Boosted Mixture Learning of Gaussian Mixture Hidden Markov Models Based on Maximum Likelihood for Speech Recognition

Jun Du, Yu Hu, and Hui Jiang, *Member, IEEE*

***Abstract*—In this paper, we apply the well-known boosted mixture learning (BML) method to learn Gaussian mixture HMMs in speech recognition. BML is an incremental method to learn mixture models for classification problems. In each step of BML, one new mixture component is estimated according to the functional gradient of an objective function to ensure that it is added along the direction that maximizes the objective function. Several techniques have been proposed to extend BML from simple mixture models like the Gaussian mixture model (GMM) to the Gaussian mixture hidden Markov model (HMM), including Viterbi approximation for state segmentation, weight decay and sampling boosting to initialize sample weights to avoid overfitting, combination between partial updating and global updating to refine model parameters in each BML iteration, and use of the Bayesian Information Criterion (BIC) for parsimonious modeling. Experimental results on two large-vocabulary continuous speech recognition tasks, namely the WSJ-5k and Switchboard tasks, have shown that the proposed BML yields significant performance gain over the conventional training procedure, especially for small model sizes.**

***Index Terms*—Boosted mixture learning (BML), boosting, functional gradient, speech recognition.**

I. Introduction

In state-of-the-art automatic speech recognition (ASR) systems, we normally use Gaussian mixture HMMs as acoustic models to model basic speech units, ranging from context-independent whole words in small vocabulary ASR tasks to context-dependent phonemes (e.g., triphones) in large vocabulary ASR. Traditionally, the HMM-based acoustic models are estimated from available training data using the well-known EM algorithm based on the maximum-likelihood (ML) criterion. To deal with data sparseness problems in model training, we normally use phonetic decision trees to tie HMM states from different triphone contexts, which leads to state-tying triphone HMMs. In order to derive a simple closed-form solution, we normally grow the decision trees based on simple models, such as single Gaussian HMMs [1]. After the state-tied structure is determined from the decision trees, a separate "mixing-up" step is used to gradually increase the number of Gaussian mixtures in each tied HMM state until the optimal performance is achieved. In today's ASR systems, e.g., HTK, "mixing-up" is normally implemented in two steps [2]: 1) all existing Gaussians or the most dominant Gaussian mixture component in an HMM state is split based on some random or heuristic strategies; 2) all split Gaussians are re-estimated based on the EM algorithm. Obviously, this incremental method for increasing model complexity is a good strategy to learn very large-scale statistical models without getting trapped in any bad local optimum. However, we still face some problems when increasing model complexity in the above "mixing-up" strategy. First of all, the random splitting strategy is not optimal in terms of the model estimation criterion. For example, there is no guarantee that the newly added Gaussian components from random splitting always increase the likelihood function prior to re-estimation. Second, since the subsequent EM-based re-estimation is sensitive to the initial parameters of the randomly split Gaussians, there is no guarantee that the EM-based re-estimation can always converge to the optimal point if starting from randomly split Gaussians as initial values.

On the other hand, the concept of boosting has been widely applied to various pattern classification problems in machine learning. Boosting is a general method to combine multiple classifiers to improve the overall classification accuracy for almost any type of learning algorithm [3]. The basic idea is to sequentially train and combine a collection of weak classifiers to construct a strong and reliable classifier in such a way that these individual classifiers focus more and more on the hard-to-classify training examples [7]. As one example, in the well-known AdaBoost algorithm [4], a probability distribution is introduced and maintained for all training samples in the input space. Initially, all training samples are assigned with equal weights. In the following training iterations, the weights of those hard-to-classify examples are gradually increased, being updated automatically based on classification results of the current classifier on the whole training set. Using the updated training weights, a new classifier is trained to concentrate more on these hard examples. Next, the new classifier is combined with the current classifier to derive an enhanced classifier based on the boosting theory that analytically minimizes an upper bound on training error rate. This process is repeated to produce a set of individual ("base") classifiers. The boosting algorithm has been successfully applied to a variety of classification tasks, which can

Manuscript received September 20, 2010; revised November 21, 2010 and January 20, 2011; accepted January 21, 2011. Date of publication February 07, 2011; date of current version July 22, 2011. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Engin Erzin.

J. Du was with iFlytek Research, Hefei, Anhui, China. He is now with the Visual Computing Group, Microsoft Research Asia, Beijing 100190, China. (e-mail: unuedjwj@ustc.edu).

Y. Hu is with iFlytek Research, Hefei, Anhui 230088, China (e-mail: yuhu@iflytek.com).

H. Jiang is with the Department of Computer Science and Engineering, York University, Toronto, ON M3J 1P3, Canada (e-mail: hj@cse.yorku.ca).

Digital Object Identifier 10.1109/TASL.2011.2112352

improve accuracy and generalization over the individual base classifiers [6]. Furthermore, some recent theoretical work has shown that the boosting algorithm can effectively increase the *margin* of all training samples, which can be explained by a theoretical view related to functional gradient techniques [4]–[8]. The application of this boosting framework for probability density estimation has been widely studied [9]–[11]. More recently, the traditional boosting algorithms have been extended to some learning problems of mixture models [12], [13], which is called boosted mixture learning (BML). The basic idea of BML is to learn mixture models in an incremental and recursive manner. The BML always starts from a single mixture model and gradually adds a new mixture component in such a way that it always optimizes a predefined objective function. There are actually many other methods [14]–[16] which can follow a similar procedure to learn mixture models, but the essential point of BML is that a new mixture component is estimated in each step according to the functional gradient of the objective function so that each new component is always added to the direction that increases the objective function the most. Compared with the traditional random splitting, BML is less sensitive to the initial parameter values and it may converge to a better optimal point.

In this paper, we study how to apply the boosting algorithm to acoustic modeling in automatic speech recognition [28]. In previous work, some AdaBoost-like traditional boosting algorithms have been directly applied to phoneme classification and continuous speech recognition as in [19]–[21]. In this paper, we use BML to learn Gaussian mixture HMMs as an alternative method for random-split based mixing-up in speech recognition. As the first step, we only consider the maximum-likelihood (ML) estimation criterion in BML, where the objective function of BML is defined as the likelihood function of model parameters. In this paper, we first review the standard BML algorithm for Gaussian mixture models (GMMs) and then extend it to Gaussian mixture HMMs for speech recognition. Furthermore, several modifications have been proposed to make BML feasible and effective in the HMM framework, especially for state-of-the-art speech recognition systems. First, the Viterbi approximation is proposed to obtain state segmentation and BML of HMMs is conducted according to the Viterbi state segmentation. In this way, BML of Gaussian mixture HMMs can be formulated as the same BML problem of GMMs. Second, weight decay [17] using power scaling and sampling boosting [4] are proposed to deal with the over-fitting problem caused by unbounded sample weights. Third, we propose to update the entire Gaussian mixture model whenever a new component is added to the mixture while only the newly added mixture component is normally updated in each traditional BML step. This is called global updating, which is found to significantly improve recognition performance in speech recognition. Finally, the Bayesian Information Criterion (BIC) [23], [24] is used as the convergence criterion in BML to control the size of model parameters for parsimonious modeling. The proposed BML methods to learn Gaussian mixture HMMs have been evaluated on two standard large-vocabulary continuous speech recognition tasks using the Wall Street Journal (WSJ0) and Switchboard databases. Experimental results have shown that the proposed BML methods significantly outperform the standard training procedure in the WSJ-5k task. In the more challenging Switchboard task, compared with the standard training procedure, the BML methods clearly yield much better performance for small model sizes and comparable performance when the model size is further increased.

It is noted that a similar idea of functional gradient based learning of Gaussian mixture HMMs has recently been proposed in [29]. The method in [29] significantly differs from the algorithm developed in this paper. First, the algorithm in [29] is based on the traditional framework of the Baum-Welch algorithm, which is totally different from our method. The impact of functional gradient based boosted learning is reflected in the sample weight, which is heuristically inserted into the updating formulas of parameters for Gaussian mixture state emission densities in Baum-Welch algorithm. Second, in [29] the method is applied to speech emotion recognition while in this work we focus on some more challenging recognition tasks. As a result, several key techniques have been proposed to make BML effective for large vocabulary continuous speech recognition.

The remainder of the paper is organized as follows: In Section II, we give a review of general BML formulation for mixture models. In Section III, the BML formulation is expanded and discussed in detail for a special mixture model, namely the Gaussian mixture model (GMM). In Section IV, we extend the BML methods for GMMs to estimation of Gaussian mixture HMMs for speech recognition. In Section V, we report experimental results on two standard large vocabulary speech recognition tasks using the WSJ0 and Switchboard databases and finally we conclude the paper with our conclusions and findings in Section VI.

II. Review of Boosted Mixture Learning (BML)

In this section, we first review the well-known boosted mixture learning (BML) algorithm that has been successfully applied to many pattern classification problems.

First of all, as in [22], a general mixture model $F_K(\mathbf{x})$ can be represented as

$$F_K(\mathbf{x}) = \sum_{k=1}^{K} c_k f_k(\mathbf{x}), \quad c_k \geq 0, \sum_{k=1}^{K} c_k = 1 \tag{1}$$

where K is the number of mixtures, $\mathbf{x}$ is a feature vector, and c_k and $f_k(\mathbf{x})$ stand for the weight and component distribution of the kth mixture, respectively.

Learning of mixture models has been extensively studied in machine learning. As mentioned above, the traditional method is based on random splitting and EM-based re-estimation. In this paper, we focus on a different method to learn mixture models, which is named boosted mixture learning (BML) [12], [13]. At each stage of BML, a new component (c_k, f_k) is added to the previous mixture model F_{k-1} with $k-1$ mixture components to grow into a new mixture model F_k with k mixture components as follows:

$$F_k(\mathbf{x}) = (1 - c_k)F_{k-1}(\mathbf{x}) + c_k f_k(\mathbf{x}) \tag{2}$$

where c_k denotes a weight to combine the new mixture component with the current mixture model. This procedure is repeated until some convergence condition is met. The general procedure

TABLE I
DESCRIPTION OF BML PROCEDURE

Step 1:	Initialize F_1 .
Step 2:	For $k = 2, 3, ...$
	$\{c_k^*, f_k^*\} = \arg\max_{c_k, f_k} \mathcal{C}(F_k)$
	Continue to add the new component?
	Yes: $F_k(\mathbf{x}) = (1 - c_k^*)F_{k-1}(\mathbf{x}) + c_k^* f_k^*(\mathbf{x})$
	No: Goto Step 4
Step 3:	Goto Step 2
Step 4:	Output final mixture model F_k

of BML can be described as in Table I. The key idea of BML is that each new mixture component f_k and its mixture weight c_k can be learned based on a predefined objective function, denoted as $\mathcal{C}$, in an optimal way.

If we consider maximum-likelihood (ML) estimation, the objective function $\mathcal{C}$ can be defined as the log likelihood function of the mixture model F_k, based on all training data $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T\}$ as follows:

$$\mathcal{C}(F_k) = \sum_{t=1}^{T} \log F_k(\mathbf{x}_t) \tag{3}$$

where T is the number of training samples. In order to derive a new mixture component and its weight optimally as in Step 2 of Table I, a functional gradient method [12], [13] is used. Assume the objective function $\mathcal{C}(F)$ is viewed as a functional of mixture model F. When a new mixture component f_k is added, hopefully it will increase the objective function with respect to F as much as possible:

$$\mathcal{C}\left((1-\varepsilon)F_{k-1} + \varepsilon f_k\right) > \mathcal{C}(F_{k-1}) \tag{4}$$

where ε is a small deviation constant.

In order to expand the above functional, let us first define an inner product space where the above functional can be expanded according to a vector Taylor's series. Let us define a functional inner product space based on an inner product between any two mixture models P and Q using the available training samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T\}$ as follows:

$$\langle P, Q \rangle = \frac{1}{T} \sum_{t=1}^{T} P(\mathbf{x}_t) Q(\mathbf{x}_t). \tag{5}$$

Assume we use the functional Taylor series to expand the left-hand side of (4) in this inner-product space, we have

$$\begin{aligned}
&\mathcal{C}\left((1-\varepsilon)F_{k-1} + \varepsilon f_k\right) \\
&\quad = \mathcal{C}\left(F_{k-1} + \varepsilon(f_k - F_{k-1})\right) \\
&\quad = \mathcal{C}(F_{k-1}) + \varepsilon \left\langle \nabla\mathcal{C}(F_{k-1}), (f_k - F_{k-1}) \right\rangle \\
&\qquad + O\left(\|\varepsilon(f_k - F_{k-1})\|\right) \\
&\quad \approx \mathcal{C}(F_{k-1}) + \varepsilon \left\langle \nabla\mathcal{C}(F_{k-1}), (f_k - F_{k-1}) \right\rangle .
\end{aligned} \tag{6}$$

If deviation ε is small enough, all high-order items $O(\|\varepsilon(f_k - F_{k-1})\|)$ can be ignored. Thus, the objective function at the new mixture model can be approximated by only the linear terms as in (6).

Ideally speaking, the new mixture component f_k should be estimated in such a way that increases the objective function $\mathcal{C}((1-\varepsilon)F_{k-1} + \varepsilon f_k)$ most rapidly. If we adopt the above linear approximation in the inner-product space, it means that f_k should be chosen to maximize the inner product $\langle \nabla\mathcal{C}(F_{k-1}), (f_k - F_{k-1}) \rangle$. Therefore, the basic idea of BML is to learn each individual mixture component, f_k, incrementally as follows:

$$f_k^* = \arg\max_{f_k} \left\langle \nabla\mathcal{C}(F_{k-1}), (f_k - F_{k-1}) \right\rangle . \tag{7}$$

This equation clearly shows that the new mixture component f_k is estimated along the direction of functional gradient where the objective function grows the most. The reason why the inner product between the functional gradient and the mixture model is used is to ensure that the new component f_k is estimated in such a way that the new model F_k still falls into the same model space as F_{k-1}.

Given the estimated f_k^*, in BML, the mixture weight c_k^* can be obtained by using the following line search:

$$c_k^* = \arg\max_{c_k \in [0,1]} \mathcal{C}\left((1-c_k)F_{k-1} + c_k f_k^*\right). \tag{8}$$

In practice, the optimal mixture weight c_k^* can be found efficiently by using a grid search in the interval [0,1].

If we consider the objective function as in (3), it is easy to show that the functional gradient can be calculated as

$$\nabla\mathcal{C}(F_{k-1})(\mathbf{x}_t) = \nabla\mathcal{C}(F)|_{F=F_{k-1}} = \frac{1}{F_{k-1}(\mathbf{x}_t)} \tag{9}$$

where the above functional gradient is evaluated at $F = F_{k-1}$, which is actually a function calculated at $\mathbf{x}_t$.

As a result, the BML learning formula in (7) can be rewritten for maximum-likelihood estimation (MLE) as follows:

$$\begin{aligned}
f_k^* &= \arg\max_{f_k} \frac{1}{T} \sum_{t=1}^{T} \frac{f_k(\mathbf{x}_t) - F_{k-1}(\mathbf{x}_t)}{F_{k-1}(\mathbf{x}_t)} \\
&= \arg\max_{f_k} \sum_{t=1}^{T} \frac{f_k(\mathbf{x}_t)}{F_{k-1}(\mathbf{x}_t)}.
\end{aligned} \tag{10}$$

Obviously, (10) is a general form to derive each new mixture component in BML based on the maximum-likelihood (ML) estimation criterion. In next section, we will consider how to solve (10) for the GMM.

III. BML OF GAUSSIAN MIXTURE MODELS

In this section, we consider to solve the optimization problem in (10) for Gaussian mixture models (GMMs), where each mixture component f_k is a D-dimensional multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$ as

$$\begin{aligned}
f_k(\mathbf{x}_t|\Phi_k) &= \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\
&= \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_k|^{1/2}} \\
&\quad \times \exp\left\{-\frac{1}{2}(\mathbf{x}_t - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_t - \boldsymbol{\mu}_k)\right\}
\end{aligned} \tag{11}$$

where Φ_k denotes parameters for kth Gaussian, i.e., $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, and Ψ_k in $F_k(\mathbf{x}_t|\Psi_k)$ stands for parameters of all Gaussian components in F_k. Obviously, we have $\Psi_k = \{\Phi_k, \Psi_{k-1}\}$.

There is no closed-form solution to solve the optimization problem in (10) for GMMs. In this paper, we optimize (10) iteratively using the EM algorithm to search for the optimal component f_k^*. It is noted that the same solution has been proposed without any detailed derivation for video processing in [12] and [13]. In this paper, we demonstrate that the solution can be derived by optimizing a lower bound on the right-hand side of (10) with the EM algorithm and the method will be applied to HMMs in speech recognition.

For convenience, we take the log of (10). Then the objective function becomes a log sum as follows:

$$\mathcal{L}(\Phi_k) = \log \sum_{t=1}^{T} \frac{f_k(\mathbf{x}_t|\Phi_k)}{F_{k-1}(\mathbf{x}_t|\Psi_{k-1})}. \tag{12}$$

Although there is no simple way to optimize the above log-sum function, it is possible to construct a tangential lower-bounded auxiliary function based on Jensen's inequality. As in the EM algorithm, the above $\mathcal{L}(\Phi_k)$ can be iteratively optimized by maximizing the auxiliary function in each iteration.

Let us define the posterior probability of the tth sample as follows:

$$\gamma_t\left(\Phi_k^{(n)}\right) = \frac{f_k\left(\mathbf{x}_t|\Phi_k^{(n)}\right)/F_{k-1}(\mathbf{x}_t|\Psi_{k-1})}{\sum_{t=1}^{T} f_k\left(\mathbf{x}_t|\Phi_k^{(n)}\right)/F_{k-1}(\mathbf{x}_t|\Psi_{k-1})}. \tag{13}$$

where $\Phi_k^{(n)}$ denotes initial model parameters in nth iteration.

According to Jensen's inequality, the auxiliary function, which is a lower bound of $\mathcal{L}(\Phi_k)$, can be constructed as

$$\begin{aligned}
\mathcal{L}(\Phi_k) &= \log \sum_{t=1}^{T} \gamma_t\left(\Phi_k^{(n)}\right) \frac{f_k(\mathbf{x}_t|\Phi_k)}{\gamma_t\left(\Phi_k^{(n)}\right) F_{k-1}(\mathbf{x}_t|\Psi_{k-1})} \\
&\geq \sum_{t=1}^{T} \gamma_t\left(\Phi_k^{(n)}\right) \log \left[\frac{f_k(\mathbf{x}_t|\Phi_k)}{\gamma_t\left(\Phi_k^{(n)}\right) F_{k-1}(\mathbf{x}_t|\Psi_{k-1})}\right] \\
&\quad \text{(Jensen's inequality)} \\
&= \sum_{t=1}^{T} \gamma_t\left(\Phi_k^{(n)}\right) \log f_k(\mathbf{x}_t|\Phi_k) + C\left(\Phi_k^{(n)}\right) \\
&= \mathcal{Q}\left(\Phi_k|\Phi_k^{(n)}\right)
\end{aligned} \tag{14}$$

where $C(\Phi_k^{(n)})$ is a constant independent of model parameters Φ_k

$$\begin{aligned}
C\left(\Phi_k^{(n)}\right) = &-\sum_{t=1}^{T} \gamma_t\left(\Phi_k^{(n)}\right) \\
&\times \left[\log \gamma_t\left(\Phi_k^{(n)}\right) + \log F_{k-1}(\mathbf{x}_t|\Psi_{k-1})\right].
\end{aligned} \tag{15}$$

As the M-Step of EM, we maximize the auxiliary function $\mathcal{Q}(\Phi_k|\Phi_k^{(n)})$ with respect to Φ_k, which is equivalent to maximizing the log likelihood of $f_k(\mathbf{x}_t|\Phi_k)$ given posterior probabilities $\gamma_t(\Phi_k^{(n)})$. A closed-form solution can be easily derived as follows by setting the derivative to zero:

$$\frac{\partial \mathcal{Q}\left(\Phi_k|\Phi_k^{(n)}\right)}{\partial \Phi_k} = 0. \tag{16}$$

Substituting (14) into (16), we can derive the iterative re-estimation formula in the $(n+1)$th for model parameters of f_k as follows:

$$\begin{aligned}
\boldsymbol{\mu}_k^{(n+1)} &= \frac{\sum_{t=1}^{T} w^{(n)}(\mathbf{x}_t)\cdot \mathbf{x}_t}{\sum_{t=1}^{T} w^{(n)}(\mathbf{x}_t)} \\
&= \sum_{t=1}^{T} \gamma_t\left(\Phi_k^{(n)}\right)\cdot \mathbf{x}_t
\end{aligned} \tag{17}$$

$$\begin{aligned}
\boldsymbol{\Sigma}_k^{(n+1)} &= \frac{\sum_{t=1}^{T} w^{(n)}(\mathbf{x}_t)\cdot \left(\mathbf{x}_t - \boldsymbol{\mu}_k^{(n+1)}\right)\left(\mathbf{x}_t - \boldsymbol{\mu}_k^{(n+1)}\right)^{\top}}{\sum_{t=1}^{T} w^{(n)}(\mathbf{x}_t)} \\
&= \sum_{t=1}^{T} \gamma_t\left(\Phi_k^{(n)}\right)\cdot \left(\mathbf{x}_t - \boldsymbol{\mu}_k^{(n+1)}\right) \\
&\quad \times \left(\mathbf{x}_t - \boldsymbol{\mu}_k^{(n+1)}\right)^{\top}
\end{aligned} \tag{18}$$

$$w^{(n)}(\mathbf{x}_t) = \frac{f_k\left(\mathbf{x}_t|\Phi_k^{(n)}\right)}{F_{k-1}(\mathbf{x}_t|\Psi_{k-1})} \tag{19}$$

where $w^{(n)}(\mathbf{x}_t)$ denotes a weight assigned to sample $\mathbf{x}_t$ after nth iteration, similar to sample weights used in the traditional boosting algorithms. The physical meaning of sample weight is that samples with lower probability by F_{k-1} are given larger weights than those more likely samples. Hence, the new component f_k focuses on those samples poorly modeled by the current mixture model F_{k-1}.

In many BML algorithms, the sample weight is often initialized for the first iteration as

$$w^{(0)}(\mathbf{x}_t) = \frac{1}{F_{k-1}(\mathbf{x}_t|\Psi_{k-1})}. \tag{20}$$

In the subsequent iterations, we use (17), (18), and (19) to update mean vector, covariance matrix, and sample weights iteratively until it converges. After that, we can apply the line search in (8) to determine the optimal mixture weight c_k^*.

IV. BML of Gaussian Mixture HMMs for Speech Recognition

In this section, we extend the above BML algorithm for GMMs to estimation of Gaussian mixture HMMs for speech recognition. We have proposed several approximations and techniques to make the above standard BML procedure feasible and effective for speech recognition under the HMM framework.

A. Viterbi Approximation for State Segmentation

Due to the dynamic and sequential nature of HMMs, if we directly apply the above BML formulation in (2) to HMMs, it

leads to the so-called *mixture of HMMs*, where each component f_k is a single Gaussian HMM and $\mathbf{X}$ stands for a speech segment modeled by one HMM. Here a single Gaussian HMM means a state-tied triphone HMM having a single Gaussian for each state. As a result, the mixture model F_k becomes a mixture of many single Gaussian HMMs. In each iteration, the BML algorithm attempts to learn a new single Gaussian HMM, f_k, and combine it with the current model, F_{k-1}, to form a new mixture of HMMs, F_k. Therefore, the mixture of HMMs is a possible framework to apply BML to HMM in speech recognition. However, under the framework of a mixture of HMMs, we need to redesign the entire training and recognition procedure in speech recognition, and especially significant changes must be made in the Viterbi decoder to accommodate the new model form. In this paper, for simplicity, we consider an alternative way to use the BML algorithm under the current HMM framework instead of completely switching to a new modeling framework with a mixture of HMMs. In other words, we still use the traditional training procedure to estimate single Gaussian HMMs and then use the BML method to add more and more Gaussian components in each HMM state to grow model complexity. As discussed above, in BML, each new Gaussian component is estimated based on the functional gradient of the objective function so that it may have some benefits over the traditional "mixing-up" relying on random splitting.

Under the HMM framework, the log likelihood function can be viewed as a mixture of all possible hidden state sequences $s_0 s_1 \dots s_T$

$$\mathcal{C}(F) = \log p(\mathbf{X}) = \log \sum_{s_0 s_1 \dots s_T} \pi_{s_0} \prod_{t=1}^{T} a_{s_{t-1} s_t} F(\mathbf{x}_t | s_t) \tag{21}$$

where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T\}$ is an utterance from the training set, $\{\pi_i\}$ and $\{a_{ij}\}$ denote the initial state probabilities and state transition probabilities in HMMs, and $F(\mathbf{x}_t|s)$ represents the output probability distribution of one HMM state s, which is modeled by a GMM in Gaussian mixture HMMs. Obviously, if we use the BML algorithm to add a new Gaussian component to an HMM state, denoted as s, we need to calculate the functional gradient of the objective function, $\mathcal{C}(F)$, with respect to this state model given one frame $\mathbf{x}_t$, i.e., $F(\mathbf{x}_t|s)$, which can be easily derived as follows:

$$\begin{aligned}
&\nabla \mathcal{C}(F)|_{F=F(\mathbf{x}_t|s)} \\
&\quad = \left. \frac{\partial \mathcal{C}(F)}{\partial F} \right|_{F=F(\mathbf{x}_t|s)} \\
&\quad = \frac{1}{p(\mathbf{X})} \sum_{s_0 s_1 \dots s_T} \frac{\partial}{\partial F(\mathbf{x}_t|s)} \left[\pi_{s_0} \prod_{\tau=1}^{T} a_{s_{\tau-1} s_\tau} F(\mathbf{x}_\tau | s_\tau) \right] \\
&\quad = \frac{1}{p(\mathbf{X})} \sum_{s_0 s_1 \dots s_T} \Big[\delta(s_t, s) \cdot a_{s_{t-1} s_t} \pi_{s_0} \\
&\qquad\qquad \times \prod_{\tau \neq t, \tau=1}^{T} a_{s_{\tau-1} s_\tau} F(\mathbf{x}_\tau | s_\tau) \Big] \\
&\quad = \frac{1}{p(\mathbf{X})} \cdot \frac{p(\mathbf{X}, s = s_t)}{F(\mathbf{x}_t|s)} = \frac{p(s = s_t | \mathbf{X})}{F(\mathbf{x}_t|s)}
\end{aligned} \tag{22}$$

where $p(s = s_t|\mathbf{X})$ denotes the posterior probability, which can be calculated based on state occupancy statistics of s collected for $\mathbf{X}$ in the forward–backward algorithm. Obviously, (22) shows that the functional gradient of $\mathcal{C}(F)$ w.r.t. $F(\mathbf{x}_t|s)$ depends on the entire utterance $\mathbf{X}$ and the whole HMM model, not only the current frame $\mathbf{x}_t$ and the underlying state s. Therefore, if we follow the BML algorithm to add Gaussian components to an existing HMM, we are unable to decouple the BML algorithm to process all HMM states independently. In other words, if we substitute (22) to (7) to determine each new Gaussian component, we have to calculate the posterior probability for all HMM states and choose the maximum one as the candidate to add a new Gaussian component. After the new Gaussian is added to the HMM, posterior probabilities for all HMM states must be recalculated. Therefore, the computational complexity is quite high. In this paper, for simplicity, we accept the Viterbi approximation where the likelihood function is calculated based on the optimal Viterbi path instead of a summation over all possible state sequences. In this way, we can decouple BML of HMMs to deal with all HMM states independently so that the above BML algorithm of GMMs can be directly used to estimate state distribution of all HMM states independently.

Under the Viterbi approximation, we can calculate the log likelihood function as follows:

$$\begin{aligned}
\mathcal{C}(F) &\approx \log \left[\max_{s_0 s_1 \dots s_T} \pi_{s_0} \prod_{t=1}^{T} a_{s_{t-1} s_t} F(\mathbf{x}_t | s_t) \right] \\
&= \log \left[\pi_{s_0^*} \prod_{t=1}^{T} a_{s_{t-1}^* s_t^*} F\left(\mathbf{x}_t | s_t^*\right) \right] \\
&= \sum_{t=1}^{T} \log F\left(\mathbf{x}_t | s_t^*\right) + C
\end{aligned} \tag{23}$$

where $s_0^* s_1^* \dots s_T^*$ denotes the optimal state sequence obtained by the Viterbi algorithm. Under the Viterbi approximation, the functional gradient of the log likelihood function with respect to one HMM state s can be calculated as

$$\nabla \mathcal{C}(F)|_{F=F(\mathbf{x}_t|s)} = \frac{1}{F(\mathbf{x}_t|s)}. \tag{24}$$

Obviously, the above functional gradient has the same form as that of the GMM in (9). Therefore, the BML method for the GMM can similarly be used to estimate each individual HMM state under the Viterbi approximation. In this case, we first use an initial HMM to align each training utterance $\mathbf{X}$ to obtain its optimal Viterbi state sequence. Then, the BML method in Section III is used to incrementally estimate a Gaussian mixture for every HMM state based on all feature vectors aligned to this state. The estimated models can be used to realign all feature vectors so that the whole learning procedure can be run in an iterative manner. In this way, the BML method has similar computational complexity to the conventional training procedure since the BML training can be performed in parallel for all HMM states as long as training utterances are aligned based on the Viterbi state segmentation. It is noted that $\{\pi_i\}$ and $\{a_{ij}\}$ are not updated in the BML procedure since they are not critical to final performance in speech recognition.

B. Initialization of Sample Weights

After state segmentation, the GMM parameters of each HMM state can be learned using the BML algorithm in Section III, but we have realized several problems when we apply the BML to Gaussian mixture HMMs. The first problem is related to initialization of sample weights for each new mixture component using (20). In Gaussian mixture HMMs for speech recognition, it is found that the dynamic range of F_{k-1} is so large that the initial sample weights $w^{(0)}(\mathbf{x}_t)$ are dominated by only a very small number of samples with low probability. If we use (20) to initialize sample weights, the subsequent BML algorithm may cause overfitting since it quickly concentrates only on a few samples with low probability, which may be outliers. To deal with this problem, we have investigated two different methods to initialize sample weights.

In the first method, we use the so-called weight decay in [17] to smooth initial sample weights based on power scaling:

$$w^{(0)}(\mathbf{x}_t) = \left(\frac{1}{F_{k-1}(\mathbf{x}_t|\Psi_{k-1})} \right)^{\alpha} \tag{25}$$

where α is the exponential scaling factor $0 < \alpha < 1$. In the first iteration, (25) is first used to initialize sample weights for all training data and then (17) and (18) are used to initialize the corresponding mean vector and covariance matrix. As shown in [18], weight decay can significantly smooth sample weights and in turn make BML more robust especially in presence of outliers.

In the second method, the idea of sampling boosting in [4] is used to initialize sample weights. In sampling boosting, the initial sample weights calculated in (20) are used to sample all training data to form a subset according to some rules. In next iteration, only the selected subset of training data (assuming equal sample weights) is used to estimate the new Gaussian component. One possible way to sample training data is based on mean and variance of all initial sample weights. Assume mean μ and variance σ^2 of initial sample weights are denoted as $\mu = \text{mean}\{\log w^{(0)}(\mathbf{x}_t)\}$ and $\sigma^2 = \text{variance}\{\log w^{(0)}(\mathbf{x}_t)\}$, where $w^{(0)}(\mathbf{x}_t)$ stands for sample weights calculated in (20). Then, a subset of training samples with large sample weights is selected based on

$$\mathbf{X}_{\text{sub}} = \left\{ \mathbf{x}_t | \log w^{(0)}(\mathbf{x}_t) > \mu + \beta\sigma, \mathbf{x}_t \in \mathbf{X} \right\} \tag{26}$$

where $\mathbf{X}$ denotes the set of all training samples and β is a linear scaling factor to control the size of $\mathbf{X}_{\text{sub}}$. The goal of this selection mechanism is to ensure that the selected subset is large enough so that f_k is not dominated by a few samples. In next step, we only use the subset $\mathbf{X}_{\text{sub}}$ with equal sample weights to estimate mean vector and covariance matrix of the corresponding Gaussian.

It has been observed in our experiments that the above two initialization strategies for sample weights are very important to achieve good performance in speech recognition since they can effectively avoid the overfitting problem in BML. Typically, the proper values of exponential factor α and linear scaling factor β are not sensitive to any particular ASR task.

C. Partial and Global Updating in BML

In the traditional BML, when a new mixture component f_k is added to the mixture model, we first estimate a new mixture component as in (10) and then the mixture weight is estimated from a separate line search process as in (8). In this section, we propose an alternative method to estimate each mixture component and its weight. After we derive each new mixture component f_k using functional gradient, as in [22], we can use it as an initial point, and then apply the EM algorithm to optimize the original log likelihood function only with respect to the new mixture component f_k and weight c_k while F_{k-1} are assumed to be constants:

$$\Phi_k^* = \{f_k^*, c_k^*\} = \arg\max_{f_k, c_k} \mathcal{C}\left((1-c_k)F_{k-1} + c_k f_k\right). \tag{27}$$

The lower-bound based optimization as in Section III can similarly be used to solve the above optimization. For GMMs, it can be easily derived that mixture weight c_k, mean vector and covariance matrix of f_k are estimated in the $(n+1)$th $(n \geq 0)$ iteration as follows:

$$w^{(n)}(\mathbf{x}_t) = \frac{f_k\left(\mathbf{x}_t|\Phi_k^{(n)}\right)}{c_k^{(n)} f_k\left(\mathbf{x}_t|\Phi_k^{(n)}\right) + \left(1-c_k^{(n)}\right) F_{k-1}(\mathbf{x}_t|\Psi_{k-1})} \tag{28}$$

$$c_k^{(n+1)} = \frac{1}{T}\sum_{t=1}^{T} c_k^{(n)} w^{(n)}(\mathbf{x}_t) \tag{29}$$

$$\boldsymbol{\mu}_k^{(n+1)} = \frac{\sum_{t=1}^{T} w^{(n)}(\mathbf{x}_t)\cdot \mathbf{x}_t}{\sum_{t=1}^{T} w^{(n)}(\mathbf{x}_t)} \tag{30}$$

$$\boldsymbol{\Sigma}_k^{(n+1)} = \frac{\sum_{t=1}^{T} w^{(n)}(\mathbf{x}_t)\cdot \left(\mathbf{x}_t - \boldsymbol{\mu}_k^{(n+1)}\right)\left(\mathbf{x}_t - \boldsymbol{\mu}_k^{(n+1)}\right)^{\top}}{\sum_{t=1}^{T} w^{(n)}(\mathbf{x}_t)}. \tag{31}$$

In this paper, this step of re-estimation is named *partial EM*. Compared with the re-estimation based on the functional gradient method from (17) to (19), the updating formula for mean vector and covariance matrix are the same and the main difference is the estimation formula of sample weights. Comparing the estimation formula (19) with (28), it is easy to see that sample weights in partial EM have much smaller dynamic range due to normalization in (28). As a result, it may result in more robust and reliable estimation of the new mixture component. In partial EM, we simply initialize each mixture weight $c_k^{(0)}$ as $1/k$ for each BML iteration.

Our experimental results show that in each stage of BML, if only the newly added mixture component is updated, the convergence of recognition performance is quite slow. Therefore, similar to the re-estimation in partial EM, an additional EM-based re-estimation step can be applied to re-estimate all mixture components in F_k, not just f_k:

$$\Psi_k^* = \arg\max_{F_{k-1}, f_k, c_k} \mathcal{C}\left((1-c_k)F_{k-1} + c_k f_k\right). \tag{32}$$

Along the same line, the EM algorithm can be used to iteratively update all model parameters. In this paper, this step of re-estimation is called *global EM*. It has been found that

the additional global EM step can significantly improve performance of Gaussian mixture HMMs in speech recognition. It should be clarified that both partial EM and global EM are implemented under the state-level Viterbi approximation. For global EM, the updating formulas are very similar to those of the standard Baum–Welch EM. Actually, given an initial f_k, the system can be updated as in normal Baum–Welch EM.

D. BIC for Parsimonious Modeling

BML is an incremental and recursive learning process where only one new mixture component is added in each iteration. In this section, we use Bayesian information criterion (BIC) to select the optimal number of mixture components. The BIC has been widely used as a model selection criterion and it can be viewed as a regularized likelihood function as follows:

$$\mathrm{BIC}(k) = \mathcal{C}(F_k) - \frac{\lambda}{2} M_k \log(T) \tag{33}$$

where $\mathcal{C}(F_k)$ is the conventional log likelihood function defined in (3), M_k is the number of parameters used in mixture model F_k, and T represents total number of training samples. λ is the control parameter of updated model size. In our BML procedure, we first run BML to gradually increase the number of mixture components until a certain point. At last, we use the BIC criterion to roll back the model size and select the optimal value of k which maximizes the BIC criterion in (33) for each state. By doing so, we can typically reduce model size significantly for parsimonious modeling.

At last, we summarize the whole procedure of BML for HMMs in Table II. As shown in Table II, we first build single Gaussian state-tied triphone HMMs using the conventional training procedure and then apply the proposed BML methods to incrementally increase the number of Gaussians in each tied state until the optimal performance is achieved. During each step of BML, we only add one Gaussian component to each HMM state. The new Gaussian and its mixture weight are first calculated by the BML method based on the functional gradient of the log likelihood function, where each sample weight is initialized by either weight decay (WD) or sampling boosting (SB). Next, the new Gaussian and all other Gaussian components in the HMM state are refined by using the proposed partial EM and/or global EM, respectively. At last, BIC is used as the convergence criterion to determine the optimal number of Gaussian components for each HMM state.

V. Experiments

The proposed BML algorithms have been evaluated in two large-vocabulary continuous speech recognition tasks using the WSJ0 and Switchboard databases. The proposed BML methods have been compared with the conventional maximum-likelihood estimation (MLE) method as in [1], [2]. In the BML experiments, we set the exponential factor α in weight decay to 0.05. The linear scaling factor β in sampling boosting is set to -0.5. The initial single Gaussian HMMs are estimated using the standard HTK method as in [2]. For EM iterations, we set $N_P = N_G = 2$ for step 2.3 and 2.4 in Table II. The initial state probabilities and state transition probabilities in HMMs are not updated in the BML stage.

TABLE II
BML Procedure of HMMs for Speech Recognition

1.0 Initialization of BML:
Generate single Gaussian F_1 for all tied states in triphone HMMs
2.0 For $k = 2, 3, \ldots$ (each boosting stage for all F_k in all tied states)
2.1 State segmentation:
Use previous model F_{k-1} to do forced alignment to update state segmentation for all training samples
2.2 BML of new mixture component and its weight:
Use functional gradient to estimate f_k and c_k (initialized by WD or SB)
2.3 Refinement of new mixture component and its weight:
Use partial EM to refine f_k and c_k for N_P iterations
2.4 Refinement of all mixture components and all weights:
Use global EM to refine all parameters of F_k for N_G iterations
2.5 BIC as a convergence criterion of boosting:
If $\mathrm{BIC}(k) > \mathrm{BIC}(k-1)$ then continue; Otherwise, terminate and output F_{k-1}

A. Experimental Results in the WSJ-5k Task

In the WSJ0 task, the training set is the standard SI-84 set, consisting of 7133 utterances from 84 speakers (about 12 hours speech data in total). Evaluation is performed on the standard Nov'92 non-verbalized 5k closed-vocabulary test set (WSJ-5k), including 330 utterances from eight speakers. For the baseline system, we use HTK to build standard state-tied cross-word triphone HMMs [1], which includes a total number of 2774 tied states. The feature vectors are 39-dimensional MFCC features (including delta and delta-delta features) with sentence level cepstral mean normalization processing. A standard trigram language model is used in evaluation.

1) Compare BML With HTK in WSJ-5k: First of all, we build the baseline system using HTK [1], [2] to compare with our proposed BML algorithms. Here we adopt the standard training strategy to build our HTK baseline. As we know, we first build state-tied single Gaussian HMMs based on the phonetic decision tree and then gradually increase the number of Gaussian mixtures in each tied state. The strategy of HTK to split Gaussians is

$$k^* = \arg \max_{1 \le k \le K} (c_k - n_k) \tag{34}$$

where the mixture component with the largest mixture weight c_k is selected to split, which is equivalent to selecting the mixture component with the most of data assigned to it. Here n_k is the number of times that the kth mixture has been split, which is a penalty item to the mixture weight. Then, the selected mixture component k^* is replaced by the following two new components:

$$c_{1,2} = \frac{c_{k^*}}{2}, \quad \boldsymbol{\mu}_{1,2} = \boldsymbol{\mu}_{k^*} \pm 0.2\sqrt{\mathrm{diag}\{\boldsymbol{\Sigma}_{k^*}\}}, \quad \boldsymbol{\Sigma}_{1,2} = \boldsymbol{\Sigma}_{k^*} \tag{35}$$

where the actual split is performed by copying the selected mixture component, dividing the weights of both copies by 2, and finally perturbing the mean vectors by plus or minus

TABLE III
VARIOUS BML CONFIGURATIONS

	Weight Decay	Sampling Boosting	Partial EM	Global EM
BML1			√	
BML2			√	√
BML3	√		√	
BML4	√			√
BML5	√		√	√
BML6		√	√	√

0.2 standard deviations. If multiple components should be split at a time, the above procedure is repeated until the target mixture number is achieved. In our HTK baseline training procedure, we increase the number of mixtures by one in each state, where two EM iterations are run. The results are shown in Table IV, denoted as *HTK*. Our best HTK-trained baseline system (with $K = 9$) yields 5.06% in word error rate (WER), which is comparable with the best performance reported on this task.[1]

Next, we investigate several different ways to configure the BML method as summarized in Table III, which are all slightly different from the general BML algorithm as shown in Table II. For these BML methods, we use single Gaussian HMMs (with 2774 tied states) from the baseline system as the initial models for BML. *BML1* can be considered as the original version of BML widely used in other pattern classification areas [12], [13], [29]. *BML2* is modified by adding global EM to *BML1*. In *BML3*, we use the functional gradient to calculate each new Gaussian component and weight decay (WD) to initialize sample weights. In each iteration only partial EM is used to refine the new Gaussian component and its weight after functional gradient. On the other hand, *BML4* is configured in a similar way as *BML3* except that global EM is used to refine the whole mixture rather than partial EM. In *BML5*, both partial EM and global EM (one after another) are used to refine mixture models after functional gradient. Furthermore, *BML6* is similar to *BML5* except that sampling boosting (SB) is used to initialize sample weights in the functional gradient in place of weight decay. For all these configurations, recognition results are summarized in Table IV. First, it is clear that *BML1* is ineffective in speech recognition due to the unbounded sample weight. Although the performance of *BML2* is better than that of *BML1*, it is still much worse than that of our baseline system. However, we can see that *BML4* significantly outperforms *BML3* in recognition performance and it indicates that global EM is an important strategy to refine BML in speech recognition. Furthermore, we can see that global EM and partial EM can be combined to yield even better performance, as in the case of *BML5*. At last, we also compare different initialization strategies (WD versus SB) in functional gradient methods as shown between *BML5* and *BML6*. Experimental results show that both weight decay (*BLM5*) and sampling boosting (*BML6*) are quite effective in BML, but weight decay slightly outperforms sampling boosting in this task. Therefore, in the following experiments we will only examine the best configuration, i.e., *BML5*.

2) Performance Comparison With BIC: In this section, we summarize the above-mentioned experiments and combine them with BIC for parsimonious modeling. In Table V, we list the recognition performance of HTK and BML when combined with BIC. With different settings of control parameter λ, the average number of Gaussians per tied state is generated. From these results, we can draw the conclusion that it can make a good balance between the recognition performance and model size to combine HTK or BML with BIC. In the case of "BML+BIC," when λ is set to 0.9, WER is 4.50 with an averaged of 8.8 Gaussians per tied state. Compared with the best BML performance with $K = 10$ in Table IV, the model size has been significantly reduced (12% relative reduction) without any loss in recognition performance. On the other hand, when λ is set to 8.0, the model is extremely compressed to an average of 2.3 Gaussians per tied state, but the corresponding recognition performance is still maintained at a high level compared with the results with $K = 2$ and $K = 3$ in Table IV. All these results show that BIC is really effective for parsimonious modeling.

[1][Online]. Available: http://www.inference.phy.cam.ac.uk/kv227/htk/

B. Experimental Results in Switchboard Task

For the Switchboard task, the training data set consists of two parts: Switchboard-1 Phase-1 Release 2 [25] and Call Home English (CHE). The Switchboard set is a large multi-speaker corpus of conversational telephony speech including 2435 conversations of 241 female and 302 male speakers. Each conversation is double-side and about 3–10 minutes in duration. The training set amounts to 250 hours speech data in total. The transcriptions of the Switchboard data are available from Mississippi State University. The CHE database has 200 conversations with about 10 minutes for each conversation, among which only 120 conversations are selected for training, accounting for about 20 hours of speech data in total. To make a balance between these two databases, three-fold CHE data is mixed with Switchboard data. The test set is 1998 Hub-5 evaluation released by NIST, which consists of 20 conversations from Switchboard-2 Phase-2 and 20 conversations from CHE, about 3 hours data in total. The standard HTK toolkit is used to build cross-word triphone HMMs with a total number of 9000 tied-states using 39-dimensional PLP features with side-based cepstral mean and variance normalization. The same methods as in [26] are used to build the test dictionary and trigram language models. We use the standard NIST scoring software [27] to calculate recognition performance for this task.

In this section, we further examine the effects of BML on the Switchboard task compared with the conventional training procedure on the above 1998 evaluation set. Experimental results are summarized in Table VI. The fourth row of this table marked with "p-value" is the minimum value of p which can find a significant difference at the level of p in the statistical significance tests for HTK and BML systems. Here we adopt the "Matched Pair Test" method mentioned in [27]. The significance test is a two-tailed test with the null hypothesis that there is no performance difference between the two systems. The smaller "p-value" is, the bigger significant differences between two systems are. Based on the results in Table VI, we can observe that BML is more effective than HTK for small mixture numbers (K =2, 3, 4, 5) but performance of bigger model sizes is almost the same as that of HTK (K =6, 7, 8). We analyze the reason of this observation from the Switchboard task itself.

TABLE IV
RECOGNITION PERFORMANCE (WER IN %) COMPARISON AMONG DIFFERENT BML STRATEGIES ON THE WSJ-5k TEST SET

WER(%)	K=1	K=2	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10
HTK	11.02	9.90	8.59	8.11	7.02	5.92	5.34	5.10	5.06	5.23
BML1	-	10.98	10.52	10.22	8.80	8.76	8.87	8.07	7.66	7.55
BML2	-	10.89	10.20	9.66	9.34	8.63	8.00	7.49	7.08	6.74
BML3	-	9.14	7.51	7.04	7.01	6.91	6.93	6.93	6.93	6.93
BML4	-	8.07	6.63	5.81	5.47	5.23	5.01	5.10	4.89	5.01
BML5	-	7.53	6.28	5.72	5.53	5.23	5.03	4.75	4.69	4.50
BML6	-	8.14	6.52	5.81	5.53	5.32	4.97	4.84	4.99	5.08

TABLE V
RECOGNITION PERFORMANCE (WORD ERROR RATE) OF BASELINE AND BML WITH BIC ON THE WSJ-5k TEST SET

	HTK+BIC					BML5+BIC				
λ	0.9	2.0	3.0	5.0	8.0	0.9	2.0	3.0	5.0	8.0
Avg. # of Gaussians	9.5	6.4	4.7	3.2	2.5	8.8	5.2	3.8	2.7	2.3
WER(%)	5.04	5.16	5.34	5.70	6.39	4.50	4.75	4.88	5.47	5.55

TABLE VI
PERFORMANCE (WORD ERROR RATE IN %) COMPARISON BETWEEN HTK BASELINE AND BML IN SWITCHBOARD TEST SET. THE P-VALUE ROWS INDICATE SIGNIFICANCE TEST RESULTS BETWEEN BML AND HTK BASELINE

	K=1	K=2	K=3	K=4	K=5	K=6	K=7	K=8
HTK	66.0	63.9	61.0	58.0	55.6	54.1	52.9	52.2
BML	-	61.0	58.1	56.1	54.8	53.8	52.9	52.5
p-value	-	<0.001	<0.001	<0.001	<0.001	0.215	0.697	0.653
Iterative-BML	-	59.7	56.1	54.4	53.2	52.6	52.1	51.4
p-value	-	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
Iterative-BML + BIC	Avg. 7.1 Gaussians per state, WER is 51.0%							

It is well known that the Switchboard task is one of the most challenging LVCSR tasks. The speech is conversational and spontaneous, with many instances of the so-called disfluencies such as partial words, hesitation and repairs, etc. Another major problem of Switchboard is poor quality of the training transcriptions. Since we accept the Viterbi approximation in BML, this may affect the quality of state segmentation in each iteration. To confirm this idea, an experiment called "Iterative-BML" is designed. "Iterative-BML" denotes that Viterbi state segmentation is regenerated using the best HMMs (with $K = 8$) of "BML" in Table VI, and then the same BML training procedure is repeated to grow HMMs from 1-mix to 8-mix without regenerating the state labels at each stage. As shown in Table VI, with more accurate state segmentation, "Iterative-BML" consistently outperforms "BML" in Table VI. In the sixth row of Table VI, we perform a significance test between "HTK" system and "Iterative-BML" system in Table VI. Now for all cases of mixture number K, the difference between these two systems is significant. When BIC is applied to "Iterative-BML," from Table VI, we can see that model size can be reduced from 8 Gaussians to 7.1 Gaussians per tied state and meanwhile it slightly improves recognition performance from 51.4% to 51.0% in WER.

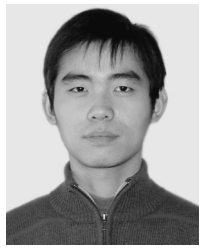
VI. CONCLUSION

In this paper, we have presented a novel boosted mixture learning (BML) framework based on maximum-likelihood (ML) criterion for Gaussian mixture HMMs in speech recognition. To accommodate BML under the HMM framework, the Viterbi approximation has been accepted for state segmentation to extend BML from GMMs to Gaussian mixture HMMs. Several techniques have been proposed to improve performance of BML in speech recognition, such as weight decay to initialize sample weights to avoid overfitting, combination of partial updating and global updating to refine model parameters in BML, and the use of BIC as a convergence criterion in BML for parsimonious modeling. Experimental results on two large-vocabulary ASR tasks, WSJ0 and Switchboard, have shown that the proposed BML method yields significantly better performance than the conventional HMM training procedure. Even though the BML framework is investigated only for the maximum-likelihood criterion in this paper, it is straightforward to extend to other discriminative training criteria to grow Gaussian mixtures discriminatively for speech recognition.

REFERENCES

[1] P. Woodland, J. Odell, V. Valtchev, and S. Young, "Large vocabulary continuous speech recognition using HTK," in *Proc. ICASSP*, 1994, vol. 2, pp. 125–128.
[2] S. Young *et al., The HTK Book (Revised for HTK Version 3.4.1).* Cambridge, U.K.: Cambridge Univ., 2009.
[3] R. Schapire, "The strength of weak learnability," *Mach. Learn.*, vol. 5, pp. 197–227, 1990.
[4] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, no. 55, pp. 119–139, 1997.
[5] R. Schapire, Y. Freund, P. Bartlett, and W. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Ann. Statist.*, vol. 26, pp. 1651–1686, 1997.
[6] R. Schapire, "Theoretical views of boosting and applications," in *Proc. Comput. Learn. Theory*, 1999, pp. 1–10.
[7] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent in function space," *NIPS*, vol. 11, pp. 512–518, 1999.

[8] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Functional gradient techniques for combining hypotheses," in *Advances in Large Margin Classifiers (Edited by A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans)*. Cambridge, MA: MIT Press, 2000, pp. 221–246.
[9] S. Rosset and E. Segal, "Boosting density estimation," *Adv. Neural Inf. Process. Syst.*, vol. 15, pp. 641–648, 2003.
[10] F. Wang, C. Zhang, and N. Lu, "Boosting GMM and its two applications," *Multiple Classifier Syst.*, pp. 12–21, 2005.
[11] H. Tang, S. Chu, M. Hasegawa-Johnson, and T. Huang, "Emotion recognition from speech via boosted Gaussian mixture models," in *Proc. ICME*, 2009, pp. 294–297.
[12] V. Pavlovic, "Model-based motion clustering using boosted mixture modeling," in *Proc. CVPR*, 2004, pp. 811–818.
[13] M. Kim and V. Pavlovic, "A recursive method for discriminative mixture learning," in *Proc. ICML*, 2007, pp. 409–416.
[14] N. Vlassis and A. Likas, "A greedy EM algorithm for Gaussian mixture learning," *Neural Process. Lett.*, vol. 15, no. 1, pp. 77–87, 2002.
[15] N. Ueda, R. Nakano, Z. Ghahramani, and G. Hinton, "SMEM algorithm for mixture models," in *Advances in Neural Information Processing Systems 11*, M. Kearns, S. Solla, and D. Cohn, Eds. Cambridge, MA: MIT Press, 1998.
[16] M. Meek, B. Thiesson, and D. Heckerman, "Staged mixture modeling and boosting," in *Proc. UAI'02*, 2002, pp. 335–343.
[17] S. Rosset, "Robust boosting and its relation to bagging," in *Proc. ACM SIGKDD*, 2005, pp. 249–255.
[18] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
[19] G. Zweig and M. Padmanabhan, "Boosting Gaussian mixtures in an LVCSR system," in *Proc. ICASSP*, 2000, pp. 1527–1530.
[20] C. Dimitrakakis and S. Bengio, "Boosting HMMs with an application to speech recognition," in *Proc. ICASSP*, 2004, vol. 5, pp. 621–624.
[21] C. Meyer and H. Schramm, "Boosting HMM acoustic models in large vocabulary speech recognition," *Speech Commun.*, vol. 48, pp. 532–548, 2006.
[22] G. McLachlan, *Finite Mixture Models*. New York: Wiley, 2001.
[23] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, 1978.
[24] S. Chen, E. Eide, and M. Gales *et al.*, "Recent improvements to IBM's speech recognition system for automatic transcription of broadcast news," in *Proc. ICASSP*, 1999, pp. 37–40.
[25] J. Godfrey, E. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone speech corpus for research and development," in *Proc. ICASSP*, 1992, pp. 517–520.
[26] T. Hain, P. Woodland, T. Niesler, and E. Whittaker, "The 1998 HTK system for transcription of conversational telephone speech," in *Proc. ICASSP*, 1999, pp. 57–60.
[27] D. Pallett, W. Fisher, and J. Fiscus, "Tools for the analysis of benchmark speech recognition tests," in *Proc. ICASSP*, 1990, pp. 97–100.
[28] J. Du, Y. Hu, and H. Jiang, "Boosted mixture learning of Gaussian mixture HMMs for speech recognition," in *Proc. Interspeech*, 2010, pp. 2942–2945.
[29] H. Tang, M. Hasegawa-Johnson, and T. Huang, "Toward robust learning of the Gaussian mixture state emission densities for hidden Markov models," in *Proc. ICASSP*, 2010, pp. 5242–5245.

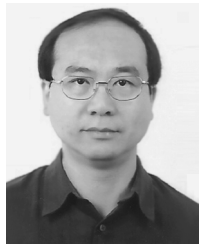


Jun Du received the B.Eng. and Ph.D. degrees from the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, in 2004 and 2009, respectively.

From 2004 to 2009, he was with the iFlytek Speech Lab, USTC, where he conducted research on speech recognition. During the above period, he worked as an Intern twice for nine months at Microsoft Research Asia (MSRA), Beijing, China, doing research on discriminative training and noise robust front-end for speech recognition and speech enhancement. In 2007, he also worked as a Research Assistant for six months at the Department of Computer Science, The University of Hong Kong, doing research on robust speech recognition. From July 2009 to June 2010, he worked at iFlytek Research, Anhui, China, on speech recognition. In July 2010, he joined the Visual Computing Group of MSRA as an Associate Researcher.

Yu Hu received the B.Eng., M.Eng., and Ph.D. degrees, all in electrical engineering, from the University of Science and Technology of China (USTC), Hefei, in 2000, 2003, and 2009, respectively.

In 1999, he became a Research Engineer with iFlytek, Ltd., Anhui, China, as a cofounder, working on Mandarin speech synthesis and speech prosody analysis. He was one of researchers who built first few generations of iFlytek Mandarin speech synthesis engines. Since 2004, his research interest has changed to robust speech recognition, and began to work on the iFlytek Mandarin speech recognition system. He is currently the director of iFlytek Research and working on speech recognition over mobile internet.



Hui Jiang (M'00) received the B.Eng. and M.Eng. degrees from the University of Science and Technology of China (USTC), Hefei, and the Ph.D. degree from the University of Tokyo, Tokyo, Japan, in September 1998, all in electrical engineering.

From October 1998 to April 1999, he worked as a Researcher at the University of Tokyo. From April 1999 to June 2000, he was with Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, as a Postdoctoral Fellow. From 2000 to 2002, he worked in Dialogue Systems Research, Multimedia Communication Research Lab, Bell Labs, Lucent Technologies Inc., Murray Hill, NJ. He joined Department of Computer Science and Engineering, York University, Toronto, ON, Canada, as an Assistant Professor in Fall 2002 and was promoted to Associate Professor in 2007. His current research interests include speech and audio processing, machine learning, statistical data modeling, and bioinformatics, especially discriminative training, robustness, noise reduction, utterance verification, and confidence measures.

Dr. Jiang has served as an Associate Editor for IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING since 2009.