PROBABILISTIC MAPPING NETWORKS FOR SPEAKER RECOGNITION

Haizhou Li³  Yifan Gong³  Jean-Paul Haton³

³ CRIN-CNRS/INRIA-Lorraine, Campus Scientifique BP 239, 54506 Vandoeuvre-lès-Nancy Cedex, France
³ Institute of Radio Engineering and Automation, South China University of Technology, P.R. China

ABSTRACT
The Expectation-Maximization (EM) algorithm is a general technique for maximum likelihood estimation (MLE). In this paper, we present two important theoretical issues concerning Gaussian mixture modeling (GMM) within the EM framework. First, we propose an EM algorithm for estimating the parameters of a GMM structure dedicated to speaker recognition, the probabilistic mapping network (PMN), where the Gaussian probability density function is realized as an internal node. Hence, the EM algorithm is extended to deal with the supervised learning of a multcategory classification problem and serves as a parameter estimator of the neural network classifier. Then, a generalized EM (GEM) algorithm is developed as an alternative to the MLE problem of PMN. The effectiveness of the proposed PMN architecture and developed EM algorithms are assessed by conducting a set of speaker recognition experiments. It is shown that GEM converges faster than EM to the same solution space.

1 INTRODUCTION
A Gaussian mixture model (GMM) approach for speaker verification has been studied previously [1], where the GMM is used to represent a speaker as a class. One flaw of the class dependent GMM is that, when a large population is involved, the computation of pdf might become enormous. A pooled model, the probabilistic mapping networks (PMN) has been proposed in [2] to alleviate this problem by pooling GMMs across speakers into a same base of Gaussian kernel. Speakers are discriminated only by the mixture weighting coefficients, so that the mixture components, or Gaussian kernel, for all speakers could be computed in one pass. It is known that pooling parameters may reduce the discriminability of the system, which can be compensated by increasing the kernel size. Unfortunately, the procedure in [2] is considered to fix the shared kernels rather than to tie them. In other words, the shared kernels are not involved in model learning, which may lead to a sub-optimal solution. A maximum likelihood estimation (MLE) algorithm was proposed in [3] for the same model structure, which is based on the gradient descent procedure. It is generally agreed that the EM framework is more efficient and less complex than the general optimization procedure for maximum likelihood estimation. This paper investigates the PMNs trained with EM and their application to speaker recognition.

2 GMM AND PMN
Let \( \mathcal{X} = \{ x_i \}_{i=1}^{I}, x_i = \{ x_{ik} \}_{k=1}^{T_i}, \) 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 \( x_{ik} \) is the \( k \)-th realization of class \( i \). Assuming 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

\[
p(x_{ik} | \Omega_i) = \sum_{j=1}^{J_i} p(x_{ik} | \omega_{ij}) Pr(\omega_{ij})
\]

(1)

where \( p(x_{ik} | \omega_{ij}) \) is called the component density and \( Pr(\omega_{ij}) \) the mixture coefficient.

Let \( p(x_{ik} | \omega_{ij}) \) be \( N(x_{ik}; m_{ij}, \Sigma_{ij}) \), the Gaussian pdf; \( \beta_{ij} = Pr(\omega_{ij}) \). Thus, \( \beta_{ij} \)'s are subject to

\[
\sum_{j=1}^{J_i} \beta_{ij} = 1 \quad \beta_{ij} > 0
\]

(2)

Considering the multcategory classification problem of \( I \) classes, the Bayes decision rule is applied to find the class with the maximum \textit{a posteriori} probability by

\[
p(\Omega_i|x) = \frac{p(x|\Omega_i) Pr(\Omega_i)}{\sum_{s=1}^{I} p(x|\Omega_s) Pr(\Omega_s)} = \frac{p(x, \Omega_i)}{p(x)}
\]

(3)
where $Pr(\Omega_i)$, denoted as $\alpha_i$, is subject to the constraints,
\[ \sum_{i=1}^{I} \alpha_i = 1 \quad \alpha_i > 0 \] (5)

The parameters of the Gaussian mixture model (GMM) can be collectively represented by the notation:
\[ \Lambda_{ij} = \{m_{ij}, \Sigma_{ij}, \beta_{ij}\} \]
\[ \Lambda_i = \{\Lambda_{ij}\}_{j=1}^{J} \]
\[ \Lambda = \{\Lambda_i, \alpha_i\}_{i=1}^{I} \]

\( \Lambda \) 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[4], as it reduces the number of free parameters to be estimated. Let the number of mixtures be 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 \( \{m_{ij} = m_j, \forall i\}, \{\Sigma_{ij} = \Sigma_j, \forall i\} \). To provide an insight into the above definition, a four-layer feedforward neural network, the probabilistic mapping network (PMN), 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 the classes are the class-dependent mixture coefficients \( \{\beta_{ij}\} \). The parameter \( \Lambda_{ij} \) is thus redefined as \( \Lambda_{ij} = \{m_j, \Sigma_j, \beta_{ij}\} = \{\omega_j, \beta_{ij}\} \) and \( \Lambda \) consists of three parameter sets
\[ \alpha = \{\alpha_i\}_{i=1}^{I} \]
\[ \beta = \{\{\beta_{ij}\}_{j=1}^{J}\}_{i=1}^{I} \]
\[ \omega = \{\omega_j\}_{j=1}^{J} \]

It is obvious that, when the parameters are tied in this way, the training of the Gaussian kernel in \( \Lambda, \Lambda_i \in \Omega \) where \( \Omega \) is the parameter space, is concerned with the data from multiple classes.

3 EM ALGORITHM FOR PMN

The EM algorithm aims at maximizing the overall likelihood of training data presented to the estimated classifier. Within the framework of EM[5], the sample data \( x_{ik} \) is observable and is called incomplete data. To develop an EM algorithm for the PMN architecture, we have to 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 \( x_{ik} \) is defined, such that \( y_{ik} \in J, J \) 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 \( x_{ik} \) with \( y_{ik}, (x_{ik}, y_{ik}) \), 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 EM procedure

The likelihood function of the labeled training data set \( \mathcal{X} \) is given as
\[ L(\mathcal{X} | \Omega) = \log \left[ \prod_{i=1}^{I} \prod_{k=1}^{T_i} \alpha_i p(x_{ik} | \Omega_i) \right] \] (6)

Following the EM procedure, the auxiliary function \( Q(\Lambda | \Lambda') \) takes the expectation over the incomplete data [5].
\[ Q(\Lambda | \Lambda') = \sum_{i=1}^{I} T_i \log \alpha_i + \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{T_i} p(\omega_j | x_{ik}) \log \beta_{ij} \]
\[ + \sum_{j=1}^{J} \sum_{i=1}^{I} T_i \sum_{k=1}^{T_i} p(\omega_j | x_{ik}) \log p(x_{ik} | \omega_j) \]
\[ = Q_\alpha + \sum_{i=1}^{I} Q_\beta + \sum_{j=1}^{J} Q_\omega \] (8)

To solve the maximization problem, \( Q_\alpha, Q_\beta \) and \( Q_\omega \) can be maximized separately. With the help of the Lagrange multipliers, one can introduce the parameter constraints on \( \alpha \) and \( \beta \) in Eq.(5) and Eq.(2), into the optimization problem. \( \Lambda \) is solved by differentiating the Lagrangian with respect to each parameter in the set, setting the partial derivatives equal to zero:
\[ \alpha_i = \frac{T_i}{T} \] (9)
\[ \beta_{ij} = \frac{1}{T_i} \sum_{k=1}^{T_i} p(\omega_j | x_{ik}) \] (10)
\[ m_j = \frac{\sum_{i=1}^{I} \sum_{k=1}^{T_i} p(\omega_j | x_{ik}) x_{ik}}{\sum_{i=1}^{I} \sum_{k=1}^{T_i} p(\omega_j | x_{ik})} \] (11)
\[ \Sigma_j = \frac{\sum_{i=1}^{I} \sum_{k=1}^{T_i} p(\omega_j | x_{ik}) x_{ik}^2}{\sum_{i=1}^{I} \sum_{k=1}^{T_i} p(\omega_j | x_{ik})} - m_j m_j^T \] (12)
The EM procedure can be formulated as:

1. Initialization: Compute the class \( a \) priori probability \( \{a_i\} \) by Eq. (9). Set \( c = 1 \). Choose an initial estimation \( \Lambda^c \).

2. E-step: Compute \( Q(\Lambda | \Lambda^c) \) by Eq. (7) given \( \Lambda^c \).

3. M-step: Solve for \( \Lambda^{c+1} \) such that

\[
\Lambda^{c+1} = \arg \max_{\Lambda} Q(\Lambda | \Lambda^c) \tag{13}
\]

by Eq. (10), Eq. (11) and Eq. (12). \( \Lambda^{c+1} = M(\Lambda^c) \).

4. Evaluation: Set \( c = c + 1 \), repeat from step 2 until convergence.

### 3.2 Generalized EM procedure

Training a PMN, the convergence rate of EM depends on the information loss due to the data incompleteness, i.e., the number of missing data relative to the number of training data [5]. Given \( \Lambda^* \) as a fixed point of EM,

\[
DM(\Lambda^*) = D^{20} H(\Lambda^* | \Lambda^*)[D^{20} Q(\Lambda^* | \Lambda^*)]^{-1} \tag{14}
\]

With the tying scheme, all the PMN parameters and the training data are involved in each iteration. Hence, the rate of convergence becomes a critical issue. To cope with this problem, a generalized EM algorithm is proposed making an effort to increase Eq. (14) at each iteration [4].

In the EM algorithm, the M-step involves the maximization of a likelihood function that is reevaluated by the E-step. In general, 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 \( \Lambda \) is only partially updated at each step. To be more specific, only \( \omega_j \) is reestimated at each step. This implies that the \( Q \)-function will be evaluated \( J \) times before the Gaussian kernel is totally updated. If the EM algorithm is viewed as a batch learning of Gaussian pdfs, GEM allows us to learn the pdfs sequentially.

In GEM, two levels of iterations are involved. In each inner iteration, \( J \) maximization iterations 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 to update pdfs in the kernel one by one. All \( \{\omega_{ij}, j \neq j\} \) remain unchanged when updating \( \omega_j \). In comparison with EM, the extra computation involved in GEM is the \( J \) times more frequent updating of \( \beta_{ij} \). However, one should note that the calculation of Gaussian pdf is the most time-consuming one. Therefore, the computation of \( J \) maximization mappings in GEM is almost the same as that of one EM step. The convergence theorems for EM and GEM algorithms for PMN structure can be found in [4].

### 4 EXPERIMENTS

We examined the convergence and classification aspects for text-dependent speaker recognition. Since the capacity of the PMN is an important issue, three convergence examples are given. The convergence of a PMN structure was observed with respect to different speaker populations. In the recognition assessment, the model trained with respect to different speaker populations was also evaluated. Another critical factor 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 kernel in all the experiments were initialized by LBG algorithm. The same initializations are used in the EM and the GEM tests.

A database of French sentences from 60 speakers, 30 males and 30 females, was used, in which a three-word sentence of approximately one second was naturally uttered five times in separate sessions by each speaker. A 13th 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 for a speaker were used in learning just to examine the differences between the convergences of EM and GEM. The convergent curves of one experiment are plotted in Figures 2 and 3, where the comparisons of \( Q(\Lambda^{c+1} | \Lambda^c) \) and \( L(\lambda | \Omega) \) at \( [c] \)th EM step and \( [c] \)th GEM outer iteration are illustrated. In the speaker recognition experiment, using one of the five sessions as test data and the others as the training data, rotating the orders, resulted in five assessment sets. The five open tests and their average are reported in Tables 1 and 2, showing the result of recognition assessment with respect to different populations and kernel size.

### 5 CONCLUSION

Two EM algorithms have been derived theoretically for PMN learning, and also put into practice with successful results. The algorithms benefit from the well-defined EM framework. By observing EM and GEM, it is noted that \( L(\lambda | \Omega) \) provides a good indication of how well an estimate of \( \Lambda \) is reached. It is shown that
GEM converges faster and better than EM algorithm, due to the frequently updating of $\beta_j$.

In comparison with the class dependent GMM, the PMN reduces the computation of Gaussian pdf in a recognition trial while maintaining high recognition performance. It was shown that the PMN is suitable for speaker recognition with limited training data. The EM and the GEM estimates of $\Lambda$ performed the same in the recognition tests. This is confirmed by a theorem which shows that the GEM converges to the same solution space as EM[4].

REFERENCES


<table>
<thead>
<tr>
<th>$J$</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>90.0</td>
<td>95.0</td>
<td>95.0</td>
<td>95.0</td>
<td>90.0</td>
<td>93.0</td>
</tr>
<tr>
<td>50</td>
<td>82.0</td>
<td>94.0</td>
<td>96.0</td>
<td>96.0</td>
<td>92.0</td>
<td>92.0</td>
</tr>
<tr>
<td>60</td>
<td>93.3</td>
<td>95.0</td>
<td>93.3</td>
<td>90.0</td>
<td>86.7</td>
<td>91.7</td>
</tr>
</tbody>
</table>

Table 1: Recognition accuracy (%) with respect to number of speakers, $I$, for PMN with 32 pdfs, $J = 32$ (five open tests & average)

<table>
<thead>
<tr>
<th>$J$</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>82.0</td>
<td>94.0</td>
<td>96.0</td>
<td>96.0</td>
<td>92.0</td>
<td>92.0</td>
</tr>
<tr>
<td>64</td>
<td>96.0</td>
<td>98.0</td>
<td>98.0</td>
<td>94.0</td>
<td>90.0</td>
<td>95.2</td>
</tr>
<tr>
<td>128</td>
<td>96.0</td>
<td>100.0</td>
<td>98.0</td>
<td>100.0</td>
<td>100.0</td>
<td>98.8</td>
</tr>
</tbody>
</table>

Table 2: Recognition accuracy (%) with respect to kernel size, $J$, for 50 speakers, $I = 50$ (five open tests & average)

Figure 1: An architecture for PMN with Gaussian kernel

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

Figure 3: Convergent sequences of $L(\lambda|\Lambda)$ ($I = 50, J = 32$)