EM Algorithms for Probabilistic Mapping Networks

Haizhou Li, Yifan Gong, Jean-Paul Haton

▶ **To cite this version:**

Haizhou Li, Yifan Gong, Jean-Paul Haton. EM Algorithms for Probabilistic Mapping Networks. [Research Report] RR-2614, INRIA. 1995, pp.39. inria-00074071

HAL Id: inria-00074071
https://inria.hal.science/inria-00074071

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

EM Algorithms for Probabilistic Mapping Networks

Haizhou Li, Yifan Gong, Jean-Paul Haton

N° 2614

Juillet 1995

PROGRAMME 3

ISSN 0249-6399

EM Algorithms for Probabilistic Mapping Networks

Haizhou Li, Yifan Gong, Jean-Paul Haton

Programme 3 — Intelligence artificielle, systèmes cognitifs
et interaction homme-machine
Projet SYCO

Rapport de recherche n2614 — Juillet 1995 — 39 pages

Abstract: The Expectation-Maximization (EM) algorithm is a general technique for maximum likelihood estimation (MLE). In this paper we present several of the important theoretical and practical issues associated with Gaussian mixture modeling (GMM) within the EM framework. First, we propose an EM algorithm for estimating the parameters of a special GMM structure, named a probablistic mapping network (PMN), where the Gaussian probability density function is realized as an internal node. In this way, the EM algorithm is extended to deal with the supervised learning of a multicategory classification problem and serve as a parameter estimator of the neural network Gaussian classifier. Then, a generalized EM (GEM) algorithm is developed as an alternative to the MLE problem of PMN. This is followed by a discussion on the computational considerations and algorithmic comparisons. It is shown that GEM converges faster than EM to the same solution space. The computational efficiency and the numerical stability of the training algorithm benefit from the well-established EM framework. The effectiveness of the proposed PMN architecture and developed EM algorithms are assessed by conducting a set of speaker recognition experiments.

Key-words: EM algorithm, maximum likelihood classifier, probabilistic mapping network, speaker recognition

◇ Haizhou LI is on leave from South China University of Technology, Canton, China

Unité de recherche INRIA Lorraine
Technôpole de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY (France)
Téléphone : (33) 83 59 30 30 – Télécopie : (33) 83 27 83 19
Antenne de Metz, technopôle de Metz 2000, 4 rue Marconi, 55070 METZ
Téléphone : (33) 87 20 35 00 – Télécopie : (33) 87 76 39 77

(Résumé : tsvp)

Apprentissage de Réseaux Probabilistes par Algorithmes EM

Résumé : L'algorithme "Expectation-Maximisation" (EM) est une technique générale pour l'estimation par maximum de vraisemblance. Ce document présente quelques aspects théoriques et pratiques importants liés à la modélisation par mélanges de gaussienne (GMM) dans le cadre EM. Nous proposons d'abord un algorithme EM pour estimer le paramètres d'un modèle GMM particulier, appelé PMN ("probabilistic mapping network"), dans lequel la fonction de densité de probabilité gaussienne est calculée par un noeud interne du réseau. Nons présentons ensuite un algorithm EM généralisé (GEM) comme une solution alternative à l'estimation par maximum de vraisemblance d'un PMN. Cette présentation est complétée par une discussion sur les aspects calculatoires comparés des deux algorithmes EM et GEM. Nous montrons en particulier que GEM converge plus vite que EM vers le même espace de solutions. L'efficacité pratique de l'architecture PMN et des algorithmes EM proposés est évaluée à travers un ensemble d'exemples pratiques dans le domaine de la reconnaissance de locuteurs.

Mots-clé : Algorithms EM, classification par maximum de vraisemblance, réseau probabiliste PMN, reconnaissance de locuteurs

1 Introduction

The EM framework was first proposed by Dempster *et al* [1] as a general technique for maximum likelihood estimation. For a long time, EM has been applied in practice almost exclusively to unsupervised learning problems. One of the most successful applications is the Baum-Welch reestimation formulas[2] in HMM speech recognition. Some others appear in the neural network literature, where EM is involved in context clustering[3] and density estimation[4], as well as in the statistics literatures, in which applications include missing data problem[5] and factor analysis[1]. The EM approach to the classification problem has been explored recently in neural network research, where the EM algorithm is developed for some probabilistic neural networks, such as hierarchical mixtures of expert[6] and generalized Fisher training probabilistic neural networks[7].

Probabilistic neural networks is a research topic which is currently given serious consideration in classifier design for several reasons. First, it inherits the properties of a general neural network classifier. For example, with its massive and adaptive architecture, it possesses better robustness or fault tolerance, better learning capability and graceful degradation. (cf. Wu[8] and Lippmann [9] for good reviews of the field) Second, the use of a probability density function (pdf) as the nonlinear transformation enables the explicit interpretation of the performance of a neural network by the well-established probability theory, in which some stochastic approaches can be incorporated into the network learning and evaluation.

As pointed out by Lee[10], nonsigmoidal activation functions as well as multiple hidden layers might provide the network with the capability of forming more complex nonlinear manifolds of the input domain than sigmoidal ones through intermediate mappings. Some probabilistic neural networks, such as the Gaussian potential function network(GPFN)[11], the probabilistic neural network(PNN)[4], the probabilistic mapping networks (PMN) [7, 8] and the Gaussian clustering network(GCN)[12], have been developed using Gaussian-like nonlinear functions and appealed to our interests in classical pattern classification applications. However, there are no stochastic constraints imposed to the mixture weightings between the Gaussian basis units and the output units in the GPFN, so the network output of the GPFN could not be consi-

dered as a probability count. Given a prescribed criterion, the classifier solves for the decision boundaries rather than creating a probabilistic model. The PNN[4] is analogous to a Parzen window in statistical pattern recognition. For each training pattern, it creates a corresponding radial basis unit in the hidden layer. All basis units share the same reception field width σ, thus called homoscedastic, which is determined according to some heuristic knowledge. Since no effort is made to evaluate the capacity of the network and remove the redundant training data, a large number of basis units will be involved for accurate density estimation of a distribution. Both the GCN and the PMN approximate the probability densities by a *semi-parametric* estimation method with Gaussian kernels[12]. The PMN in [8] learns the model with MLE criterion, and gives an integrated method to learn all network parameters in one training phase for multiclass density estimation problems. However, the learning rule is derived by means of the principle of stochastic gradient descent searching, Some other studies of neural network classifier with maximum likelihood learning were reported in [8, 13], but apparently without recognition of the possible realization of the practical and elegant EM algorithm.

The remainder of this paper is organized as follows. In Section 2, the architecture of probabilistic mapping network is described. In Section 3, the EM algorithm for PMN is formulated. The existence of an EM solution is also proved. In Section 4, a generalized EM algorithm is proposed as an alternative to solving for the EM problem discussed in Section 2. In Sections 5 and 6, numerical evaluations of the algorithms are given by conducting a set of speaker recognition experiments. Finally, some remarks on the possible extensions of the theory are presented.

2 Probabilistic mapping network

The probabilistic mapping network is designed with a Gaussian probability function (pdf) to compute the class *a posteriori* probability for a given sample. To this end, all computational components in the expression of the class *a posteriori* probability should have their proper interpretations in the network. In this section, the notation and the network architecture are presented.

2.1 Bayesian classifier with Gaussian kernel

Let

$$\mathcal{X} = \{\mathcal{X}_i\}_{i=1}^{I} \quad (1)$$
$$\mathcal{X}_i = \{\mathbf{x}_{ik}\}_{k=1}^{T_i} \quad (2)$$

where $\sum_{i=1}^{I} T_i = T$, be a set of data for I mutually exclusive classes. Suppose that the D-dimensional observation vector $\mathbf{x}_{ik}$ is the k-th realization of the class i. Assume that the realization can be generated by one of J_i random sources, with the probabilities of $Pr(\omega_{ij})$, the class-conditional probability can be given as

$$p(\mathbf{x}_{ik}|\Omega_i) = \sum_{j=1}^{J_i} p(\mathbf{x}_{ik}|\omega_{ij}) Pr(\omega_{ij}) \quad (3)$$

where $p(\mathbf{x}_{ik}|\omega_{ij})$ is called the component density and $Pr(\omega_{ij})$ the mixture coefficient. In this paper, we have $p(\mathbf{x}_{ik}|\omega_{ij}) = N(\mathbf{x}_{ik}; \mathbf{m}_{ij}, \Sigma_{ij})$ as the Gaussian density and $Pr(\omega_{ij})$ is denoted as β_{ij}, subject to

$$\sum_{j=1}^{J_i} \beta_{ij} = 1 \qquad \beta_{ij} > 0 \quad (4)$$

Considering the multicategory classification problem of I classes, the Bayes decision rule is applied to find the class with the maximum *a posteriori* probability by

$$p(\Omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\Omega_i) Pr(\Omega_i)}{\sum_{s=1}^{I} p(\mathbf{x}|\Omega_s) Pr(\Omega_s)} \quad (5)$$
$$= \frac{p(\mathbf{x}, \Omega_i)}{p(\mathbf{x})} \quad (6)$$
$$\mathbf{x} \in \Omega_i \quad \text{if} \quad i = \arg\max_s Pr(\Omega_s|\mathbf{x}) \quad (7)$$

where $Pr(\Omega_i)$, denoted as α_i, is subject to the constraints,

$$\sum_{i=1}^{I} \alpha_i = 1 \qquad \alpha_i > 0 \quad (8)$$

The parameters of the Gaussian mixture model (GMM) can be collectively represented by the notation:

$$\Lambda_{ij} = \{\mathbf{m}_{ij}, \Sigma_{ij}, \beta_{ij}\} \quad (9)$$

$$\Lambda_i = \{\Lambda_{ij}\}_{j=1}^{J_i} \quad (10)$$

$$\Lambda = \{\Lambda_i, \alpha_i\}_{i=1}^{I} \quad (11)$$

Λ along with the Bayes decision rule defines a Gaussian classifier. Building the Gaussian classifier, we can train the models $\{\Lambda_i, \alpha_i\}$ independently by class since all the classes are mutually exclusive.

Tying of parameters is a practical technique usually used when the amount of training data is limited[14], which also reduces the number of free parameters to be estimated. There are many ways to tie the parameters in a classifier. A tying scheme, called the probabilistic mapping network (PMN) was proposed by Wu[8] and shown to be successful in a speech recognition application. The PMN along with two other tying schemes were also discussed recently by Li[15]. The PMN can be summarized as follows: Suppose the number of mixtures is equal for all classes, $\{J_i = J, \forall i\}$. All the mixtures from different classes can be tied into a shared Gaussian kernel by letting $\{\mathbf{m}_{ij} = \mathbf{m}_j, \forall i\}$, $\{\Sigma_{ij} = \Sigma_j, \forall i\}$. To provide an insight into the above definition, a four-layer feedforward neural network is depicted in Figure 1. Since all classes share a same Gaussian mixture kernel, $\{m_j, \Sigma_j\}$, in which each mixture component has its contribution to the likelihood of all classes, the only parameters to discriminate among classes are the class dependent mixture coefficients $\{\beta_{ij}\}$. The parameter Λ_{ij} is thus redefined as

$$\Lambda_{ij} = \{\mathbf{m}_j, \Sigma_j, \beta_{ij}\} = \{\omega_j, \beta_{ij}\} \quad (12)$$

and Λ consists of three parameter sets

$$\alpha = \{\alpha_i\}_{i=1}^{I} \quad (13)$$

$$\beta = \{\{\beta_{ij}\}_{i=1}^{I}\}_{j=1}^{J} \quad (14)$$

$$\omega = \{\omega_j\}_{j=1}^{J} \quad (15)$$

It is obvious that, when the parameters are tied in this way, the training of the Gaussian kernel in Λ, $\Lambda \in \Omega$ where Ω is the parameter space, is concerned with the data from multiple classes. For simplicity, in this paper, class i is

referred to by Λ_i, the parameter set, instead of Ω_i from now on. For example, we use $p(\mathbf{x}_{ik}|\Lambda_i)$ rather than $p(\mathbf{x}_{ik}|\Omega_i)$ and $p(\Lambda_i|\mathbf{x}_{ik})$ rather than $p(\Omega_i|\mathbf{x}_{ik})$.

2.2 Probabilistic mapping network: PMN

Speaking in terms of neural networks, a Gaussian probability density function (pdf) is interpreted as a neuron in the Gaussian kernel, namely the internal layer of a neural network. It is known that a linear combination of Gaussian basis functions is capable of representing a large class of distributions. The use of a Gaussian node is motivated by this understanding. The classifier defined in the previous subsection gives an neural network structure, the PMN, which is a four-layer feedforward network. The first layer is the input layer to which the sample data are presented. Thus, the D-dimensional input vector requires D input nodes in the first layer. The mixture pdfs, $p(\mathbf{x}|\omega_{ij})$, are computed as the outputs of the second layer. The connections between the second and third layers are associated with the mixture coefficients β_{ij}. Each node in the third layer performs the summation of the weighted mixture probabilities to produce the class-conditional probability. Each node in the third and fourth layers corresponds to a class. As a result, I nodes are required there. During the learning phase, the connections between the third and fourth layers, the class *a priori* probabilities, are trained. In the classification phase, the fourth layer performs the computation of Eq.(5) with the help of the lateral connections to apply the Bayes decision rule.

A maximum likelihood (ML) learning algorithm for PMN was proposed by Wu[8] where the general optimization procedure of gradient descent is used to maximize the overall likelihood. It is known that the convergence of such optimization procedures needs some heuristic control and often leads to numerical problems in implementation. In the next two sections, an EM algorithm and its variant, the generalized EM, are proposed for PMN learning. The EM algorithm is a general iterative technique for maximum likelihood estimation. It has been shown to be less complex than the conventional optimization procedures[1].

3 The EM algorithm for PMN learning

In the multicategory classification problem, the Gaussian mixture models approach builds a model to represent a class of data in a parametric way. Different learning strategies may lead to different system performance. The EM algorithm aims at maximizing the overall likelihood of training data presented to the estimated classifier. Within the framework of EM, the sample data $\mathbf{x}_{ik}$ is observable and is called *incomplete data*. To develop an EM algorithm for the PMN architecture, we should define appropriate *missing data* so as to give the likelihood function of *complete data*. To this end, an indicator variable y_{ik} associated with each sample data $\mathbf{x}_{ik}$ is defined, such that $y_{ik} \in \mathcal{J}$, $\mathcal{J}$ is the set of integers between 1 and J. These indicator variables have an interpretation as the identity of the unobservable component density in the Gaussian kernel. It is possible to specify a new data item by combining $\mathbf{x}_{ik}$ with y_{ik}, $(\mathbf{x}_{ik}, y_{ik})$, which becomes the *complete data* in the statement of an EM problem. y_{ik} is a random variable which is to be involved in the unsupervised learning of the Gaussian kernel.

3.1 An introduction to EM

Let $p(\mathbf{x}_{ik}|\Lambda)$ and $p(y_{ik}|\Lambda)$ be members of a parametric family of the pdf defined on $\mathcal{X}$ and $\mathcal{J}$ respectively for parameter Λ. For a given $\mathbf{x}_{ik} \in \mathcal{X}$, the EM algorithm is to maximize the log-likelihood of the observable, *incomplete data* $\mathbf{x}_{ik}$, $L(\mathbf{x}_{ik}|\Lambda) = \log p(\mathbf{x}_{ik}|\Lambda)$ over Λ with the help of a relationship between $p(\mathbf{x}_{ik}, y_{ik}|\Lambda)$ and $p(y_{ik}|\mathbf{x}_{ik}, \Lambda)$. Writing in the form of log-likelihood, we have

$$\log p(\mathbf{x}_{ik}|\Lambda) = \log p(\mathbf{x}_{ik}, y_{ik}|\Lambda) - \log p(y_{ik}|\mathbf{x}_{ik}, \Lambda) \tag{16}$$

It is the relationship between these two likelihood functions that motivates the EM algorithm. Note that the complete data likelihood is a random variable, because the missing variables y_{ik} are in fact unknown. For two parameter set Λ' and Λ, taking the expected value of Eq.(16) conditioned by $\mathbf{x}_{ik}$ and Λ' over the *missing data* y_{ik}, the following expression is obtained[2]:

$$L(\mathbf{x}_{ik}|\Lambda) = Q_{ik}(\Lambda|\Lambda') - H_{ik}(\Lambda|\Lambda') \tag{17}$$

Where

$$\begin{aligned} Q_{ik}(\Lambda|\Lambda') &= E[\log p(\mathbf{x}_{ik}, y_{ik}|\Lambda)|\mathbf{x}_{ik}, \Lambda'] \\ &= \sum_{y_{ik}\in\mathcal{J}} p(y_{ik}|\mathbf{x}_{ik}, \Lambda') \log p(\mathbf{x}_{ik}, y_{ik}|\Lambda) \end{aligned} \tag{18}$$

$$H_{ik}(\Lambda|\Lambda') = E[\log p(y_{ik}|\mathbf{x}_{ik}, \Lambda)|\mathbf{x}_{ik}, \Lambda'] \tag{19}$$

EM is an iterative approach where each iteration is made up of two steps: the *Expectation* step and the *Maximization* step. In E-step, we compute $Q(\Lambda|\Lambda')$ given Λ'. The M-step involves the maximization of a likelihood that is redefined at each iteration by the E-step. The iteration is repeated until the Q-function converges.

The basis of the EM algorithm lies in the fact that if $Q(\Lambda|\Lambda') \geq Q(\Lambda'|\Lambda')$, then $L(\mathbf{x}_{ik}|\Lambda) \geq L(\mathbf{x}_{ik}|\Lambda')$, since it follows from Jensen's inequality that $H(\Lambda|\Lambda') \leq H(\Lambda'|\Lambda')$[16]. This fact guarantees the monotonic increase of $L(\mathbf{x}_{ik}|\Lambda)$ by updating the parameter from Λ' to Λ at each iteration. The EM algorithm is used in applications which permit easy maximization of the Q-function instead of maximizing $L(\mathbf{x}_{ik}|\Lambda)$ directly. In practice this implies the convergence to a local maximum.

3.2 The EM formulation

The likelihood function of the labeled training data $\mathbf{x}_{ik}$, the kth sample in class i, is given as

$$\begin{aligned} L(\mathbf{x}_{ik}|\Lambda) &= \log \sum_{i'=1}^{I} [\alpha_{i'} p(\mathbf{x}_{i'k}|\Lambda_{i'})\delta(i'-i)] \\ &= \log [\alpha_i p(\mathbf{x}_{ik}|\Lambda_i)] \end{aligned} \tag{20}$$

where $\delta(.)$ is the Kronecker delta function. For the training set $\mathcal{X}$, Eq.(20) is extended to

$$L(\mathcal{X}|\Lambda) = \log \left[\prod_{i=1}^{I} \prod_{k=1}^{T_i} \alpha_i p(\mathbf{x}_{ik}|\Lambda_i) \right] \tag{21}$$

It it noted that α_i should not be zero and depends only on class identity i. Substituting Eq.(3) in Eq.(20) yields

$$p(\mathbf{x}_{ik}|\Lambda) = \alpha_i \sum_{j=1}^{J} \beta_{ij} p(\mathbf{x}_{ik}|\omega_j) \tag{22}$$

Following the interpretation of missing data, the conditional pdf for a *complete data* item $(\mathbf{x}_{ik}, y_{ik})$ is represented as the probability contribution of the coupled component in the model:

$$p(\mathbf{x}_{ik}, y_{ik} = j|\Lambda) = \alpha_i \beta_{ij} p(\mathbf{x}_{ik}|\omega_j) \tag{23}$$

It is straightforward to verify that

$$\sum_{y_{ik} \in \mathcal{J}} p(\mathbf{x}_{ik}, y_{ik}|\Lambda) = p(\mathbf{x}_{ik}|\Lambda) \tag{24}$$

The M-step of the EM method is defined so as to maximize the Q-function, which takes the expectation over the *incomplete data* y_{ik}. The *missing data* is here a discrete random variable. Eq.(18) is the written as

$$Q_{ik}(\Lambda|\Lambda') = \sum_{y_{ik} \in \mathcal{J}} p(y_{ik}|\mathbf{x}_{ik}, \Lambda') \log p(\mathbf{x}_{ik}, y_{ik}|\Lambda) \tag{25}$$

By using Eq.(22) and Eq.(23), we obtain

$$\begin{aligned} p(y_{ik} = j|\mathbf{x}_{ik}, \Lambda) &= \frac{p(\mathbf{x}_{ik}, y_{ik} = j|\Lambda)}{p(\mathbf{x}_{ik}|\Lambda)} \\ &= \frac{\alpha_i \beta_{ij} p(\mathbf{x}_{ik}|\omega_j)}{\alpha_i \sum_{j=1}^{J} \beta_{ij} p(\mathbf{x}_{ik}|\omega_j)} \\ &= p(\omega_j|\mathbf{x}_{ik}) \end{aligned} \tag{26}$$

Similarly,

$$p(y_{ik} = j|\mathbf{x}_{ik}, \Lambda') = p(\omega'_j|\mathbf{x}_{ik}) \tag{27}$$

Therefore, for data set $\mathcal{X}$,

$$Q(\Lambda|\Lambda') = \sum_{i=1}^{I} \sum_{k=1}^{T_i} \sum_{y_{ik} \in \mathcal{J}} p(y_{ik}|\mathbf{x}_{ik}, \Lambda') \log[\alpha_i \beta_{ij} p(\mathbf{x}_{ik}|\omega_j)] \tag{28}$$

$$
\begin{aligned}
&= \sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{k=1}^{T_i} p(y_{ik}=j|\mathbf{x}_{ik},\Lambda')\log[\alpha_i\beta_{ij}p(\mathbf{x}_{ik}|\omega_j)] \\
&= \sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{k=1}^{T_i} p(\omega_j'|\mathbf{x}_{ik})\log\alpha_i \qquad (29)\\
&\quad + \sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{k=1}^{T_i} p(\omega_j'|\mathbf{x}_{ik})\log\beta_{ij} \\
&\quad + \sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{k=1}^{T_i} p(\omega_j'|\mathbf{x}_{ik})\log p(\mathbf{x}_{ik}|\omega_j)
\end{aligned}
$$

Note that $p(y_{ik}|\mathbf{x}_{ik},\Lambda)$ is a conditional probability function of the *missing data* y_{ik} subject to

$$\sum_{j=1}^{J} p(y_{ik}=j|\mathbf{x}_{ik},\Lambda)=1 \qquad (30)$$

Thus, the first term of Eq.(29) becomes

$$
\begin{aligned}
&\sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{k=1}^{T_i} p(y_{ik}=j|\mathbf{x}_{ik},\Lambda')\log\alpha_i \qquad (31)\\
&= \sum_{i=1}^{I} T_i\log\alpha_i. \qquad (32)
\end{aligned}
$$

The M-step of the EM method maximizes the Q-function with respect to the parameter set Λ given the previous estimate Λ'. Examining the Q-function, it is found that the elements of the parameter sets $\{\alpha,\beta,\omega\}$ are decoupled from each other. The Q-function could be rewritten as

$$
\begin{aligned}
Q(\Lambda|\Lambda') &= \sum_{i=1}^{I} T_i\log\alpha_i + \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{T_i} p(\omega_j'|\mathbf{x}_{ik})\log\beta_{ij} \\
&\quad + \sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{k=1}^{T_i} p(\omega_j'|\mathbf{x}_{ik})\log p(\mathbf{x}_{ik}|\omega_j) \\
&= Q_\alpha + \sum_{i=1}^{I} Q_\beta^i + \sum_{j=1}^{J} Q_\omega^j \qquad (33)
\end{aligned}
$$

To solve the maximization problem, Q_α, Q_β^i and Q_ω^j can be maximized separately. With the help of the Lagrange multipliers, one can introduce the parameter constraints of Eq.(4) and Eq.(8) into the optimization problem. Λ is solved by differentiating the Lagrangian with respect to each parameter in the set, setting the partial derivatives equal to zero. Starting with the parameter set $\{\alpha_i\}$, the Lagrangian for $\{\alpha_i\}$ is

$$\overline{Q}_\alpha = Q_\alpha + \gamma \left(\sum_{i=1}^{I} \alpha_i - 1 \right) \tag{34}$$

$\{\alpha_i\}$ is obtained, which leads to the unique global maximum of Q_α[16]

$$\alpha_i = \frac{T_i}{T} \tag{35}$$

is the frequency count of occurrence of patterns from class Ω_i, generally, understood as the maximum likelihood estimation of the class *a priori* probability[17]. Similarly, the appropriate Lagrangian for $\{\beta_{ij}\}$ is followed as

$$\overline{Q}_\beta = Q_\beta^i + \gamma \left(\sum_{j=1}^{J} \beta_{ij} - 1 \right) \tag{36}$$

Then $\{\beta_{ij}\}$ is obtained, which also leads to the unique global maximum of Q_β^i[16]

$$\beta_{ij} = \frac{1}{T_i} \sum_{k=1}^{T_i} p(\omega_j' | \mathbf{x}_{ik}) \tag{37}$$

A Gaussian pdf ω_j in the kernel is defined by two parameters, $\mathbf{m}_j$ and Σ_j. They are solved by equating the partial derivative of the Q_ω^j to zero. For $\mathbf{m}_j$,

$$\begin{aligned} \frac{\partial Q_\omega^j}{\partial \mathbf{m}_j} &= \sum_{i=1}^{I} \sum_{k=1}^{T_i} p(\omega_j' | \mathbf{x}_{ik}) \frac{\partial \log p(\mathbf{x}_{ik} | \omega_j)}{\partial \mathbf{m}_j} \\ &= \sum_{i=1}^{I} \sum_{k=1}^{T_i} p(\omega_j' | \mathbf{x}_{ik}) \Sigma_j^{-1} (\mathbf{x}_{ik} - \mathbf{m}_j) \\ &= \mathbf{0} \end{aligned} \tag{38}$$

In the same way, for Σ_j,

$$\frac{\partial Q_\omega^j}{\partial \Sigma_j} = \sum_{i=1}^{I}\sum_{k=1}^{T_i} p(\omega_j'|\mathbf{x}_{ik}) \frac{\partial \log p(\mathbf{x}_{ik}|\omega_j)}{\partial \Sigma_j} \tag{39}$$

$$= \sum_{i=1}^{I}\sum_{k=1}^{T_i} \frac{1}{2} p(\omega_j'|\mathbf{x}_{ik}) \left[\Sigma_j^{-1}(\mathbf{x}_{ik}-\mathbf{m}_j)(\mathbf{x}_{ik}-\mathbf{m}_j)^t\Sigma_j^{-1} - \Sigma_j^{-1}\right] \tag{40}$$

$$= \mathbf{0} \tag{41}$$

Finally, we have solved for an estimation of mean vector in the pdf ω_j,

$$\mathbf{m}_j = \frac{\sum_{i=1}^{I}\sum_{k=1}^{T_i} p(\omega_j'|\mathbf{x}_{ik})\mathbf{x}_{ik}}{\sum_{i=1}^{I}\sum_{k=1}^{T_i} p(\omega_j'|\mathbf{x}_{ik})} \tag{42}$$

and the estimation of the related covariance matrix is found as

$$\Sigma_j = \frac{\sum_{i=1}^{I}\sum_{k=1}^{T_i} p(\omega_j'|\mathbf{x}_{ik})(\mathbf{x}_{ik}-\mathbf{m}_j)(\mathbf{x}_{ik}-\mathbf{m}_j)^t}{\sum_{i=1}^{I}\sum_{k=1}^{T_i} p(\omega_j'|\mathbf{x}_{ik})} \tag{43}$$

3.3 Proof of the EM procedure

The Q-function is one of the main mathematical foundations of the EM procedure. From the concavity of the log function in the Gaussian-based model , it follows that

Theorem 3.1 *If*

$$Q(\Lambda|\Lambda') \geq Q(\Lambda'|\Lambda') \tag{44}$$

then

$$L(\mathcal{X}|\Lambda) \geq L(\mathcal{X}|\Lambda') \tag{45}$$

A parameter Λ is a critical point of the likelihood $L(\mathcal{X}|\Lambda')$ if and only if it is a critical point of Q-function.

Proof See reference [16], Section 2, Lemma 1. □

The theorem implies that if the new parameter set Λ that satisfies the condition of Eq.(44) can be found, the model reestimation algorithm can be

guaranteed to improve $L(\mathcal{X}|\Lambda)$. Next, for the Q-function defined in this paper, the proof of the existence of Λ is necessary, which ensures that the EM procedure in section 3.2 is well defined. As described by Liporace[18], the proof involves three arguments. 1) The second derivative of Q-function along any direction in the parameter space is strictly negative at a critical point. This implies that each critical point is a local maximum. 2) $Q(\Lambda|\Lambda') \to -\infty$ as Λ approaches the finite boundary of the parameter space or the point at ∞. This property implies that the global maximum is a critical point. 3) The critical point is unique. Because of the independence of parameters in the three addends of Eq.(25), the maximization of Q-function is conducted separately by addend terms. The existence of α_i and β_{ij} can be proved to be the unique global maximum of Q_α and Q^i_β by Lemma 2 in [16]. Here, the proof is given only for the parameter set $\omega_j = \{\mathbf{m}_j, \Sigma_j\}$.

Theorem 3.2 *For $D+1$ samples in the training set $\mathcal{X}$, if any D samples are linearly independent and if the* a priori *probability $\alpha_i > 0$ for all i and the mixture weighting coefficients $\beta_{ij} > 0$ for all ij, then Q^j_ω has a unique global maximum as a function of the covariance matrix Σ_j and of the mean vector $\mathbf{m}_j$.*

Proof From Eq.(43), it is evident that the linearly independent condition on training set $\mathcal{X}$, $\mathcal{X}$ is full rank, is introduced so that a positive definitive Σ_j could be obtained. Because of pooling samples across multiple classes, the D independent samples may come from different classes. Let us examine a special case where $T = I \geq D$, i.e.: we have only one sample for each class, and the independent condition is even satisfied. This implies that fewer training data is expected than the learning problem without tying, in order to conduct the EM procedure. The advantage benefits from the tying involved. Even so, it is a rare case in practice, for instance, in the speaker recognition problem.

Recall the expression of jth summand of Q^j_ω in Eq.(33). Reparameterizing Q^j_ω by putting $C_j = \Sigma_j^{-1}$, we obtain

$$\begin{aligned} Q^j_\omega &= \sum_{i=1}^{I}\sum_{k=1}^{T_i} p(\omega'_j|\mathbf{x}_{ik}) \log p(\mathbf{x}_{ik}|\omega_j) \qquad (46) \\ &= \frac{1}{2}\sum_{i=1}^{I}\sum_{k=1}^{T_i} p(\omega'_j|\mathbf{x}_{ik}) \left[\log|C_j| - D\log(2\pi) - (\mathbf{x}_{ik}-\mathbf{m}_j)^t C_j (\mathbf{x}_{ik}-\mathbf{m}_j)\right] \end{aligned}$$

Letting $\eta_{ik}^j = p(\omega_j'|\mathbf{x}_{ik})$ and suppressing the dependence of the right hand side on j for simplicity, Eq.(46) becomes

$$Q_\omega^j = \frac{1}{2}\sum_{i=1}^{I}\sum_{k=1}^{T_i}\eta_{ik}\left[\log|C| - D\log(2\pi) - (\mathbf{x}_{ik}-\mathbf{m})^t C(\mathbf{x}_{ik}-\mathbf{m})\right] \tag{47}$$

where $\eta_{ik} > 0$. It is not straightforward to give the second derivative of Q_ω^j with respect to Σ and $\mathbf{m}$ in an explicit form. Hence, some auxiliary variables are introduced. Let $\overline{\omega}$ be the solution of Eq.(42) and Eq.(43) for a critical point of Q_ω^j. We now express $\overline{\omega}$ as a convex combination of two arbitrary points ω^1 and ω^2 by putting in Eq.(47) $C = \theta C^1 + (1-\theta)C^2$, $\mathbf{m} = \theta\mathbf{m}^1 + (1-\theta)\mathbf{m}^2$, where $0 < \theta < 1$ and $\omega^1 \neq \omega^2$, $\omega^1 \neq \overline{\omega}$, $\omega^2 \neq \overline{\omega}$. Then, differentiating twice with respect to θ yields

$$\begin{aligned}
\frac{\partial^2 Q_\omega^j}{\partial\theta^2} &= -\frac{1}{2}\sum_{i=1}^{I}\sum_{k=1}^{T_i}\eta_{ik}\left\{\sum_{d=1}^{D}\left[\frac{\kappa^1-\kappa^2}{\theta\kappa^1-(1-\theta)\kappa^2}\right]^2\right. \\
&\quad -2(\mathbf{m}^1-\mathbf{m}^2)^t[\theta C^1+(1-\theta)C^2](\mathbf{m}^1-\mathbf{m}^2) \\
&\quad \left.-4(\mathbf{m}^1-\mathbf{m}^2)^t(C^1-C^2)[\mathbf{x}_{ik}-\theta\mathbf{m}^1-(1-\theta)\mathbf{m}^2)]\right\} \qquad (48)\\
&= -\frac{1}{2}\sum_{i=1}^{I}\sum_{k=1}^{T_i}\eta_{ik}\left\{R_1+R_2+R_3\right\} \qquad (49)
\end{aligned}$$

where κ^1 and κ^2 are the diagonal entries of UC^1U^t and UC^2U^t respectively, and U is the orthogonal matrix diagonalizing $\theta C^1 + (1-\theta)C^2$, or diagonalizing C^1 and C^2 simultaneously.

$$\begin{aligned}
R_1 &= \sum_{d=1}^{D}\left[\frac{\kappa^1-\kappa^2}{\theta\kappa^1-(1-\theta)\kappa^2}\right]^2 \qquad (50)\\
R_2 &= 2(\mathbf{m}^1-\mathbf{m}^2)^t C(\mathbf{m}^1-\mathbf{m}^2) \qquad (51)\\
R_3 &= 4(\mathbf{m}^1-\mathbf{m}^2)^t(C^1-C^2)(\mathbf{x}_{ik}-\mathbf{m}) \qquad (52)
\end{aligned}$$

It should be noted that $R_1 > 0$ if $\omega^1 \neq \omega^2$. Since Σ is positive definitive, $R_2 > 0$ if $\mathbf{m^1} \neq \mathbf{m^2}$. Rewrite Eq.(38) at the critical point $\overline{\omega}$

$$\begin{aligned}
\frac{\partial Q_\omega^j}{\partial\mathbf{m}_j} &= C\sum_{i=1}^{I}\sum_{k=1}^{T_i}\eta_{ik}(\mathbf{x}_{ik}-\mathbf{m}_j) \qquad (53)\\
&= \mathbf{0}
\end{aligned}$$

Therefore, the third term of the summation in Eq.(49) vanishes at the critical point

$$\sum_{i=1}^{I}\sum_{k=1}^{T_i}\eta_{ik}R_3 = 0 \tag{54}$$

We can conclude that $\partial^2 Q_\omega^j/\partial\theta^2 < 0$ is strict, independently of ω^1 and ω^2, i.e. along any direction. The proofs of the other two arguments can be found in [18]. This theorem guarantees what is estimated by Eq.(42), Eq.(43) is the global maximum of Q_ω^j. □

3.4 Some notes on EM algorithm

In the problem of *class separate* GMM learning[14, 19], the class dependent parameters are trained independently by class. The EM is said to train the models unsupervisedly. Some numerical examples can be found in [1, 14]. If the parameters are tied in certain ways across the classes such that they should be trained under the supervision of class identity, then the EM algorithm can be regarded as involving a supervised learning across classes and an unsupervised learning within class as formulated in Section 3.2.

By examining the reestimation formulas of Eq.(35), Eq.(37), Eq.(42) and Eq.(43), it is noted that the class *a priori* probability $\{\alpha_i\}$ is computed by Eq.(35) without the need of iteration. It can be decoupled and computed independently of other parameters in Q-function because the class label is a supervised variable which does not depend on the data distribution. All other parameters are updated in an iterative way until convergence. Reestimating the component weighting coefficients $\{\beta_{ij}\}$, only the samples in the concerned class are involved in the computation since $\{\beta_{ij}\}$ are class dependent. Finally, the parameters in the Gaussian kernel are updated based on the entire data set $\mathcal{X}$. Obviously, this is the immediate consequence of tying the Gaussian kernel.

It is of interest to examine the differences between the proposed EM algorithm and the conventional one for the untied GMM[19]. The real change lies in the differences of the model structure where EM algorithm works. Consider a case where no tying of parameters is assumed, we have a GMM for each class, resulting in a conventional unsupervised learning problem, which would be solved independently of each other by class using an usual EM approach[1].

Hence, it is shown that the EM algorithm for PMN learning is reduced to the conventional one by assuming that each class has its own Gaussian mixture.

Another probabilistic neural network (PNN) with EM learning was proposed in [7], where the covariance matrices are tied within each class mixture and also across all classes, resulting in a homoscedastic problem. It is the common covariance matrix which laterally links the individual class into an unified model structure, a neural network. One should note that by omitting the tying of the covariance matrix, the PNN in [7] then reduces to the conventional EM problem as well.

The tying of parameters provides the distributed computation and memory structure, which explains why we call the PMN a neural network. For later algorithmic comparisons, the EM algorithm proposed is summarized as follows. From now on, all the bracketed superscripts in the text denote the iteration index.

Algorithm EM

1. **Initialization:** Compute the class *a priori* probability $\{\alpha_i\}$ by Eq.(35). Set $c = 1$. Choose an initial estimation $\Lambda^{[c]}$.

2. **E-step:** Compute $Q(\Lambda|\Lambda^{[c]})$ by Eq.(29) given $\Lambda^{[c]}$.

3. **M-step:** Solve for $\Lambda^{[c+1]}$ such that

$$\Lambda^{[c+1]} = \arg\max_{\Omega} Q(\Lambda|\Lambda^{[c]}) \tag{55}$$

   by Eq.(37), Eq.(42) and Eq.(43). Given in the form of the updating rules

$$\beta_{ij}^{[c+1]} = \frac{1}{T_i}\sum_{k=1}^{T_i}\eta_{ik}^{j[c]} \qquad \forall i,j \tag{56}$$

$$\mathbf{m}_j^{[c+1]} = \frac{\sum_{i=1}^{I}\sum_{k=1}^{T_i}\eta_{ik}^{j[c]}\mathbf{x}_{ik}}{\sum_{i=1}^{I}\sum_{k=1}^{T_i}\eta_{ik}^{j[c]}} \qquad \forall j \tag{57}$$

$$\Sigma_j^{[c+1]} = \frac{\sum_{i=1}^{I}\sum_{k=1}^{T_i}\eta_{ik}^{j[c]}(\mathbf{x}_{ik}-\mathbf{m}_j^{[c+1]})(\mathbf{x}_{ik}-\mathbf{m}_j^{[c+1]})^t}{\sum_{i=1}^{I}\sum_{k=1}^{T_i}\eta_{ik}^{j[c]}} \qquad \forall j \tag{58}$$

4. **Evaluation:** Set $c = c + 1$, repeat from step 2 until convergence.

Although the maximization of Q-function gives an unique global maximum estimation as proved in Section 3.3, the EM algorithm does not guarantee the solution point is a global maximum of the likelihood $L(\mathcal{X}|\Lambda)$. In consequence, the algorithm is usually applied several times with different initializations. Among the several solutions obtained, the Λ which yields the largest $L(\mathcal{X}|\Lambda)$ is selected. Next we propose a generalized EM (GEM) algorithm which updates the Λ more frequently than the the EM scheme. The GEM algorithm is proposed as an alternative to the learning problem of PMN, and it is expected to outperform its EM counterpart in convergence performance.

4 The generalized EM algorithm

This section discusses two important algorithmic issues of EM. A generalized EM (GEM) algorithm is firstly proposed, which leads to an efficient procedure in practice. The proof of the convergence theorem of GEM is also given. This is followed by a theorem which convinces that both GEM and EM converge to the same solution space. The GEM algorithm can be considered as a revised version of the EM one.

4.1 The GEM formulation

In the EM algorithm, the M-step involves the maximization of a likelihood function that is redefined by the E-step. In general[1], a GEM algorithm was defined as an algorithm that simply increases the Q-function during the M-step rather than maximizing the function. Here, a GEM scheme is introduced where Λ is only partially updated in a EM iteration. To be more specific, only ω_j (cf. Eq.(12)) is reestimated at Λ each step. This implies that the Q-function will be evaluated J times before the Gaussian kernel is totally updated. If one regards the EM algorithm as a batch learning of Gaussian pdfs, GEM allows us to learn the pdfs sequentially.

Definition 4.1 *Since only one pdf in the Gaussian kernel is reestimated in each GEM iteration, Λ requires J iterations to get entirely updated. Let the iterations be indexed by p, $1 \leq p \leq J$, p corresponds to a pdf identity. A Q-function of GEM is defined to have a similar form as Eq.(33). For simplicity,*

two notations are used, $\eta_{ik}^{j} = p(\omega_j|\mathbf{x}_{ik})$ *and* $\xi_{ik}^{j} = p(\mathbf{x}_{ik}|\omega_j)$.

$$\begin{aligned} Q(\Lambda|\Lambda^{(p)}) &= \sum_{i=1}^{I} T_i \log \alpha_i + \left\{ \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{T_i} \eta_{ik}^{j(p)} \log \beta_{ij} \right. \\ &\quad \left. + \sum_{i=1}^{I}\sum_{k=1}^{T_i} \eta_{ik}^{p(p)} \log \xi_{ik}^{p} \right\} + \sum_{i=1}^{I}\sum_{j,j\neq p}\sum_{k=1}^{T_i} \eta_{ik}^{j(p)} \log \xi_{ik}^{j(p)} \\ &= Q_\alpha + \left\{ \sum_{i=1}^{I} Q_\beta^i + Q_\omega^p \right\} + \sum_{j,j\neq p} Q_\omega^j \end{aligned} \tag{59}$$

$Q(\Lambda|\Lambda^{(p)})$ *is thus a function of* ξ_{ik}^{p} *and* $\{\beta_{ij}\}$, $\{\omega_p, \beta\}$, *which appears in the braces.*

Definition 4.1 gives the formula for computing expectation in E-step. To derive the M-step of GEM, we define a Q-function by noting that only one pdf in the Gaussian kernel will be updated.

Definition 4.2 *A mapping* M *is defined as* $\Lambda^{(p+1)} = M(\Lambda^{(p)})$ *if only* $\{\omega_p, \beta\}$ *is updated in* $\Lambda^{(p)}$ *such that*

$$\Lambda^{(p+1)} = \arg\max_{\Omega} Q(\Lambda|\Lambda^{(p)}) \tag{60}$$

Since the Λ needs J mappings to be completely updated for the data set $\mathcal{X}$, these J mappings are named an epoch. In each epoch, we have a sequence of points $\Lambda^{(1)}$, ...,$\Lambda^{(p)}$,...,$\Lambda^{(J)}$. An instance in (p)-th iteration for [c]-th GEM epoch is laterly denoted as $\Lambda^{[c](p)}$ when both the epoch and iteration indices need to be referred. One epoch can be regarded as an analog of one EM iteration in the EM algorithm for easier comparison.

The statement of GEM is similar to that of EM. Thus, the notations in EM are used here unless redefinition is necessary. Q_α is left alone since it can be computed prior to the iteration. Therefore, in GEM procedure, Eq.(35) remains unchanged. According to the definition, it is evident that the third term in Eq.(59) is a constant too in the M-step indexed p. Following the derivation in Section 3.2, the mapping defined in definition 4.2 can be specified

as follows:

$$\beta_{ij}^{(p+1)} = \frac{1}{T_i}\sum_{k=1}^{T_i}\eta_{ik}^{j(p)} \qquad \forall i,j \tag{61}$$

$$\mathbf{m}_p^{(p+1)} = \frac{\sum_{i=1}^{I}\sum_{k=1}^{T_i}\eta_{ik}^{j(p)}\mathbf{x}_{ik}}{\sum_{i=1}^{I}\sum_{k=1}^{T_i}\eta_{ik}^{j(p)}} \tag{62}$$

$$\Sigma_p^{(p+1)} = \frac{\sum_{i=1}^{I}\sum_{k=1}^{T_i}\eta_{ik}^{j(p)}(\mathbf{x}_{ik}-\mathbf{m}_p^{(p+1)})(\mathbf{x}_{ik}-\mathbf{m}_p^{(p+1)})^t}{\sum_{i=1}^{I}\sum_{k=1}^{T_i}\eta_{ik}^{j(p)}} \tag{63}$$

Algorithm GEM

1. **Initialization:** Compute the class *a priori* probability $\{\alpha_i\}$ by Eq.(35). Set $c=1$, $p=0$. Choose an initial estimation $\Lambda^{[c](p)}$.
2. **E-step:** Compute $Q(\Lambda|\Lambda^{[c](p)})$ by Eq.(59) given $\Lambda^{[c](p)}$.
3. **M-step:** Solve for $\Lambda^{[c](p+1)}$ which maximizes $Q(\Lambda|\Lambda^{[c](p)})$ by Eq.(61), Eq.(62) and Eq.(63). If $p<J$, then $p=p+1$, go to step 2, otherwise go to step 4.
4. **Evaluation:** If $\left|\Lambda^{[c](J)}-\Lambda^{[c](0)}\right|<\mu$, $\mu>0$, stop with $\Lambda^{[c](J)}$, otherwise set $p=0$, $\Lambda^{[c+1](0)}=\Lambda^{[c](J)}$, $c=c+1$ and go to step 2.

In GEM, two levels of iterations are involved. The outer level is indexed by c and the inner level is indexed by p. In each inner iteration J M-mappings are conducted. An outer iteration is carried out when an epoch of inner iteration is completed. The convergence of GEM is also evaluated during the outer iterations.

The difference between EM and GEM lies in the fact that EM requires the maximization of the Q-function over the entire Gaussian kernel while GEM allows us to update pdfs in the kernel one by one. In comparison with EM, the outer iteration of GEM is similar to the iteration indexed by c in EM, and an epoch of GEM corresponds to a pair of E-step and M-step in EM.

Next we prove the convergence theorem of GEM to show that the GEM is well-defined in each step. We also demonstrate a theorem which proves that GEM converges to the same solution space as EM.

4.2 Convergence theorem of GEM

The proof of the convergence theorem is in the spirit of Dempster's proof[1] of a similar result.

Lemma 4.1 *In consequence of the definition 4.1 and 4.2, for each mapping in GEM* $\Lambda^{(p+1)} = M(\Lambda^{(p)})$*, we have*

$$Q(\Lambda^{(p+1)}|\Lambda^{(p)}) \geq Q(\Lambda^{(p)}|\Lambda^{(p)}) \tag{64}$$

where M denotes the mapping defined in the definition 4.2

Lemma 4.2 *If* $Q(\Lambda^{(p+1)}|\Lambda^{(p)}) \geq Q(\Lambda^{(p)}|\Lambda^{(p)})$ *then* $L(\mathcal{X}|\Lambda^{(p+1)}) \geq L(\mathcal{X}|\Lambda^{(p)})$ *where* $L(\mathcal{X}|\Lambda^{(p)})$ *is the likelihood of* $\mathcal{X}$ *conditioned on* $\Lambda^{(p)}$*. A parameter* Λ *is a critical point of the likelihood* $L(\mathcal{X}|\Lambda^{(p)})$ *if and only if it is a fixed point of the Q-function.*

Proof See reference [1], Section 3, Theorem 1. □

The convergence of a similar algorithms has been discussed in various texts [1, 18] and is summarized next for the sake of keeping the proof of GEM self-contained.

Theorem 4.1 *Suppose* $\Lambda^{(p')}$*,* $p' = c \times J + p$*, is an instance of the GEM algorithm such that*

$$\nabla_{\Lambda^{(p')}} Q(\Lambda^{(p'+1)}|\Lambda^{(p')}) = 0 \tag{65}$$

Then for each p'*, there exists a* $\Lambda_0^{(p'+1)}$ *on the line segment joining* $\Lambda^{(p'+1)}$ *to* $\Lambda^{(p')}$ *such that*

$$Q(\Lambda^{(p'+1)}|\Lambda^{(p')}) - Q(\Lambda^{(p')}|\Lambda^{(p')}) = -(\Lambda^{(p'+1)} - \Lambda^{(p')}) \nabla^2_{\Lambda^{(p')}} Q(\Lambda_0^{(p'+1)}|\Lambda^{(p')}) (\Lambda^{(p'+1)} - \Lambda^{(p')})^t \tag{66}$$

Furthermore, if the sequence $\nabla^2 Q(\Lambda^{(p'+1)}|\Lambda^{(p')})$ *is negative definite and* $L(\mathcal{X}|\Lambda^{(p')})$ *is bounded, then the sequence* $\Lambda^{(p')}$ *converges to some* Λ^* *in the closure of* Ω*.*

Proof See reference [1], Section 3, Theorem 3. □

Lemma 4.3 *In the GEM algorithm,* $\Lambda^{(p')}$ *converges to some* Λ^* *in the closure of* Ω*.*

Proof Taking the second derivative of Q_α and Q_β^j in Eq.(59) with respect to α_i and β_{ij} respectively, considering the constraints on α_i and β_{ij}, it is easy to show that

$$\frac{\partial^2 Q_\alpha}{\partial \alpha_i^2} = -\frac{T_i}{\alpha_i^2} < 0 \tag{67}$$

$$\frac{\partial^2 Q_\beta^i}{\partial \beta_{ij}^2} = -\frac{\sum_{k=1}^{T_i} \eta_{ik}^j}{\beta_{ij}^2} < 0 \tag{68}$$

Comparing Eq.(33) with Eq.(59), it is noted that Q_ω^j in Eq.(33) is decomposed into two terms in Eq.(59). Only Q_ω^p in the braces of Eq.(59) is reestimated in (p')-th iteration. Following the proof of Theorem 3.2, let $\omega^1 = \omega^{(p'+1)}$ and $\omega^2 = \omega^{(p')}$. It can be shown that $\partial^2 Q_\omega^p / \partial \theta^2 < 0$ for all θ. Hence, $\Lambda^{(p')}$ converges according to Theorem 4.1. □

Theorem 4.2 *Suppose that $\Lambda^{(p')}$, $p' = c \times J + p$, is an instance of a GEM algorithm such that (a) $\Lambda^{(p')}$ converges to Λ^* in the closure of Ω; (b) $\nabla_\Lambda Q(\Lambda|\Lambda^{(p')}) = 0$; (c) $\nabla_\Lambda^2 Q(\Lambda|\Lambda^{(p')})$ is negative definite, then Λ^* is one of the solutions to the maximum likelihood estimation such that*

$$\nabla_\Lambda L(\mathcal{X}|\Lambda^*) = 0. \tag{69}$$

Proof See reference [1], Section 3, Theorem 4. □

To show that the Theorem 4.2 is applicable in the GEM, we need to prove that the three conditions are satisfied.

Lemma 4.4 *The procedure defined in the GEM algorithm converges to a maximum likelihood estimation solution Λ^* when p' is large enough.*

Proof The first condition is proved to be satisfied by Lemma 4.3. The second condition is the immediate consequence of definition 4.2 which is realized by Eq.(35), Eq.(61), Eq.(62) and Eq.(63). When proving the third condition is satisfied, it is not straightforward to give an explicit expression. Following the proof of Theorem 3.2 by introducing a convex combination of $\Lambda^{(p'+1)}$ and $\Lambda^{(p')}$, it is easy to show that $\nabla_\theta^2 Q(\Lambda|\Lambda^{(p')})$ is strictly negative and the third condition is thus satisfied. □

Lemma 4.4 ascertains that GEM algorithm converges to a maximum likelihood estimation. Finally, let us investigate the equivalence relation between GEM and EM.

Definition 4.3 *Let $O_g \subset \Omega$ be the set of all $\Lambda_g^{[c]}$ at critical points of $L(\mathcal{X}|\Lambda_g^{[c]})$. O_g will be called the solution set of GEM problem. (Note that the subscript g is only used to discriminate the sequence of GEM from that of EM)*

Definition 4.4 *Let $O \subset \Omega$ be the set of all $\Lambda^{[c]}$ at critical points of $L(\mathcal{X}|\Lambda^{[c]})$. O will be called the solution set of EM problem.*

Theorem 4.3 *If $\Lambda_g^{[c](p)} \in O_g$ at the (p)-th iteration of [c]-th epoch for GEM algorithm, then $\Lambda_g^{[c](p)}$ is also a solution to the EM problem, or $\Lambda_g^{[c](p)} \in O$.*

Proof Since $\Lambda_g^{[c](p)} \in O_g$ is a critical point in the [c]-th epoch of GEM, we have $\Lambda_g^{[c+1](0)} = \Lambda_g^{[c](J)} = \Lambda_g^{[c](p)}$ once the [c]-th epoch is completed. Similarly, in the $[c+1]$-th epoch,

$$\Lambda_g^{[c+1](1)} = M(\Lambda_g^{[c+1](0)}) = \Lambda_g^{[c+1](0)} \tag{70}$$

$$\dots$$

$$\Lambda_g^{[c+1](J)} = M(\Lambda_g^{[c+1](J-1)}) = \Lambda_g^{[c+1](J-1)} \tag{71}$$

furthermore,

$$L(\Lambda_g^{[c+1](0)}) = L(\Lambda_g^{[c+1](1)}) = \ldots = L(\Lambda_g^{[c+1](J)}) \tag{72}$$

Let a convergent instance $\Lambda_g^{[c+1](0)} \in O_g$ be a parameter estimate for EM algorithm at [c']-th iteration, $\Lambda^{[c']} = \Lambda_g^{[c+1](0)}$. Reestimate $\Lambda^{[c']}$ by applying Eq.(37), Eq.(42) and Eq.(43) as M-step of EM iteration. According to Eq.(70),Eq.(71) and to the definition of M-mapping, in the course of EM iteration, we have

$$\Lambda^{[c'+1]} = \Lambda^{[c']} \tag{73}$$

$$L(\Lambda^{[c'+1]}) = L(\Lambda^{[c']}) \tag{74}$$

Recalling Theorems 3.1 and 3.2, $\Lambda^{[c']}$ converges to a point in O. In other words, if $\Lambda_g^{(p)}$ converges to a point in O_g then it converges to a point in O. □

Theorem 4.3 implies that the GEM algorithm converges to the same solution space as EM. It has been shown that GEM gives a new approach to the maximum likelihood solution of PMN as stipulated by the Theorem 4.3. Therefore, one can expect that GEM has the same classification performance as EM.

5 Implementation issues

Before proceeding to the numerical examples, some implementation issues are addressed which will help us deal with the computation in a real computer.

It is known that GEM reestimates the parameter set $\{\beta_{ij}\}$ more frequently than EM. It might appear that GEM introduces more computation than EM in the inner iteration. It should be recalled that the computation of the mixture pdf on the kernel $\{\omega_j\}$ is the most time-consuming. The mapping in the inner iteration can be simplified, since only the Gaussian pdf concerning the changed ω_j needs to be recomputed, while all others remain the same, in other words, the kernel of $\Lambda^{(p+1)}$ differs from $\Lambda^{(p)}$ only by one pdf.

Let us examine the pdf computation in the GEM algorithm. According to the statement, at the (p)-th E-step, to recompute the Q-function Eq.(59), only the pdf of ξ_{ik}^p in

$$\eta_{ik}^p = \frac{\beta_{ip}\xi_{ik}^p}{\sum_{p'=1}^J \beta_{ip'}\xi_{ik}^{p'}} \tag{75}$$

is required to be updated for sample $\mathbf{x}_{ik}$. Thus, the total number of pdf computations in a GEM epoch is the same as that in an EM iteration. The extra computation introduced in GEM remains in the J times reevaluation of Q-function and M-mapping where only the operations of weighted normalization are involved. By contrast, faster convergence is expected in the numerical evaluation due to the frequent updating of $\{\beta_{ij}\}$. By examining the Eq.(43) and Eq.(63), we note that the computation of covariance relies on the reestimated mean vector. In practice, Eq.(43) is inconvenient to implement. However, it can be expressed as

$$\Sigma_j = \frac{\sum_{i=1}^I \sum_{k=1}^{T_i} p(\omega_j'|\mathbf{x}_{ik})\mathbf{x}_{ik}\mathbf{x}_{ik}^t}{\sum_{i=1}^I \sum_{k=1}^{T_i} p(\omega_j'|\mathbf{x}_{ik})} - \mathbf{m}_j\mathbf{m}_j^t \qquad \forall j \tag{76}$$

and similarly, Eq.(76) is applicable to Eq.(63) by letting $j = p$. This implies that the first term of Eq.(76) can be computed along with the computation of $\mathbf{m}_j$, Eq.(42), and saves us from the repetition of scanning over all $p(\omega_j'|\mathbf{x}_{ik})$.

We have dealt primarily with the theory of the algorithms. It might appear that the solution to the GEM or EM problems can be obtained by straightforward translation of the relevant formulas into computer programs. Unfortunately, in reality, the training data from different classes are not collected

with equal quantities. Because of the tying of Gaussian mixture, all the classes share the same Gaussian kernel in a PMN. For a class with relative small quantity of samples, it might occur that some of the weighting coefficients $\{\beta_{ij}\}$ result in underflow on any real computer since fewer mixtures might be sufficient to describe the data distribution properly. Although it is possible for a local maximum of Λ to lie on a boundary of the parameter manifold, it is not tractable in practice. To cope with the problem, we maximize Q_β^i subject to the new boundary constraints

$$\beta_{ij} \geq \epsilon > 0. \tag{77}$$

EM and GEM formulations can be slightly modified by building the new constraint into the derivation. Since $\{\beta_{ij}\}$ is a disjoint set from other parameters in Λ, it suffices to show how Eq.(37) is modified[16]. According to the Lemma 2 in [16], Q_β^i attains its unique global maximum when Eq.(37) holds. Suppose that there are q weighting coefficients falling outside the constraint of Eq.(77)

$$\beta_{ij}^{(p+1)} = \frac{1}{T_i} \sum_{k=1}^{T_i} \eta_{ik}^{j(p)} \begin{cases} \geq \epsilon & \text{for } j = 1, \ldots, J-q \\ < \epsilon & \text{for } j = J-q+1, \ldots, J \end{cases} \tag{78}$$

Let

$$\beta_{ij}^{(p+1)} = \epsilon \qquad \text{for } j = J-q+1, \ldots, J \tag{79}$$

We come up with the problem of maximizing

$$Q_\beta^i = \sum_{j=1}^{J-q} \sum_{k=1}^{T_i} \eta_{ik}^{j(p)} \log \beta_{ij} \tag{80}$$

subject to the constraint

$$\sum_{j=1}^{J-q} \beta_{ij} = 1 - q\epsilon \tag{81}$$

Following the derivation of Lemma 2 in [16], Eq.(37) is modified as

$$\beta_{ij}^{(p+1)} = \frac{1-q\epsilon}{T_i} \sum_{k=1}^{T_i} \eta_{ik}^{j(p)} \qquad \text{for } j = 1, \ldots, J-q \tag{82}$$

If some new values of β_{ij} do not satisfy the constraint, the reestimation of β_{ij} continues and q increases until no violation ocurrs. The maximization of Q^i_β is obtained on the boundary of parameter space specified by the Eq.(77) when the constraint is violated.

6 Applications to speaker recognition

Several classification examples will now be presented to illustrate the effectiveness of the PMN and their EM and GEM algorithms, where each class in the PMN represents the voice of a speaker. We examine the convergence and classification aspects for text-dependent speaker recognition. In speaker recognition, the test speaker is supposed to be inside the known speaker group, or design group. The trained model makes a recognition according to the test data. The experiments use realistic speech data in a practical system[20].

The capacity of the PMN is an important issue. Three convergence examples are given, the convergence of a defined PMN structure is observed with respect to different speaker populations, the number of classes. In the recognition assessment, the model trained with respect to different speaker populations are also evaluated. Another critical factor concerned is the number of mixtures. To explore the effects of different settings of J, a set of experiments was designed to examine the performance with respect to three different values of J for speaker recognition tasks. The Gaussian kernels in all the experiments are initialized by LBG[21] algorithm. For comparison between the EM and GEM algorithms, same initializations are used.

6.1 Practical considerations

The Gaussian pdf is introduced as a probabilistic function unit in the second layer of the feedforward network with output:

$$p(\mathbf{x}|\omega) = N(\mathbf{x};\mathbf{m},\Sigma) \tag{83}$$

$$= \frac{1}{(2\pi)^{D/2}\,|\Sigma|^{1/2}} \exp[-\frac{1}{2}(\mathbf{x}-\mathbf{m})^t\Sigma^{-1}(\mathbf{x}-\mathbf{m})] \tag{84}$$

Instead of using the general form of $\Sigma = \{\sigma_{pq}\}$, the full covariance matrix, it is possible to use a simpler but a more restricted form, the diagonal covariance

matrix where $\sigma_{pq} = 0$ for $p \neq q$. It implies that the elements in the input space are assumed to be uncorrelated. Supposing that a sufficient number of mixtures are allowed to act together to model the overall pdf of a class. Full covariance matrices are not necessary, even if the features are not statistically independent. The linear combination of diagonal covariance Gaussian is capable of modeling the correlations between feature vector elements. The performance of a full covariance Gaussian can be equally obtained by using a large enough set of diagonal covariance Gaussian[22].

In this specific application, all samples in a test utterance are pooled together to give the test data set. Because of the time dynamics of speech, the relative abundance of sample data for each class does not reflect the class *a priori* probability α_i, with which a speaker presents. Consequently, we assume equal α_i for all speakers.

6.2 Experimental setup

It is reasonable to assume that the acoustic space corresponding to a speaker's voice can be characterized by a set of acoustic classes representing some implicit phonetic events, such as vowels, nasals or consonants[23]. These acoustic classes are described by the GMM and reflect some general speaker-dependent vocal source and vocal tract configurations that are discriminative among speakers[15, 22, 24].

A database of French sentences from 60 speakers, 30 males and 30 females, is used[25], in which a three-word sentence[1] of approximately one second was naturally uttered five times in separate sessions by each speaker. The silences and stops in the utterance were eliminated since they are not discriminative. A 12th mel-scale cepstral analysis, performed each 10ms interval with a 32ms Hamming window, was used as the input to the PMN. In the convergence experiment, all the five sessions from a speaker were put together into a class to examine the convergences of PMN with EM and GEM algorithm. In the speaker recognition experiment, using one of five sessions as test data and others as training data, rotating the orders, resulted in five assessment sets. The results are reported for the five tests. The experiments were conducted in this way

[1] *"Cent beaux papis"*

as a similar strategy of tolerance intervals assessment[7] for classification or recognition performance.

6.3 Convergence behavior

The convergence of two algorithms has been proved in the previous sections. As pointed out above, EM and GEM converge to the same solution space. However, it is of interest to have an idea about how fast the convergence is. From the statement of algorithms, two convergent sequences, the Q-function and $L(\mathcal{X}|\Lambda)$ indicate the progress of the convergence. In light of Theorem 3.1, it is expected that the sequence of $Q(\Lambda^{[c+1]}|\Lambda^{[c]})$ increases to a stable point; $L(\mathcal{X}|\Lambda)$ increases to a local maximum on convergence. The convergence criteria require a relative increase of 10^{-4} in the likelihood function $L(\mathcal{X}|\Lambda)$, i.e., the iteration stops when

$$\frac{L(\mathcal{X}|\Lambda^{[c+1]}) - L(\mathcal{X}|\Lambda^{[c]})}{L(\mathcal{X}|\Lambda^{[c+1]})} < 10^{-4} \tag{85}$$

It should be noted that although GEM and EM converge to the same solution space, it is not guaranteed that they converge to the same solution because only a local maximum is expected in both algorithms. Hence, $\Delta Q^{[c+1]}$ was illustrated to show the speed of convergence and $L(\mathcal{X}|\Lambda)$ to investigate how well a local maximum is found. A larger $L(\mathcal{X}|\Lambda)$ implies a solution nearer to the global maximum. Three examples are given where 40, 50 and 60 speakers are involved respectively. The Gaussian kernel recruits 32 pdfs, or 32 Gaussian nodes. The curves are referred to as EM and GEM recorded in the first 50 iterations of convergence as given in Figure 2-7. It was reported that example 1 converged in 48 iterations for GEM and 56 iterations for EM; in example 2, 62 for GEM and 87 for EM; in example 3, 80 iterations for GEM and 110 for EM. In general, GEM converges faster than EM does. From the plots of convergent sequences, we note that GEM has solved for better local maxima than EM has done in the sense of log-likelihood $L(\mathcal{X}|\Lambda)$.

6.4 Recognition behavior

For speaker recognition purpose, a decision is made by using the maximum *a posteriori* decoding. To test a data set $\mathcal{X}' = \{\mathbf{x}_k\}_{k=1}^{T'}$, the overall *a posteriori*

probability for class s is computed as

$$p(\Lambda_i|\mathcal{X}') = \frac{\alpha_i p(\mathcal{X}'|\Lambda_i)}{\sum_{s=1}^{I} \alpha_s p(\mathcal{X}'|\Lambda_s)} \tag{86}$$

where

$$p(\mathcal{X}'|\Lambda_i) = \prod_{\mathbf{x}_k \in \mathcal{X}'} p(\mathbf{x}_k|\Lambda_i) \tag{87}$$

and a decision is made as

$$\mathcal{X}' \in \Omega_i \qquad \text{if } i = \arg\max_s p(\Lambda_s|\mathcal{X}') \tag{88}$$

which is reduced to

$$\mathcal{X}' \in \Omega_i \qquad \text{if } i = \arg\max_s p(\mathcal{X}'|\Lambda_s) \tag{89}$$

as α_i are assumed equal.

Two experiments are given to illustrate the classification performance. Experiment 1 trains on the data sets used in the convergence examples, but the test data is an independent session from the training sessions such that the resultant performance will not be optimistically biased. Five assessments and the average results for three examples are listed in Table 1. Experiment 2 trains three PMNs on the data set of 50 speakers, which give three examples showing the performances of PMN with respect to different J settings, $J = 32$, 64 and 128. Five assessments and the average results for three examples are listed in Table 2.

No significant difference is observed in the classification performances between GEM and EM as expected. The results above are reported as the performances of those trained by GEM. The recognition or classification performances are evaluated when the PMN converges.

The *close test* refers to the test on training data and *open test* refers to the test using an independent session. The examples in this section give convincing evidence of the utility of EM and GEM algorithms in PMN learning. We got 100% of recognition accuracy in all close tests. This implies that the PMN is well-trained in all cases. In experiment 1, the fact is confirmed that higher accuracy is obtained when a smaller population is involved. In experiment 2, it is shown that $J = 128$ performs the best for discriminating 50 speakers.

7 Concluding remarks

In this paper, we have developed two algorithms for PMN learning. The convergence theorems guarantee that the solutions of EM and GEM algorithms are well-defined for the maximum likelihood classifier. For the same PMN structure, the GEM is shown to outperform its EM counterpart in convergence due to more frequent updatings of β in the convergent process. It has also been shown that the computation in an epoch of GEM inner iteration is almost the same as a couple of EM steps in the EM procedure. To understand why an epoch converges faster than an EM couple, consider the (p)-th instance in the $[c]$-th epoch. When updating ω_p, β_{ij} have already been updated for $p-1$ times to be in accord with the $p-1$ previous pdfs, $\omega_1, \ldots, \omega_{p-1}$, so that the updating of ω_p could be based on the information of the updated pdfs. In general, this enables the GEM to learn with a smaller step size and tune in more precisely, especially when a large number of samples and classes are involved.

It is accepted that the EM scheme avoids the numerical problems when compared to the general optimization procedure, the gradient descent algorithm. The elegance of the algorithms benefits from the well-established EM procedure and the well-defined Gaussian family pdf.

The algorithms derived in this paper easily accommodate several useful extensions in statistical pattern recognition. Four of them are summarized briefly here.

Firstly, when considering the PMN as an one-state HMM[26], it is of interest to note that the generalization to HMM can be made by defining a new type of HMM with sharing Gaussian kernel, the semi-continuous HMM[14]. The semi-continuous HMM is especially suitable for the cases where the speech for a certain context or a certain speaker is considered to be generated or represented by an unique feature source, the Gaussian kernel, which covers well the vocal source and vocal tract features of the speech concerned. The proposed GEM algorithm is applicable to semi-continuous HMM training. To this end, the derivations given in this paper can straightforwardly be extended.

Secondly, as discussed in Section 3.4, it is the tying of parameters which links the multiple GMM into an unified network. The PMN we proposed gives an example of tying parameters across classes. Besides, it is evident that there will exist many other tying schemes in practice, as in some of our previous

studies[15], where the EM algorithm is also applicable. The model specific EM procedure could be derived in a similar way as stated above in this paper.

Thirdly, the proposed EM and GEM iterations can be made adaptive by introducing Bayesian learning when the class pdfs are time varying. This extension requires reformulating the likelihood function and Q-function with a Bayesian prior distribution of the parameter set Λ[1, 27]. The extension leads to a Bayesian learning for maximum *a posteriori* estimation of Gaussian mixture models or Gaussian Markov chains. This is potentially useful in applications where class statistics are either non-stationary, or treated as non-stationary to ensure robustness and parameter smoothing[27]. Examples can be found in speaker adaptation for speech recognition and time-drifting adaptation for speaker recognition.

Finally, it should be pointed out that PMN and its algorithms can be extended to mixtures of discrete pdfs and exponential family of pdfs[1]. The discrete pdfs are usually adopted when nonparametric estimation is required while the non-Gaussian pdfs will be a more reasonable modeling in certain applications.

The PMN architecture, its learning algorithm and the extensions provide us with a new framework within which a kind of multicategory classifiers can be trained efficiently and well-interpreted in probability theory.

Acknowledgement
The authors would like to thank Dr Richard Washington and Dr Chorkin Chan for the helpful discussions during the preparation of this manuscript that improved the quality of the document.

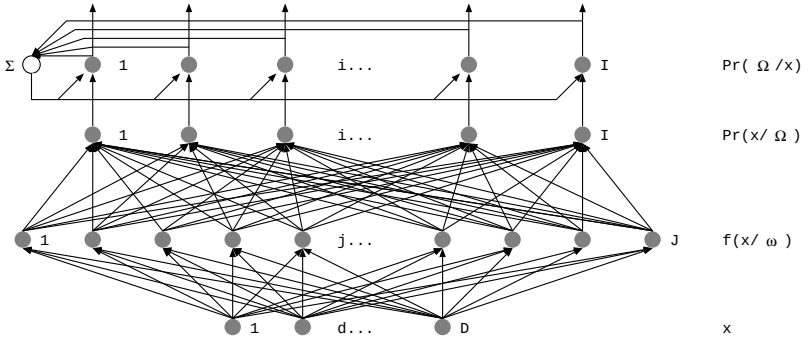

Figure 1: An architecture for PMN with Gaussian kernel

I	Test 1	Test 2	Test 3	Test 4	Test 5	Average
40	90.0	95.0	95.0	95.0	90.0	93.0
50	82.0	94.0	96.0	96.0	92.0	92.0
60	93.3	95.0	93.3	90.0	86.7	91.7

Table 1: Speaker recognition accuracy (%) for PMN with 32 pdfs, $J = 32$ (open test)

J	Test 1	Test 2	Test 3	Test 4	Test 5	Average
32	82.0	94.0	96.0	96.0	92.0	92.0
64	96.0	98.0	98.0	94.0	90.0	95.2
128	96.0	100.0	98.0	100.0	100.0	98.8

Table 2: Speaker recognition accuracy (%) for 50 speakers, $I = 50$ (open test)

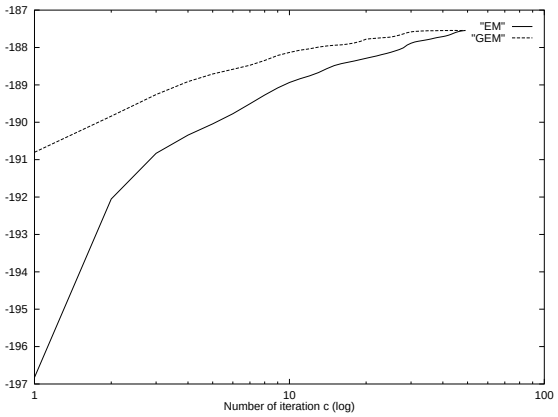

Figure 2: Example 1: Convergent sequences of $Q(\Lambda^{[c+1]}|\Lambda^{[c]})$ $(I = 40, J = 32)$

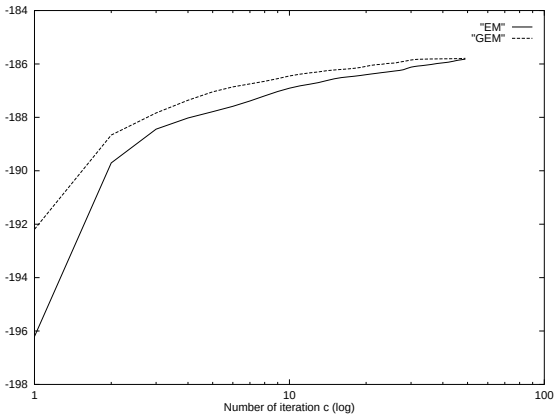

Figure 3: Example 1: Convergent sequences of $L(\mathcal{X}|\Lambda)$ $(I = 40, J = 32)$

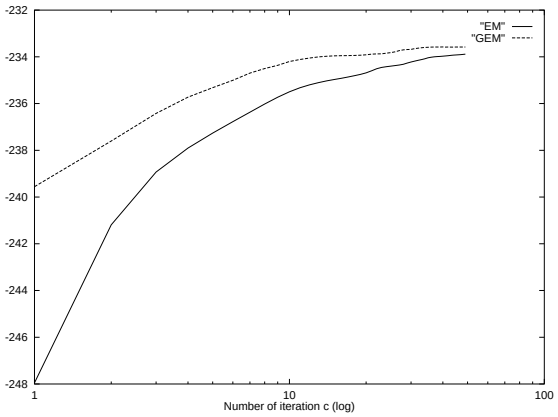

Figure 4: Example 2: Convergent sequences of $Q(\Lambda^{[c+1]}|\Lambda^{[c]})$ $(I = 50, J = 32)$

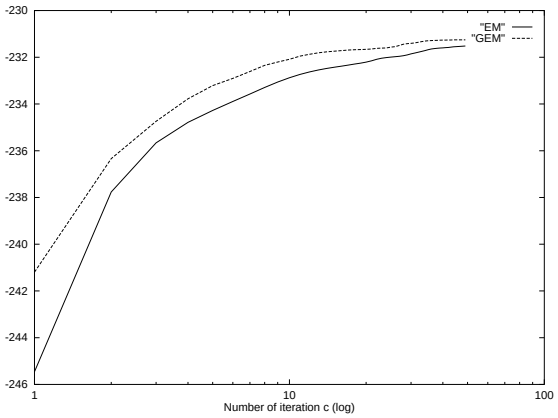

Figure 5: Example 2: Convergent sequences of $L(\mathcal{X}|\Lambda)$ $(I = 50, J = 32)$

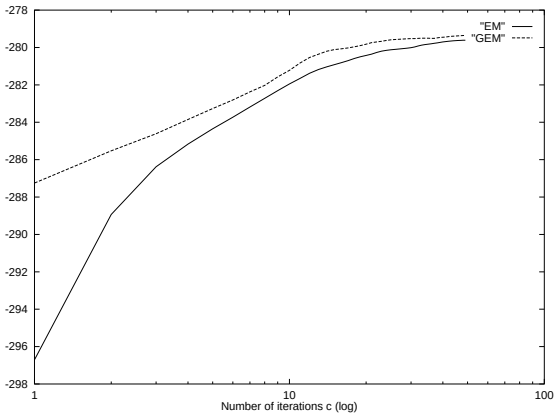

Figure 6: Example 3: Convergent sequences of $Q(\Lambda^{[c+1]}|\Lambda^{[c]})$ $(I = 60, J = 32)$

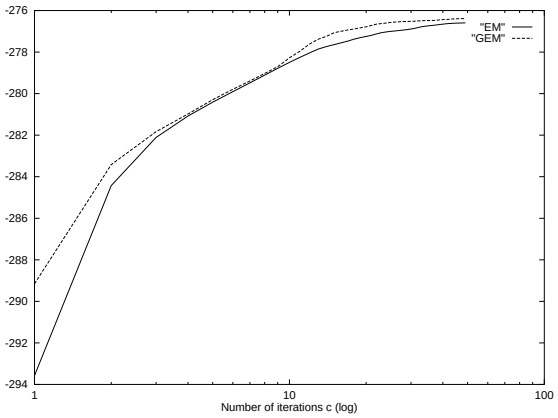

Figure 7: Example 3: Convergent sequences of $L(\mathcal{X}|\Lambda)$ $(I = 60, J = 32)$

References

[1] A. P. Dempster, N. M. Laird, D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistic Soc. Ser. B (methodological)*, 39:1–38, 1977.

[2] L. E. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[3] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: A Bayesian classification system. In *Proceedings of the 5th International Conference on Machine Learning*, Ann Arbor, 1988.

[4] P. F. Specht. Probabilistic neural networks. *Neural Networks*, 3(1):109–118, Jan. 1990.

[5] R. J. A. Little and D. B. Rubin. *Statistical analysis with missing data.* John Wiley, New York, 1987.

[6] M. I. Jordan and R. A. Jacobs. Hierachical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, October 1994.

[7] R. L. Streit and T. E. Luginbuhl. Maximum likelihood training of probabilistic neural networks. *IEEE Trans. on Neural Networks*, 5(5):764–783, September 1994.

[8] J. Wu and C. Chan. Isolated word recognition by neural network models with cross-correlation coefficient for speech recognition. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 15(11):1174–1185, November 1993.

[9] R. P. Lippmann. Review of neural networks for speech recognition. *Neural Computation*, 1(1):1–38, 1989.

[10] S. Lee. Supervised learning with Gaussian potentials. In *Neural Networks for Signal Processing.* Prentice-Hall, 1992.

[11] A. S. Lee and R. M. Kil. A Gaussian potential function network with hierarchically self-organizing learning. *Neural Networks*, 4(1):207–224, 1991.

[12] H. G. C. Traven. A neural network approach to statistical pattern classification by semi-parametric estimation of probability density function. *IEEE Trans on Neural Networks*, 2(3):366–377, May 1991.

[13] H. Ney. On the probabilistic interpretation of neural network classifiers and discriminative training criteria. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(2):107–119, February 1995.

[14] X.D. Huang and M.A.Jack. *Hidden Markov Models for Speech Recognition.* University of Eidinbourg Press, Edinbourg, 1989.

[15] H. Li, J.-P. Haton, and Y. Gong. The MMI learning of probabilistic neural networks. In *International Conference on Artificial Neural Networks*, Paris, France, 1995.

[16] S. E. Levison, L. R. Rabiner, and M. M. Sondhi. An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition. *The Bell System Technical Journal*, 62(4):1035–1074, 1983.

[17] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, 1973.

[18] L. A. Liporace. Maximum likelihood estimation for multivariate observations of Markov sources. *IEEE Trans. on Information Theory*, 28(5):729–734, September 1982.

[19] D. A. Reynolds. Speaker identifiaction and verification using Gaussian mixture speaker models. In *Proc. of ESCA Workshop on Automatic Speaker Recognition, Identification and Verification*, pages 27–30, Martigny, Switzerland, April 1994.

[20] H. Li, J.-P. Haton, J. Su, and Y. Gong. Speaker recognition with temperal transition models. In *Proceedings of European Conference on Speech Technology*, Madrid, Spain, 1995.

[21] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for the vector quantizer design. *IEEE Trans. on Communication*, 28(1):84–95, Jan. 1980.

[22] D. A. Reynolds and R. C. Rose. Robust text-independent speaker identification using Gaussian mixture speaker models. *IEEE Trans on Speech and Audio Processing*, 3(1):72–83, Jan. 1994.

[23] H. Li, C. Chan, J. Bao, and G. Cai. Hidden markov models in terms of a mixture of guassian phonetic states. *Computer processing of Chinese and oriental languages*, 5(2):193–200, March 1991.

[24] H. Li, J.-P. Haton, and Y. Gong. On MMI learning of Gaussian mixture for speaker models. In *Proceedings of European Conference on Speech Technology*, Madrid, Spain, 1995.

[25] Y. Gong and J.-P. Haton. Non-linear interpolation methods for speaker recognition and verification. In *Proc. of ESCA Workshop on Automatic Speaker Recognition, Identification and Verification*, pages 23–26, Martigny, Switzerland, April 1994.

[26] B. H. Juang and L. Rabiner. *Fundementals of speech recognition*. Prentice Hall, 1993.

[27] J.-L. Gauvain and C.-H. Lee. Maximum a Posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE Trans on Speech and Audio Processing*, 2(2):291–298, April 1994.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399