the mixing proportions of the other four extra Gaussians (objects) become less than 0.04. That is, the correct number of the objects could still be detected on this color image. Finally, the adaptive gradient learning algorithm was implemented on the color image of peppers, which is shown in Figure 8(a), with $k = 6$. As shown in Figure 8(b), the three objects are located accurately, with the mixing proportions of other three extra Gaussians reduced below 0.01.

As a result, the adaptive gradient learning algorithm can detect the number of actual objects in each of these color images. Moreover, the segmentation results of the adaptive gradient learning algorithm are much better than those of the generalized competitive clustering (GCC) algorithm [29] (based on the fuzzy clustering theory) given in the web http://www-rocq.inria.fr/~boujemaa/Partielle2.html.

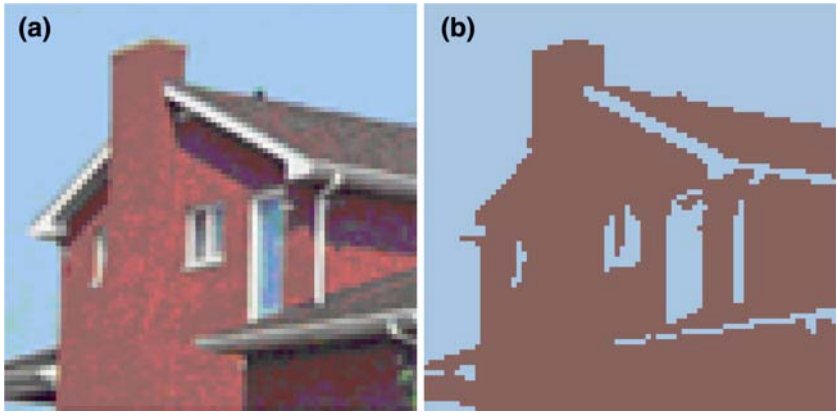

Figure 6. The segmentation result on the color image of house. (a) The original color image of house; (b) The segmented image of house via the adaptive gradient learning algorithm.

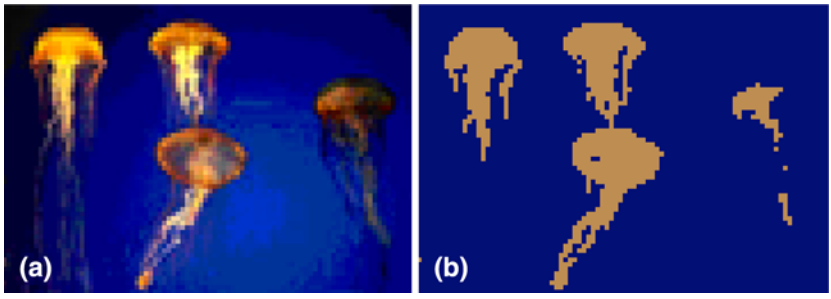

Figure 7. The segmentation result on the color image of jellies: (a) The original color image of jellies; (b) The segmented image of jellies via the adaptive gradient learning algorithm.

Figure 8. The segmentation result on the color image of peppers: (a) The original color image of peppers; (b) The segmented image of peppers via the adaptive gradient learning algorithm.

By comparison, we can easily find that the adaptive gradient learning algorithms lead to a more accurate segmentation on the contours of the objects in each image.

5. Conclusions

After introducing a BI-architecture of the BYY system and the harmony function, we have derived an adaptive gradient leaning algorithm to make parameter learning on finite mixture with automated model selection. It is shown by theoretic analysis that the adaptive gradient learning rule enforces a kind of floating RPCL mechanism such that for each input x_t, there is a threshold value $T(t)$ such that the components (or posterior probabilities) above $T(t)$ are rewarded to increase, while the components (or posterior probabilities) below $T(t)$ are penalized to decrease, simultaneously with a conscience mechanism on the mixing proportions. Moreover, the adaptive gradient learning algorithm is demonstrated well by the simulation and practical results on the sample data sets from Gaussian mixtures with a certain degree of overlap.

Acknowledgements

This work was supported by the Natural Science Foundation of China for Projects 60071004, 60471054. The authors thank Prof. Lei Xu for his strong support and helpful discussions, and Prof. Taijun Wang for his simulation supports.

Appendix

The Proof of Theorem 1 on the Three Properties on $\lambda_j(t)$

(i) According to Equation (9), we have

$$\begin{aligned}\lambda_j(t) &= 1 + lnU_j(x_t) - \sum_{i=1}^{k} p(i|x_t) lnU_i(x_t) \\ &= 1 + lnp(j|x_t) - \sum_{i=1}^{k} p(i|x_t) lnp(i|x_t) \\ &= 1 + lnp(j|x_t) + H(x_t), \end{aligned} \tag{25}$$

where

$$H(x_t) = H(p(1|x_t), \ldots, p(k|x_t)) = -\sum_{i=1}^{k} p(i|x_t) lnp(i|x_t) \tag{26}$$

is the Shannon entropy of the posterior probabilities $p(i|x_t)$ at the sample x_t. Since $H(x_t)$ is invariant with j and $lnp(j|x_t)$ increases with $p(j|x_t)$, we certainly have $\lambda_{j_1}(t) \leqslant \lambda_{j_2}(t) \leqslant \cdots \leqslant \lambda_{j_k}(t)$ if $p(j_1|x_t) \leqslant p(j_2|x_t) \leqslant \cdots \leqslant p(j_k|x_t)$. It also follows from Equation (25) that there exists a threshold value $T(t) = e^{-(1+H(x_t))}$ for $p(i|x_t)$ such that if $p(i|x_t) > (\leqslant) T(t)$, $\lambda_i(t) > (\leqslant) 0$.

(ii) If $j_c = \text{argmax} p(j|x_t)$, i.e., $p(j|x_t) \leqslant p(j_c|x_t)$ for any j, and thus $lnp(j|x_t) \leqslant lnp(j_c|x_t)$ for any j. Therefore, we have

$$\sum_{j=1}^{k} p(j|x_t) lnp(j|x_t) \leqslant \sum_{j=1}^{k} p(j|x_t) lnp(j_c|x_t) = lnp(j_c|x_t), \tag{27}$$

from which we further have

$$\lambda_{j_c}(x_t) \geqslant 1. \tag{28}$$

(iii) If $p(j|x_t) > \frac{1}{e}$, i.e., $ep(j|x_t) > 1$, then $ln[ep(j|x_t)] = 1 + lnp(j|x_t) > 0$. Since $H(x_t) \geqslant 0$, we have

$$\lambda_j(t) = 1 + lnp(j|x_t) + H(x_t) > 0. \tag{29}$$

If $p(j|x_t) \leqslant \frac{1}{ke}$, i.e., $ep(j|x_t) \leqslant \frac{1}{k}$, then $ln[ep(j|x_t)] = 1 + lnp(j|x_t) \leqslant -lnk$. By the property of Shannon entropy, $H(x_t) = H(p(1|x_t), \ldots, p(k|x_t)) \leqslant lnk$ with the equality only holding when each $p(i|x_t) = \frac{1}{k}$, we have

$$\lambda_j(x_t) = 1 + lnp(j|x_t) + H(x_t) < 0. \tag{30}$$

References

1. Mclachlan, G. J. and Basford, K. E.: *Mixture Models: Inference and Applications to Clustering*, Marcel Dekker, New York, 1988.
2. Devijver, P. A. and Kittter, J.: *Pattern Recognition: A Statistical Approach*, Prentice Hall, Englewood Cliffs, NJ, 1982.
3. Render, R. A. and Walker, H. F.: Mixture densities, maximum likelihood and the EM algorithm, *SIAM Review* **26**(2) (1984), 195–239.
4. Hartigan, J. A.: Distribution problems in clustering, In: J. Van Ryzin (ed), *Classification and Clustering*, pp. 45–72, Academic press, New York, 1977.
5. Akaike, H.: A new look at the statistical model identification, *IEEE Trans. on Automatic Control* **AC-19** (1974), 716–723.
6. Scharz, G.: Estimating the dimension of a model, *The Annals of Statistics*, **6** (1978), 461–464.

7. Bozdogan, H: Model selection and Akaike's information criterion: the general theory and its analytical extensions, *Psychometrika* **52** (1987), 345–370.
8. Ball, G. H. and Hall, D. J.: ISODATA: a novel method of data analysis and pattern classification, Technique Report No. 699616, 1965, Stanford Research International.
9. Makhoul, J., Rpucos, S. and Gish, H.: Vector quantization in speech coding, *Proceedings of IEEE* **73** (1985), 1551–1558.
10. Rumelhart, D. E. and Zipser, D.: Feature discovery by competitive learning, *Cognitive Sciences* **9** (1985), 75–112.
11. Ahalt, S. C., Krishnamurty, A. K., Chen, P. and Melton, D. E.: Competitive learning algorithm for vector quantization, *Neural Networks* **3** (1990), 277–291.
12. Xu, L., Krzyzak, A. and Oja, E.: Rival penalized competitive learning for clustering analysis, RBF net, and curve detection, *IEEE Transactions on Neural networks* **4** (1993), 636–648.
13. Xu, L.: Rival penalized competitive learning, finite mixture, and multisets clustering, In: *Proceedings of 1998 IEEE International Joint Conference On Neural Networks* **3** (1998), 251–2530.
14. Xu, L.: Ying-Yang machine: a Bayesian–Kullback scheme for unified learnings and new results on vector quantization, In: *Proceedings of the 1995 International Conference on Neural Information Processing (ICONIP'95)* **2** (1995), 977–988.
15. Xu, L.: Best harmony, unified RPCL and automated model selection for unsupervised and supervised learning on Gaussian mixtures, three-layer nets and ME-RBF-SVM models, *International Journal of Neural Systems* **11** (2001), 43–69.
16. Xu, L.: Ying-Yang learning, In: M. A. Arbib (2nd ed.), *The Handbook of Brain Theory and Neural Networks,* pp. 1231–1237. The MIT Press, Cambridge, MA, 2002.
17. Xu, L.: BYY harmony learning, structural RPCL, and topological self-organizing on mixture modes, *Neural Networks* **15** (2002), 1231–1237.
18. Xu, L.: Bayesian-Kullback YING-YANG learning scheme: reviews and new results, In: *Proceedings of the 1996 International Conference on Neural Information Processing (ICONIP'96)* **1** (1996) 59–67.
19. Hu, X. and Xu, L.: A comparative study of several cluster number selection criteria, In: *Lecture Notes of Computer Science,* vol. 2690 pp. 195–202, 2003.
20. Liu, Z. and Xu, L.: Smoothed local PCA by BYY data smoothing learning, In: *Proceedings of the International Conference on Control, Automation, and System (ICCAS'01)*, 621–622 (2001).
21. Ma, J., Wang, T. and Xu, L.: An annealing approach to BYY learning on Gaussian mixture with automated model selection, In: *Proceedings of 2003 International Conference on Neural Networks and Signal Processing (ICNN&SP'03)* **1** (2003), 23–28.
22. Ma, J., Wang, T., and Xu, L.: A gradient BYY harmony learning rule on Gaussian mixture with automated model selection, *Neurocomputing* **56** (2004), 481–487.
23. Ma, J., Gao, B., Wang, Y., and Cheng Q.: Conjugate and natural gradient rules for BYY harmony learning on Gaussian mixture with automated model selection, *International Journal of Pattern Recognition and Artificial Intelligence* **19** (2005), 701–713.
24. Robbins, H. and Monro, S.: A stochastic approximation method, *The Annals of Mathematical Statistics* **22** (1951), 400–407.
25. Robert, S. J., Everson, R., and Rezek, I.: Maximum certainty data partitioning, *Pattern Recognition* **33** (2000), 833–839.
26. Ueda, N. and Ghahramani, Z.: Bayesian model search for mixture models based on optimizing variational bounds, *Neural Network* **15** (2002), 1223–1241.

27. Blake, C. L. and Merz, C. J.: *UCI repository of machine learning databases* University of California, Irvine, Department of Information and Computer Science, 1998, http://www.ics.uci.edu/~mlearn/MLRepository.html.
28. Loo, C. K., and Rao, M. V. C.: Accurate and reliable diagnosis and classification using probablistic ensemble simplified fuzzy ARTMAP, *IEEE Transactions on Knowledge and Data engineering* **17** (2005), 1589–1593.
29. Boujeman, N.: Generalized competitive clustering for image segmentation, In: *Proceedings of 19th International Conference of the North American Fuzzy Information Processing Society*, 133–137 (2000).