SANDIA REPORT

SAND2012-2398
Unlimited Release
Printed April 2012

Efficient Speaker Verification Using Gaussian Mixture Model Component Clustering

Richard D. McClanahan, Phillip L. De Leon

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2012-2398
Unlimited Release
Printed April 2012

Efficient Speaker Verification Using Gaussian Mixture Model Component Clustering

Richard D. McClanahan
Radar and Signal Analysis Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-9999
rmcclan@sandia.gov

Dr. Phillip L. De Leon
New Mexico State University
Las Cruces, NM
pdeleon@nmsu.edu

Abstract

In speaker verification (SV) systems that employ a support vector machine (SVM) classifier to make decisions on a supervector derived from Gaussian mixture model (GMM) component mean vectors, a significant portion of the computational load is involved in the calculation of the *a posteriori* probability of the feature vectors of the speaker under test with respect to the individual component densities of the universal background model (UBM). Further, the calculation of the sufficient statistics for the weight, mean, and covariance parameters derived from these same feature vectors also contribute a substantial amount of processing load to the SV system. In this paper, we propose a method that utilizes clusters of GMM-UBM mixture component densities in order to reduce the computational load required. In the adaptation step we score the feature vectors against the clusters and calculate the *a posteriori* probabilities and update the statistics exclusively for mixture components belonging to appropriate clusters.

Each cluster is a grouping of multivariate normal distributions and is modeled by a single multivariate distribution. As such, the set of multivariate normal distributions representing the different clusters also form a GMM. This GMM is referred to as a hash GMM which can be considered to a lower resolution representation of the GMM-UBM. The mapping that associates the components of the hash GMM with components of the original GMM-UBM is referred to as a shortlist.

This research investigates various methods of clustering the components of the GMM-UBM and forming hash GMMs. Of five different methods that are presented one method, Gaussian mixture reduction as proposed by Runnall's [24], easily outperformed

the other methods. This method of Gaussian reduction iteratively reduces the size of a GMM by successively merging pairs of component densities. Pairs are selected for merger by using a Kullback-Leibler based metric.

Using Runnal's method of reduction, we were able to achieve a factor of 2.77 reduction in *a posteriori* probability calculations with no loss in accuracy when the original UBM consisted of 256 component densities. When clustering was implemented with a 1024 component UBM, we achieved a computation reduction of 5 with no loss in accuracy and a reduction by a factor of 10 with less than 2.4% relative loss in accuracy.

Contents

1 Introduction . . . 7
2 Introduction to Speaker Verification . . . 9
2.1 Introduction . . . 9
2.2 Frontend Processing and MFCC Based Feature Vectors . . . 10
2.3 SV System Training . . . 13
2.4 SV System Testing . . . 19
3 GMM Component Clustering . . . 20
3.1 Introduction . . . 20
3.2 Hash GMMs for Improved Scoring . . . 20
3.3 Methods for Forming Hash GMMs . . . 22
4 Results of GMM Component Clustering . . . 28
4.1 Introduction . . . 28
4.2 Data Sets and Simulation Parameters . . . 28
4.3 Performance Metric . . . 28
4.4 Experiments and Results . . . 29
5 Conclusions and Future Work . . . 43
5.1 Conclusions . . . 43
5.2 Future Work . . . 43
References . . . 44

Figures

1 Speaker Verification System. . . . 9
2 Frontend Processing/Feature Extraction used in SI/SV System . . . 10
3 Triangular Weighting Functions for MFCC Computations . . . 11
4 Frequency Response of Bandpass Filter Used in RASTA Processing . . . 12
5 Steps involved in training the SVM using GMM supervectors SV system [9]. . . 13
6 MAP Adaptation of UBM Mixture Components. . . . 16
7 Support Vector Machine as a Sparse Maximum Margin Classifier [4]. . . . 18
8 Clustering the components of a GMM-UBM with a large number of components (blue) into a hash GMM with a smaller number of components (red) . . 21
9 Results for Applying Clustering Method to a 256 Component UBM for both Training and Testing . . . 33
10 Results for Applying Clustering Method to a 256 Component UBM for Testing Only . . . 34
11 Results for Applying Clustering Method to a 1024 Component UBM for both Training and Testing . . . 36
12 Results for Applying Clustering Method to a 1024 Component UBM for Testing Only . . . 37

Tables

1 Results for Clustering based on *a posteriori* Probabilities of Training Vectors with a UBM of 256 Components 30
2 Results for K-Means Clustering with a UBM of 256 Components 31
3 Results for Normalized Mean Clustering with a UBM of 256 Components . . . 32
4 Results for Smaller UBM Mapping with a UBM of 256 Components 32
5 Results for Runnalls GMM Reduction with a UBM of 256 Components 35
6 Results for Clustering based on *a posteriori* Probabilities of Training Vectors with a UBM of 1024 Components 38
7 Results for K-Means Clustering with a UBM of 1024 Components 39
8 Results for Normalized Mean Clustering with a UBM of 1024 Components . . 40
9 Results for Smaller UBM Mapping with a UBM of 1024 Components 41
10 Results for Runnalls GMM Reduction with a UBM of 1024 Components 42

1 Introduction

Speaker recognition (SR) can be broken down into two tasks: speaker verification (SV) and speaker identification (SI). In SV systems the task is to determine whether a person is who he/she claims to be. In SI systems there is no claim of identity for the unknown speaker and so the system must determine who is talking from a set of known speakers. For an SI trained with N_{SI} unique speakers, the SI task of determining which of these N_{SI} speakers spoke the utterance is a $1 : N_{SI}$ classification problem [20]

More recently, some research has examined ways to reduce the required computation of SR systems without sacrificing accuracy—which will always be the most important factor. Computational reduction in SR systems is aimed at the test-stage where fast recognition or low power consumption (in embedded applications) may be important factors. Since training an SR system is normally a one-time, up-front cost, emphasis is not normally placed on fast training. In fact, it may be argued that increasing training time for potentially faster test-stage time is an acceptable trade-off.

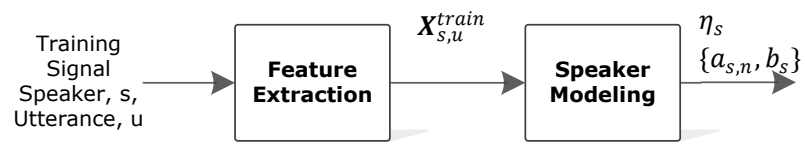
Much of the research on reducing computation in SR has focused on SI. The reason being that in the test-stage of a Gaussian Mixture Model (GMM) SI system, a maximum *a posteriori* (MAP) decision is made to identify the unknown speaker. The MAP decision requires a likelihood calculation of an unknown speaker's test feature vectors for *each* speaker model in the system. In SI systems with a large number of speaker models, the MAP decision is the recognized bottleneck [13], [1].

A number of different methods have also been proposed to reduce the computation in GMM-UBM SV systems where one of the primary computational bottlenecks is in the number of *a posteriori* probability calculations required in the log likelihood ratio. For example, a method was proposed by Sarkar to reduce the number of features being evaluated by eliminating those with substantial redundancy [25, 26]. Depending on the corpus and distance metric used to measure redundancy, they were able to achieve little to no performance degradation with a frame rate reduced by a factor of 4. Auckenthaler and Mason used hash tables [2] and were able to reduce the computational burden by creating a shortlist between a smaller hash GMM and the full component size GMM achieving a processing reduction factor of about 6 with "no noticeable performance degradation." The approach by Xiang and Berger [27] was to generate a structural background model (SBM) and structural GMMs (SGMM) for the target speakers. The SBM and SGMM were multilayered GMMs that could be considered GMMs of different resolution. The SGMM-SBM method achieved a computational reduction by a factor of 17 with a 5% reduction in equal error rate (EER).

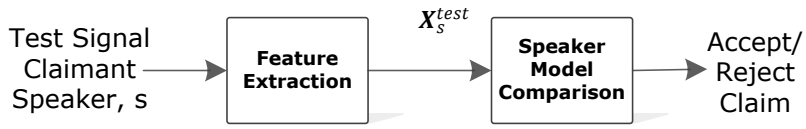
The focus of this dissertation is on how to reduce the computational load in the SV test-stage of a state-of-the-art, support vector machine (SVM) using a GMM supervectors system. We propose several different methods of creating a hash GMM. The first method is similar to the GS1 hash of [2] but with a different method of generating shortlists which map the component densities from the hash GMM to components of the GMM-UBM. We use a method based on the Kullback-Leibler (KL) divergence between the components of

the hash GMMs and the GMM-UBM components to generate shortlists for the hash GMMs. Two other hash generation methods we analyze are based on the idea of "GMM reduction." Our approach differs from Auckenthaler and Xiang [2, 27] primarily in that we use a new method of GMM reduction for creating the hash GMM and that we are using a SVM-based SV system described by Campbell [5] rather than the GMM-UBM system.

The remainder of the dissertation is organized as follows. In Chapter 2 we review SV systems based on SVM using GMM supervectors. Next, in Chapter 3 we present several methods of clustering the GMM-UBM component densities to create a hash GMM. In Chapter 4 we present the results that we have obtained using these various methods. Finally, Chapter 5 presents the conclusions from this work as well as proposes possible future work.



(a) SV Training



(b) SV Testing

Figure 1. Speaker Verification System.

2 Introduction to Speaker Verification

2.1 Introduction

While the task of SI and SV may be different, SV verifies a single claim of identity whereas SI tries to determine a speaker's identity from a possibly large set of potential speakers, both involve the same two major stages: training and testing. Whether training for SI or SV, we assume that we have access to voice samples from all speakers that will be identified or verified in the test stage. Fig. 1(a) shows that within the training stage, the SV system first calculates feature vectors from the training speech utterances. Next speaker models are fit to the feature vectors. Within the test stage, Fig. 1(b), a speech utterance and claim of identity are provided and again the first step in processing is to calculate feature vectors. From these feature vectors the SV system must make a decision to accept/reject the identity claim based on the voiced sample.

As the process of generating feature vectors is common to both the training and testing stages, the process used to calculate the feature vectors is first examined. Next, the training stage is examined including the creation of a UBM. Finally, the test stage of the SVM based SV system is presented.

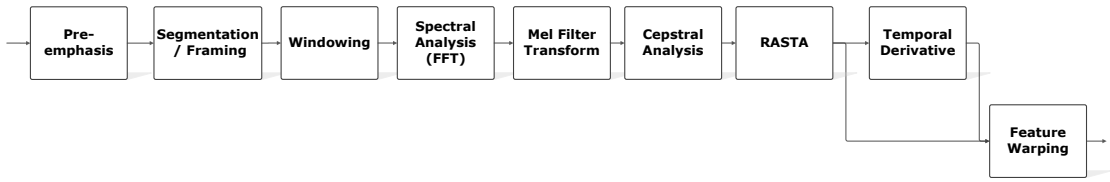


Figure 2. Frontend Processing/Feature Extraction used in SI/SV System

2.2 Frontend Processing and MFCC Based Feature Vectors

Many modern SI/SV systems use mel-frequency cepstral coefficients (MFCCs) as a form of parametric representation of the speech signal. The MFCC representation has several key characteristics. The first is that it is a more compact representation than the discrete Fourier transform (DFT) and thus provides for dimensional reduction. A second characteristic is that it segments the frequency spectrum into critical bands using mel frequency filters. The mel filters are designed to be representative of the human auditory system. Finally, MFCCs are formed using homomorphic processing. A homomorphic process is a process that transforms convolution into addition which can simplify deconvolution. In particular, for SI/SV applications, this provides a means of decoupling the excitation source as well as channel effects while maintaining information about the time-varying vocal tract.

Fig. 2 depicts the different processing steps that are used in the feature extraction stage. The first step in feature extraction is often pre-emphasis of the speech signal. The pre-emphasis filters suppresses frequencies below about 200 Hz as there is little important speech energy below these frequencies [3, p. 120]. The filter is a simple high pass filter of the form

$$s_{pr}[n] = s_{in}[n] - a s_{pr}[n-1] \tag{1}$$

where $s_{in}[n]$ is the input speech signal, $s_{pr}[n]$ is the pre-emphasized signal, and a is the feedback coefficient controlling the cutoff frequency of the high-pass filter. Typically a takes on a value of around $0.9 - 1.0$ This work uses a value of 0.97.

The pre-emphasized speech signal is then analyzed using time-frequency analysis by means of a short-time Fourier transform. This is represented by the three blocks of Segmentation, Windowing, and Spectral analysis in Fig. 2. The STFT of the speech signal is calculated by

$$S_{\hat{n}}(e^{j\hat{\omega}}) = \sum_{m=-\infty}^{\infty} w[m] s_{pr}[\hat{n}-m] e^{j\hat{\omega}(\hat{n}-m)} \tag{2}$$

where $w[m]$ is the window function.

Next, mel-frequency cepstrum coefficients are calculated. This is done in the blocks labeled as Mel Filter Transform and Cepstral analysis in Fig. 2. The mel cepstrum was for-

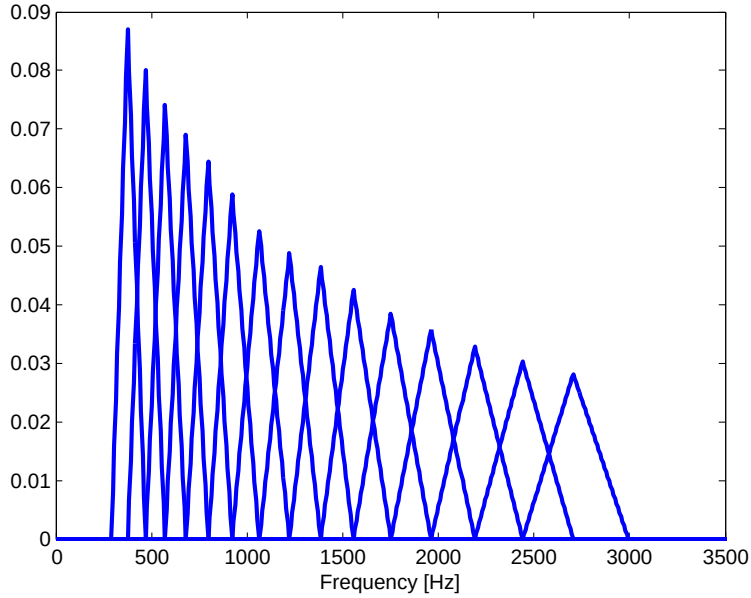


Figure 3. Triangular Weighting Functions for MFCC Computations

mulated by Davis and Mermelstein [8] in order to add the effects of human audio perception to cepstral processing [7]. The mel filters are a set of triangular filters that are logarimically spaced to group the DFT values of the STFT into critical bands [19]. The filters are linearly spaced in the mel scale which is found by the equation

$$pitch\ in\ mels = 1127 log_e(1 + F/700) \tag{3}$$

A set of example mel filters is shown in Fig. 3. The mel-spectrum is computed for the mth frame as

$$MF_m[r] = \frac{1}{A_r} \sum_{k=L_r}^{U_r} |V_r[k] S_m[k]|^2 \tag{4}$$

where $V_r[k]$ is the rth filter ranging from DFT index L_r to index U_r. The term A_r is a normalization term for the rth filter and is found as

$$A_r = \frac{1}{A_r} \sum_{k=L_r}^{U_r} |V_r[k]|^2 \tag{5}$$

The MFCCs are computed by calculating the cepstrum of the mel-spectrum

$$mfcc_m[n] = \frac{1}{R} \sum_{r=1}^{R} \log\left(MF_m[r]\right) \cos\left[\frac{2\pi}{R}\left(r + 1/2\right) n\right] \tag{6}$$

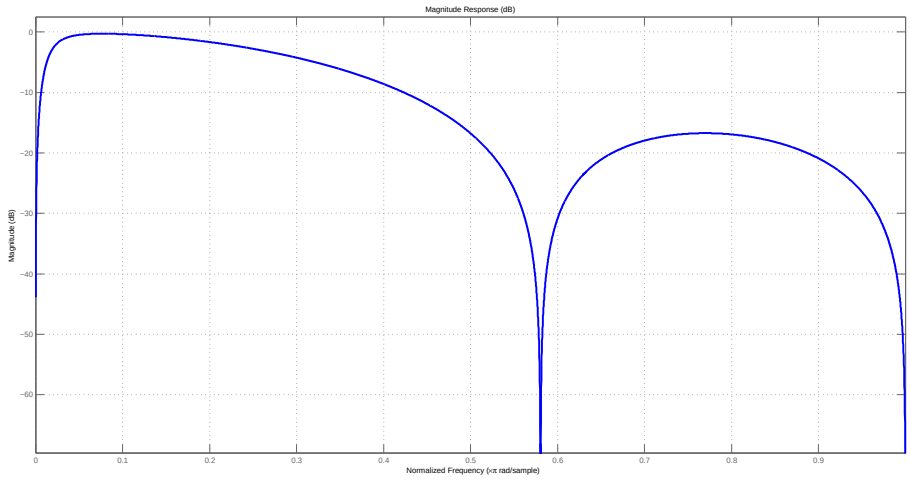


Figure 4. Frequency Response of Bandpass Filter Used in RASTA Processing

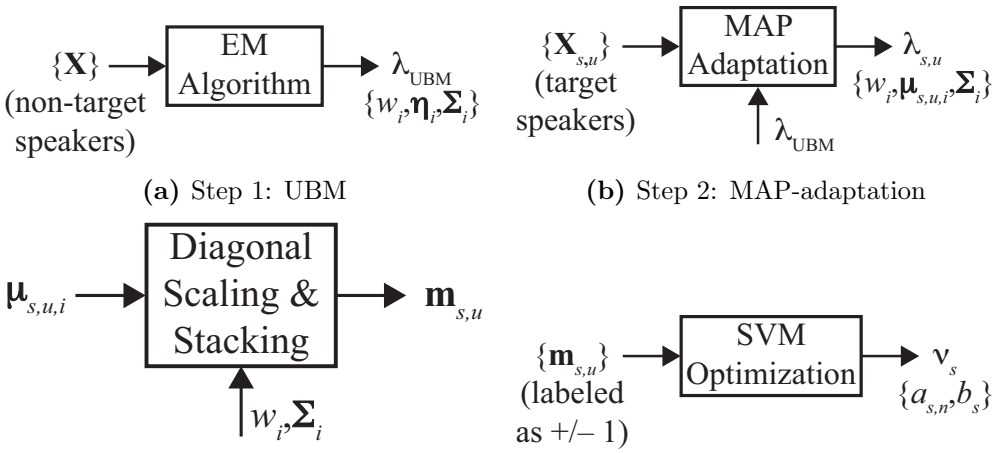
Or more simply, $MFCC = DCT(LOG(MF))$.

The MFCC coefficients are then processed with RASTA channel compensation [12]. The purpose of RASTA processing is to suppress the effect of convolutional noise sources that might be present in a speech signal. For example switching from one microphone to another where both have linear characteristics. The convolutional noise becomes additive in the log domain from cepstral processing. In the RASTA method, each frequency channel is band-pass filtered in the time domain with a band-pass filter with a notch at DC. If the channel characteristics vary much slower than the speech signal characteristics, this notch will suppress slow variations in the short-term spectrum due to the slow varying channel. The filter that is used in the RASTA processing step is shown in Fig. 4.

After RASTA processing, a temporal derivative of the data is calculated within each frequency channel. Clearly, speech is a time varying signal and as such it would seem reasonable to try to capture information about this dynamic nature. Delta-cepstral coefficients are used in an attempt to capture some of this dynamic information. The delta-cepstrum is an approximation to the first derivative while the delta-delta cepstrum is an estimate of the second derivative. In this work, we have only made use of the delta cepstrum. Typically, the delta cepstrum is computed with a locally smoothed estimate of the first derivative as

$$\Delta mfcc_m[n] = \frac{\sum_{k=-M}^{M} k(mfcc_{m+k}[n])}{\sum_{k=-M}^{M} k^2} \tag{7}$$

The final processing step shown in Fig. 2 is feature warping. The purpose of feature warp-



(a) Step 1: UBM **(b)** Step 2: MAP-adaptation

(c) Step 3: Supervectors **(d)** Step 4: SVM Training

Figure 5. Steps involved in training the SVM using GMM supervectors SV system [9].

ing is to enhance features and make them more robust to channel effects. While RASTA processing is used to remove linear convolution effects, it is not robust to additive noise. Feature warping attempts to mitigate both linear channel effects as well as additive channel noise [17]. In particular feature warping attempts to exploit the effect that additive noise generally reduces the variance of the cepstral feature parameters. The distribution of the cepstral features is mapped to a normal distribution (though the ideal distribution is multimodal). The result is that whether or not additive noise is present, the resulting feature warped distribution is very similar whereas the input distributions may have been significantly different.

The final output of the frontend processing is a set of feature vectors that contain MFCCs and delta-MFCCs that that have been processed with feature warping.

2.3 SV System Training

In this section, we briefly review the training stage of the SVM system using GMM supervectors as described in [5]. Training the SV system is achieved in four steps as depicted in Fig. 5. The first step is constructing a GMM-UBM. The second step is MAP-adaptation of the UBM to target speakers. The third step is constructing supervectors from the MAP-adapted GMMs. In the final step, SVMs are trained to target speakers.

UBM Construction

The first step in training, as depicted in Fig. 5(a), consists of constructing a GMM-UBM using the feature vectors from a large collection of non-target speakers. The GMM-UBM is represented by the model parameters $\lambda_{\text{UBM}} = \{w_i, \boldsymbol{\eta_i}, \boldsymbol{\Sigma}_i\}$ which are the weight, mean vector, and diagonal covariance matrix respectively for the i-th component density where $1 \leq i \leq M$ and M is the number of components densities of the GMM (typically 1024 or 2048). For a D-dimensional feature vector, $\mathbf{x}$, the GMM which is a superposition of multivariate Gaussian component densities is defined as

$$p(\mathbf{x}|\lambda) = \sum_{i=1}^{M} w_i p_i(\mathbf{x}). \tag{8}$$

Each of the individual densities $p_i(\mathbf{x})$ is itself a multivariate Gaussian distribution with mean $\boldsymbol{\eta_i}$ and covariance $\boldsymbol{\Sigma}_i$ of the form

$$p_i(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_i|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\eta_i})'(\boldsymbol{\Sigma}_i)^{-1}(\mathbf{x}-\boldsymbol{\eta_i})}. \tag{9}$$

Typically, parameters of the the GMM-UBM are estimated using the expectation-maximization (EM) technique [22]. EM is an iterative two step technique that first calculates the expectation values for membership values (the *a posteriori* probability) for each feature vector

$$Pr(i|\mathbf{x}_t, \lambda) = \frac{w_i p_i(\mathbf{x}_t|\boldsymbol{\eta}_i, \boldsymbol{\Sigma}_i)}{\sum_{k=1}^{M} w_k p_k(\mathbf{x}_t|\boldsymbol{\eta}_k, \boldsymbol{\Sigma}_k)} \tag{10}$$

. The second step re-estimates the sufficient statistics parameters

$$\begin{aligned} \hat{w}_i &= \frac{1}{T}\sum_{t=1}^{T} Pr(i|\mathbf{x}_t, \lambda) \\ \hat{\boldsymbol{\eta}}_i &= \frac{\sum_{t=1}^{T} Pr(i|\mathbf{x}_t, \lambda)\mathbf{x}_t}{\sum_{t=1}^{T} Pr(i|\mathbf{x}_t, \lambda)} \\ \hat{\boldsymbol{\Sigma}}_i &= \frac{\sum_{t=1}^{T} Pr(i|\mathbf{x}_t, \lambda)\mathbf{x}_t^2}{\sum_{t=1}^{T} Pr(i|\mathbf{x}_t, \lambda)} - \hat{\boldsymbol{\eta}}_i^2 \end{aligned} \tag{11}$$

where T is the number of training feature vectors. This two step process of alternately calculating the expectation (10) and updating the the sufficient statistics (11) is iterated until some convergence criterion is met (maximum number of iterations and/or percentage change in log-likelihood).

As a result, the GMM-UBM training has estimated a GMM with a substantial number of mixture components (typically 2048 components) to fit to the training feature vectors what were extracted from voice samples in a large number of speakers. Further, the UBM

is often trained using speech samples that match the conditions of the test data. If the test environment/speaker characteristics are unknown, then it is common to use training data that is a diverse as possible in terms of channels, microphones, speaker dialects, etc...

In our system, the background model has been created by training GMMs to subpopulations of the training speaker data [21]. These GMMs are then combined to form a single GMM-UBM. We have trained separate GMMs for male/female speakers from different corpora. For example with two corpora we have trained four GMMs (male and female for each corpus). To merge the GMM trained on each subpopulation we have merged the sets of mean vectors and covariance matrices. We have assumed that each of the populations is equally likely and therefore have simply scaled the weights by the reciprocal of the number of subpopulations, $w_i = w_{i_{subpop}}/4$.

MAP Adaptation

In the second step, Fig. 5(b), feature vectors extracted from target speakers' utterances are used to MAP-adapt the mean vectors of the GMM-UBM using a form of Bayesian adaptation [23]. While the typical maximum likelihood training method trains models for each speaker independently, the adapted UBM approach derives a speakers model by updating a pre-trained UBM model. This MAP adaptation provides a tighter coupling between the UBM and speaker adapted GMM.

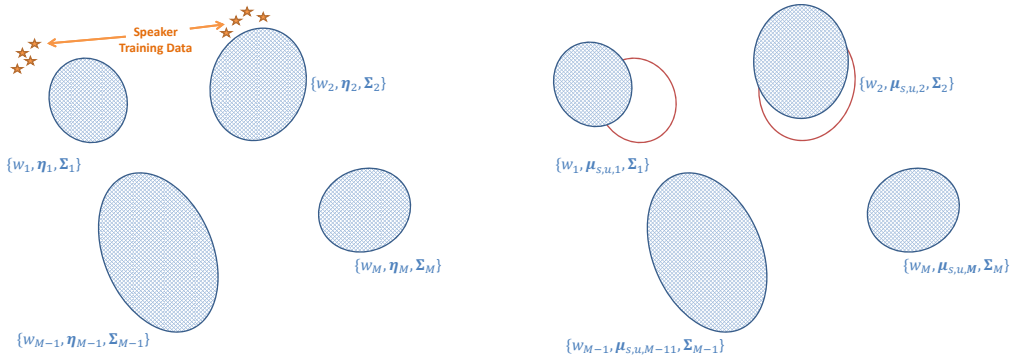
The MAP adaptation is a two step process that is very similar to the EM algorithm used for training the GMM-UBM. The MAP-adapted speaker model is denoted $\lambda_{s,u} = \{w_i, \boldsymbol{\mu}_{s,u,i}, \boldsymbol{\Sigma}_i\}$ in which $\boldsymbol{\mu}_{s,u,i}$ is the MAP-adapted mean vector for the i-th component density from utterance u of speaker s. The model, $\lambda_{s,u}$, is a GMM with the same weights w_i and covariance matrices $\boldsymbol{\Sigma}_i$ as the GMM-UBM but with mean vectors that have been adapted.

The first step of the MAP adaptation is to calculate the sufficient statistics just as would be done in the EM algorithm. First, the probabilistic alignment of the feature vectors into the UBM mixture components are calculated with

$$Pr(i|\mathbf{x}_t) = \frac{w_i p_i(\mathbf{x}_t)}{\sum_{j=1}^{M} w_j p_j(\mathbf{x}_t)} \tag{12}$$

where

$$p_i(\mathbf{x}) = \frac{1}{(2\pi)^{D/2} \left|\boldsymbol{\Sigma}_i\right|^{1/2}} e^{-(1/2)(\mathbf{x}-\boldsymbol{\mu}_\mathbf{i})'\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_\mathbf{i})}. \tag{13}$$



(a) GMM Components Before MAP Adaptation

(b) GMM Components After MAP Adapatation

Figure 6. MAP Adaptation of UBM Mixture Components.

Next, the sufficient statistics for the weights and mean vectors are computed using

$$n_i = \sum_{t=1}^{T} Pr(i|\mathbf{x}_t)$$
$$E_i(\mathbf{x}) = \frac{1}{n_i} \sum_{t=1}^{T} Pr(i|\mathbf{x}_t)\mathbf{x}_t \tag{14}$$

These new estimates of the sufficient statistics are then used to update the statistics from the GMM-UBM. The sufficient statistics for mixture i, utterance u, and speaker s are updated by

$$\boldsymbol{\mu}_{s,u,i} = \alpha_i^m E_i(\mathbf{x}) + (1 - \alpha_i^m)\boldsymbol{\eta_i} \tag{15}$$

. The adaptation coefficient α_i^m controls the degree to which the new statistics are used to update the old statistics. The closer α_i^m is to 1 the more the new statistics affect the MAP adapted model. The adaptation coefficient is calculated using the sufficient statistics for the weight, n_i,

$$\alpha_i^m = \frac{n_i}{n_i + \rho^m} \tag{16}$$

where ρ^m is a fixed scalar that is referred to as the relevance parameter. The superscript m in α_i^m and ρ^m is used to denote that the adaptation coefficient and relevance parameter are those used for the mean vector–as opposed to relevance parameters for updating the weights or covariance matrices. Often, the relevance parameter has been assigned the value of 16 in literature [23].

Fig. 6 shows the how that MAP adaptation shifts the means of the mixture densities to more closely match target speaker statistics.

In general, the weights, w_i, and covariance matrices, $\mathbf{\Sigma}_i$ could also be adapted but for this supervector SVM based method we do not do so following [23].

It is assumed that we have several utterances available for each target speaker. This assumption is necessary in order to adequately train the SVM system as described in Section 2.3. If this is not the case, then it may be possible to split a larger utterance into several smaller utterances for training.

Supervector Construction

Within the third step, Fig. 5(c), the adapted mean vectors $\boldsymbol{\mu}_{s,u,i}$ are scaled

$$\mathbf{m}_{s,u,i} = \sqrt{w_i}\mathbf{\Sigma}_i^{-1/2}\boldsymbol{\mu}_{s,u,i} \tag{17}$$

and then used to form GMM supervectors. There is a separate supervector for each speaker and utterance

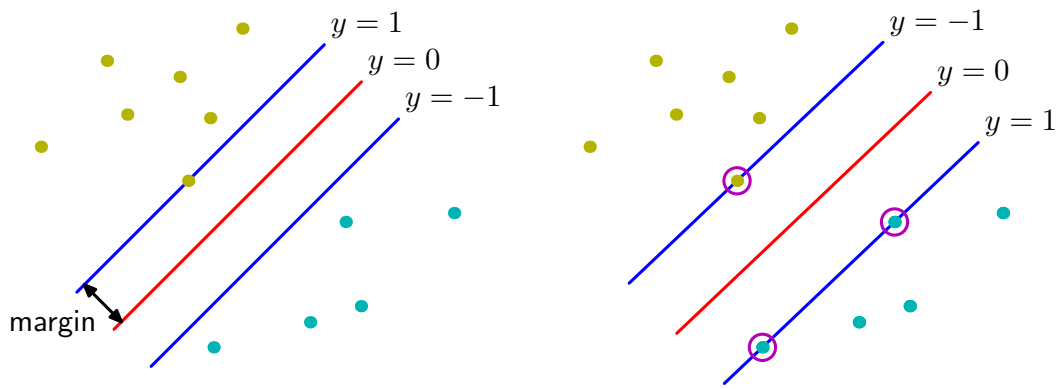
$$\mathbf{m}_{s,u} = \begin{bmatrix} \mathbf{m}_{s,u,1} \\ \vdots \\ \mathbf{m}_{s,u,M} \end{bmatrix} \tag{18}$$

. Each supervector is essentially a mapping from a speaker's utterance to a high-dimensional vector.

SVM Training

In the fourth step, Fig. 5(d), a SVM is trained to discriminate the target speaker. SVMs have several key characteristics. First, the SVM is a maximum boundary classifier. In training the SVM, the decision boundary is found by maximizing the perpendicular distance between the decision boundary and the data points closest to it. This perpendicular margin is depicted in Fig. 7(a). A second key characteristic is that SVMs are sparse kernel classifiers. They are sparse in that they use only a small subset of the training data points in a kernel function to make classification decisions. Fig. 7(b) shows that support vectors–or the data points retained in the kernel function–as well as the decision boundary that maximized the margin for the data in the figure.

The SVM is trained using a linear kernel [6], with weight and bias parameters a_n and b. Suppose we denote our kernel function as $K(u_a, u_b)$ where u_a and u_b are two different utterances to compare. The linear kernel is simply the inner product of the two vectors $K(u_a, u_b) = \boldsymbol{m}_{u,a}^t \boldsymbol{m}_{u,b}$ where $\boldsymbol{m}_{u,a}$ and $\boldsymbol{m}_{u,b}$ represent the supervectors from each utterance.



(a) SVM Margin

(b) SVM Support Vectors

Figure 7. Support Vector Machine as a Sparse Maximum Margin Classifier [4].

We have used a soft margin SVM–the SVM training algorithm allows for misclassification of training data. Training the SVMs is accomplished by minimizing [4]

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m K(\boldsymbol{m}_{u,n}, \boldsymbol{m}_{u,m}) \tag{19}$$

constrained to

$$\begin{aligned} 0 \leq a_n \leq C \\ \sum_{n=1}^{N} a_n t_n = 0 \end{aligned} \tag{20}$$

where C controls the soft penalizing of points that lie on the wrong side of the margin. The values a_n are the weights of the support vectors. Because SVMs are sparse machines, the vast majority of a_n will be equal to 0.

The SVM training system uses supervectors from the target speaker as well as a significant number of background non-target speakers. The supervectors for the target speaker are labeled $+1$ ($t_n = +1$ whereas the supervectors for all other background speakers are labeled -1 ($t_n = -1$). The resulting SVM speaker model is denoted $\nu_s = \{a_{s,n}, b_s\}$ where $a_{s,n}$ is the weight of the n-th support vector, b_s is the bias, and $n \in \mathcal{S}$ and $\mathcal{S}$ is set of indices of the support vectors

2.4 SV System Testing

In the SV test stage, we are given a speech utterance and an identity claim C. We must decide whether to accept or reject the claim. To do so we extract the feature vectors from the utterance as described in Section 2.2 and form a supervector $\mathbf{m}_{\text{test}}$ following the same procedure as in Sections 2.3 and 2.3. The supervector, $\mathbf{m}_{\text{test}}$, is evaluated against the model SVM by computing

$$y(\mathbf{X}) = \sum_{n \in \mathcal{S}} a_{C,n} t_{C,n} \mathbf{m}_{\text{test}}^T \mathbf{m}_n + b_C \tag{21}$$

where $t_{C,n}$ denotes the labels associated with the support vectors and $\mathbf{m}_n$ are the support vectors (supervectors). The claim is accepted if $y(\mathbf{X}) \geq 0$.

This decision is a binary decision though some research has gone into quantifying the quality (i.e. determining a confidence or probability) of the output [18]. It is also important to note that the SVM is a two-class classifier. As such it is well suited for the SV system (yes/no it is the claimant) but not as well suited for the SI system (choose one among many).

3 GMM Component Clustering

3.1 Introduction

Within the testing stage, a significant portion of the processing load is required during the calculation of the probabilistic alignment between the input feature vectors and the UBM mixture components in (12) and the calculation of the sufficient statistics (14). This is significant because this calculation which includes a mahanobis distance calculation must be performed for each input feature vector into each of GMM-UBM component densities. Clearly, one way to speed up the likelihood calculations is to not perform the calculation for the entire set of mixture component densities. Effectively, we only calculate the probabilistic alignments for a subset of mixture components, $\mathcal{C}$, that we have chosen wisely.

The principle behind the hash GMM [2] is that it attempts to model the same probabilistic features that the GMM-UBM models except that it does so with fewer component densities. A shortlist is created that maps clusters of components within the GMM-UBM to component densities within the hash GMM. This shortlist is formed by assigning components with the GMM-UBM to components of the hash GMM using a statistical divergence measure.

In Section 3.2 the general concept of using hash GMMs and a shortlist to improve computational efficiency is described. This is followed by Sections 3.3-3.3 in which three different methods are proposed for creating hash GMMs. These methods include training hash GMMs using EM on the original training data used to train the UBM, k-Means clustering of the GMM-UBM densities, and GMM reduction as described by Runnalls [24].

3.2 Hash GMMs for Improved Scoring

The goal of using a hash GMM with shortlists mappings to components within the GMM-UBM is ultimately to reduce to total number of component likelihood calculations and the number of components used to update the sufficient statistics for each feature vector. Figure 8 shows the principle of clustering components of the GMM-UBM as depicted in blue on the left hand side of the figure and forming new components of the hash GMM as depicted in red on the right hand side.

Within the GMM-UBM there are M multivariate Gaussian component densities each having the model parameters $\{w_i, \eta_i, \mathbf{\Sigma}_i\}$–the weight, mean vector, and covariance matrix parameters respectively. A hash GMM is formed with M_c components where $M_c < M$ with model parameters $\{w_i^h, \eta_i^h, \mathbf{\Sigma}_i^h\}$, where the superscript h is used to denote that they are the hash parameters.

A mapping associating components within the GMM-UBM to the hash GMM is found by selecting the mapping that minimizes the Kullback-Leibler (KL) divergence between the components in the UBM and the components in the hash GMM. The KL divergence is

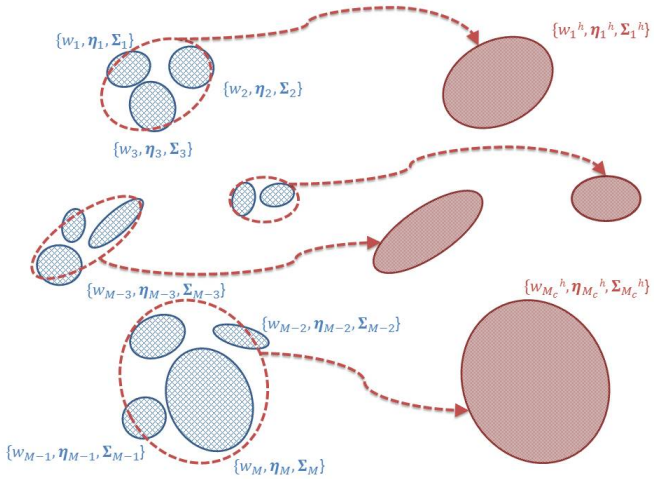


Figure 8. Clustering the components of a GMM-UBM with a large number of components (blue) into a hash GMM with a smaller number of components (red)

a measure of the dissimilarity between two probability distributions. The KL divergence between two probability distributions f and g is

$$d_{\mathrm{KL}}(f, g) = \int f \log \frac{f}{g} \tag{22}$$

One of the issues with using the KL divergence as a distance metric is that it is not symmetric, that is to say that $d_{\mathrm{KL}}(f, g) \neq d_{\mathrm{KL}}(g, f)$. Often a symmetrized version of the KL divergence is used such that $d_{\mathrm{KLS}}(f, g) \equiv d_{\mathrm{KLS}}(g, f)$. This symmetrized KL divergence is given by

$$d_{\mathrm{KLS}}(f, g) = \int f \log \frac{f}{g} + \int g \log \frac{g}{f} \tag{23}$$

For multivariate normal distributions, this symmetrized KL divergence can be simplified to

$$d(f, g) = \frac{1}{2}\mathrm{trace}\left[\left(\boldsymbol{\Sigma}_f^{-1} + \boldsymbol{\Sigma}_g^{-1}\right)\left(\boldsymbol{\mu}_f + \boldsymbol{\mu}_g\right)\left(\boldsymbol{\mu}_f + \boldsymbol{\mu}_g\right)^T + \boldsymbol{\Sigma}_f \boldsymbol{\Sigma}_g^{-1} + \boldsymbol{\Sigma}_f^{-1}\boldsymbol{\Sigma}_g - 2I\right] \tag{24}$$

It should be stressed that this is not the divergence between the GMM-UBM and hash GMM–for which there is no simple closed form solution–rather it is the divergence between individual component densities.

The adjusted MAP-adaptation using clustering is described in the following steps.

1. Calculate feature vectors as described in Section 2.2.
2. Calculate probabilistic alignment of feature vectors into UBM
   (a) Calculate probabilistic alignment of feature vectors into hash GMM using (12) with hash model parameters $\{w_i^h, \eta_i^h, \Sigma_i^h\}$ for $1 \leq i \leq M_c$.
   (b) Choose $N_{\max}$ highest scoring components of the hash GMM for current feature vector
   (c) Calculate probabilistic alignment of feature vectors into GMM-UBM using only clusters of components of the GMM-UBM associated with the $N_{\max}$ highest scoring components of the hash GMM. As an example refer to Figure 8. Suppose that $N_{\max} = 1$ and for the current feature vector the highest scoring component within the hash GMM was that with model parameters $\{w_1^h, \eta_1^h, \Sigma_1^h\}$, we would now evaluate (12) using only the components with parameters $\{w_1, \eta_1, \Sigma_1\}$, $\{w_2, \eta_2, \Sigma_2\}$, and $\{w_3, \eta_3, \Sigma_3\}$ that were mapped to $\{w_1^h, \eta_1^h, \Sigma_1^h\}$ as shown in the figure.
3. Update sufficient statistics (14) calculation using current feature vector and only those clustered components of the GMM-UBM that were found in the previous step. Following the previous example, we would update n_i and $E_i(\mathbf{x})$ where $i \in \{1, 2, 3\}$.
4. MAP adapt as usual (15)

It is important to note that the majority of the SV system using clustering has not diverged from the system described in Chapter 2. The feature vectors have not changed, the formation of supervectors has not changed, and the SVM classification has not changed.

Of interest is the choice of whether or not to implement clustering within the training stage. Ultimately, the system makes decisions based on supervectors. These supervectors have the same form (dimensions and position of associated mean vectors) whether or not clustering is done within training. It is the reduction of the computational complexity of the test stage that is our ultimate goal. Though we are using clustering in the test stage, we are not necessarily constrained to do so in the training stage. One benefit of not clustering within the training stage is that we could make use of previously trained SVM models of our target speakers. One potential disadvantage to not clustering the training stage is that the processes for creating the supervectors would not be identical–which because linear kernel measures similarity–might reduce classification performance accuracy. We have implemented both cases and present results for both.

3.3 Methods for Forming Hash GMMs

This section describes several methods for determining hash GMMs. The first described in Section 3.3 is formed using the same data that was used to train the GMM-UBM while those described in Sections 3.3 and 3.3 rely on information about the GMM-UBM itself. One advantage of these latter methods is that the hash GMM can be formed after the GMM-UBM has been trained and without the original speech feature vectors.

Training Reduced Sized GMM-UBM Using EM and Original Feature Vectors

Ideally, we would like the hash GMM to capture similar features as those modeled by the GMM-UBM. As such, we propose training a hash GMM using the same MFCC vectors used to train the UBM except with a reduced number of components. For instance, if our UBM consisted of 1024 component densities we would train a hash GMM with perhaps only 32 mixture components.

Effectively, this approach entails running the EM algorithm that is used to form the UBM twice. This process was described in Section 2.3. Ultimately, we will have trained two different UBMs, one with a large number of component densities and another with a signicantly reduced number of component densities. Within this approach we refer to this reduced size GMM-UBM as our hash GMM.

The shortlist between the components of the larger GMM-UBM and the components of the hash GMM is determined minimizing the symmetrized KL divergence as described in Section 3.2.

k-Means Clustering of Components using KL Divergence

One disadvantage of calculating a hash GMM using the EM algorithm in the same manner that the UBM is calculated is that it requires access to the original training data and requires us to go through the time consuming EM algorithm again. Another option is to use k-means to cluster the component densities of the UBM. In particular, we have implemented the k-means algorithm using the symmetric KL as in (24) as a distance measure.

While k-means typically is used to calculate a centroid vector–such as a mean vector–we have extended it to calculate several parameters for each centroid. Each centroid is a multivariate Gaussian distribution so it is necessary to calculate a mean vector, covariance matrix, and a weight parameter. We have chosen to calculate our centroids using expectation centroids as described in [15].

After assigning the various mixture components to the clusters based on minimizing the divergence, the mean and covariance of each centroid are calculated for each cluster using its N component members as

$$\begin{aligned} \boldsymbol{\mu}_c &= \frac{1}{N}\sum_{n=1}^{N} E\left[\mathbf{x}_n\right] \\ &= \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\mu}_n \end{aligned} \tag{25}$$

and

$$\begin{aligned}\boldsymbol{\Sigma}_c &= \frac{1}{N}\sum_{n=1}^{N} E\left[\left(\mathbf{x}_n - \boldsymbol{\mu}_c\right)\left(\mathbf{x}_n - \boldsymbol{\mu}_c\right)^T\right] \\ &= \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\Sigma}_n + \boldsymbol{\mu}_n\boldsymbol{\mu}_n^T - \boldsymbol{\mu}_c\boldsymbol{\mu}_c^T \end{aligned} \tag{26}$$

.

We update the component weights by simply summing the individual weights of the components within the cluster

$$w_c = \sum_{n=1}^{N} w_n \tag{27}$$

.

The k-means algorithm with KL-divergence is summarized in the following listing steps.

1. Initialize k-means. In this case we are initializing the centroids by randomly selecting M_c of the component densities of the GMM-UBM. As such each of the centroids will have the model parameters $\{w_c, \mu_c, \boldsymbol{\Sigma}_c\}$ which are the weights, mean vectors, and diagonal covariance matrices of the randomly selected component densities.

2. Iterate the two step process of *assignment* and *update* until either the assignments longer change, some convergence criteria on the total distance is met, or some maximum number of iterations has occurred.

   (a) **Assignment step**: Each component density within the GMM-UBM is assigned to the "closest" cluster. Each is assigned to the cluster for which the KL-divergence (24) between the centroid and the component density is minimized.

   (b) **Update step**: New centroids are determined for each cluster by updating the centroid weights, means, and covariances according to (25)-(27) using *only* the component members that were assigned to that cluster in the previous step.

3. After the stopping criterion has been reached (through convergence or maximum iterations), the k-means centroids are used as the component densities of the hash GMM.

KL GMM Reduction

Recently, Runnalls [24] proposed a KL based approach to GMM reduction–that is to say reduction in the number of mixture component densities. His approach was to successively merge pairs of mixture components, replacing the pairs with a single Gaussian component that matched the merged pair up to a second order. Runnalls' criterion for selecting pairs to

merge was based on minimizing the KL divergence between the GMM before the merge and the GMM after the merge. Although a closed form solution does not exist for calculating the KL divergence between two GMMs, the author does present an upper bound on the divergence between the pre-merger and post-merger GMMs and it is this upper bound that he attempts to minimize. He shows that the KL divergence of the mixture following the merge from the mixture before the merge is bounded by

$$\begin{aligned} B((w_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), (w_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)) = \frac{1}{2} &[(w_i + w_j) \log\det(\boldsymbol{\Sigma}_{ij}) \\ &- w_i \log\det(\boldsymbol{\Sigma}_i) - w_j \log\det(\boldsymbol{\Sigma}_j)] \end{aligned} \tag{28}$$

where $B[(w_i, \boldsymbol{\mu_i}, \boldsymbol{\Sigma}_i), (w_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)]$ is computed for every pair of component members with $i \neq j$ within the premerged GMM.

The two components that minimize $B((w_i, \boldsymbol{\mu_i}, \boldsymbol{\Sigma}_i), (w_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j))$ are selected for merger and are replaced by the moment-preserving merge

$$\begin{aligned} w_{ij} &= w_i + w_j \\ \boldsymbol{\mu}_{ij} &= w_{i|ij}\boldsymbol{\mu}_i + w_{j|ij}\boldsymbol{\mu}_j \\ \boldsymbol{\Sigma}_{ij} &= w_{i|ij}\boldsymbol{\Sigma}_i + w_{j|ij}\boldsymbol{\Sigma}_j + w_{i|ij}w_{j|ij}\left(\boldsymbol{\mu}_i + \boldsymbol{\mu}_j\right)\left(\boldsymbol{\mu}_i + \boldsymbol{\mu}_j\right)^T \end{aligned} \tag{29}$$

where the component weights have been normalized such that $w_{i|ij} = w_i/(w_i + w_j)$ and $w_{j|ij} = w_j/(w_i + w_j)$.

Our process of iteratively selecting components to merge and calculating the moment preserving merge is continued until the merged GMM is reduced to contain the desired number of reduced mixture components. At each stage of the merging process, a record is kept of which components are merged to later be used as a shortlist between the reduced (hash) GMM and the components of the GMM-UBM.

The summarized process of GMM reduction according to Runnalls is as follows.

1. Iterate the following steps $M - N_c$ times until only N_c merged components remain.

2. **Calculate bound**: Calculate the upper bound (28) on the divergence for all pairs of components i and j within the merged GMM from the last iteration.

3. **Merge Pairs**: Choose the two pairs of components with indices i and j that minimized the bound and merge the pair using the moment preserving merger of their weights, mean vectors, and covariance matrices (29).

Normalized Mean Vector Based Clustering

Here we describe another clustering method that normalizes the component mean vectors before using the traditional k-means algorithm with Euclidean distance to cluster them. In this method, we are normalizing our mean vectors for unit variance and weighting according to the component weights

$$\bar{\boldsymbol{\mu}}_{\boldsymbol{n}} = w_n \boldsymbol{\Sigma}_{\boldsymbol{c}}^{-1} \boldsymbol{\mu}_{\boldsymbol{n}} \tag{30}$$

. It should be noted that because we are using diagonal matrices, (30) amounts to element by element dividing the mean by the covariance in each dimension and multiplying by the component weight.

These normalized mean vectors are then fed into the typical k-means algorithm. After this clustering, we calculate a hash GMM using the expectation centroid as in (25)-(27).

Clustering Based on *a posteriori* Probabilities of Training Feature Vectors

The previous methods assumed only the availability of the GMM-UBM and not the feature vectors used to train the GMM-UBM. Here we describe a method that forms a hash GMM by analyzing the *a posteriori* probabilities that each feature vector produces when analyzed with the GMM-UBM. Using the same feature vectors that were used to train the UBM, for each MFCC feature vector we calculate the likelihood that the feature vector came from each of the component densities using (12). As such, for a UBM consisting of M component densities, each training MFCC feature vector will be used to generate a vector of component probabilities of length M, $P_{MFCC} = [Pr(1|\mathbf{x}_t), \ldots, Pr(M|\mathbf{x}_t)]$. We calculate one of these alignment vectors for each training feature vector. These vectors of probabilistic alignment are then fed into the traditional k-means algorithm for clustering.

It is straight forward to calculate the cluster centroids using the indices returned from the k-means algorithm and the MFCC feature vectors. The centroids are calculated as

$$\boldsymbol{\mu}_{\boldsymbol{c}} = \sum_{n=1}^{N} E\left[\mathbf{x}_n\right] \tag{31}$$

and

$$\boldsymbol{\Sigma}_c = \frac{1}{N} \sum_{n=1}^{N} E\left[\left(\mathbf{x}_n - \boldsymbol{\mu}_c\right)\left(\mathbf{x}_n - \boldsymbol{\mu}_c\right)^T\right] \tag{32}$$

. The weight for each cluster is calculated as the point probability of each cluster from the k-means output. More precisely the weight is calculated by taking the number of MFCC feature vectors assigned to a cluster and dividing by the total number of MFCC vectors.

The output centroids of the k-means algorithm are vectors representing the likelihood outputs for different classes of MFCC inputs. Each centroid is going to be length M. In

general, the centroids are sparse meaning that for a given class of MFCC input vector, only a subset of the component densities will have a significant response. As such, for each cluster, we only assign components that have a significant likelihood response. For each clustered centroid, we have chosen to keep the most likely components whose sum likelihood is greater than 65%–which have have chosen arbitrarily to control trade of between computational performance and classification accuracy.

This is significantly different than the previous algorithms described. Previously, each and every component would be uniquely identified with one cluster. With this approach, it is possible that a component density will be used in multiple clusters. It is also possible that some components may not be used at all which would imply that we are not capturing all the information contained in the original GMM.

4 Results of GMM Component Clustering

4.1 Introduction

This chapter describes the simulation parameters and the results obtained in those simulations.

4.2 Data Sets and Simulation Parameters

The UBM was trained by using data from both the Switchboard II Phase 1 corpus [10] and the Switchboard Cellular Part 2 corpus [11]. Two GMMs were trained for each corpus–one for male speakers and one for female speakers. A single speech utterance was chosen for each speaker so as not to bias the statistics for speakers with more utterances. After training the four GMMs using the EM algorithm, the GMMs from the subpopulations were merged and weights were adjusted accordingly. We have performed our experiments using the NIST 2002 speaker recognition evaluation (SRE) corpus [14]–in particular the single speaker cellular data. Each training file within NIST2002 was segmented into 10 individual training sequences and subsequently into 10 supervectors for each speaker. For front end processing, we calculated a 19-dimensional MFCC vector every 10 ms using a 25 ms Hamming window. The frequency content was limited to the range 300-3140 Hz. The MFCC vectors were processed with RASTA filtering. Δ-MFCCs were then calculated using a 5 sample window length. These 19 delta-cepstral coefficients were concatenated to the cepstra vector to generate a 38-dimensional vector. Finally these cepstral/delta-cepstral vectors were processed with feature warping [17] to generate the sequence of feature vectors.

The system was evaluated using GMM-UBMs with 256 member components and 1024 member components.

4.3 Performance Metric

In order to evaluate the efficiency performance of our algorithms it is necessary to choose metrics to do so. We have chosen to use the average number of *a posteriori* probability calculations (12). In the case of no clustering, the average number of *a posteriori* probability calculations per MFCC vector is simply the number of mixture components within the GMM-UBM. When clustering is used, the number of *a posteriori* probability calculations includes both the number of clusters and the number of components within each cluster selected.

This metric is useful because it allows us to compare our clustering to the case where we would simply use a GMM-UBM system trained with a reduced number of mixture components. For example if the average number of *a posteriori* probability calculations for a particular clustering scheme was 32, we could compare the EER achieved with this system to a GMM-UBM system with 32 mixture components.

4.4 Experiments and Results

In our experiments, we simulated systems with of 4, 8, 16, and 32 clusters. Further, we simulated scenarios of 1, 2, and 4 maximum likelihood clusters chosen for further processing. Finally, for all cases, we implemented the clustering during the testing stage and also simulated the training stage with and without clustering. Thus, when clustering is done in both testing and training, then the processing steps match whereas when clustering is not done in training the processing is mismatched.

The following subsections present the results for the systems based on GMM-UBMs with 256 and 1024 component densities. It is vital to note that in our simulations, a baseline simulation of the typical system without clustering resulted lower EER results with the 256 component UBM than the 1024 component system. This is counter to what is often presented in literature where systems are usually implemented with 1024 or 2048 mixture components for better performance. Because the baseline system with 1024 components has degraded performance with respect to the 256 component system, our results for clustering the 1024 component system will already be at a disadvantage to systems trained without clustering but with fewer components.

Results for 256 Component GMM-UBM

Fig. 9 and 10 show the results of the different clustering algorithms when the initial UBM consisted of 256 component densities. In the plots, the solid line represents the case of simply implementing a GMM-UBM system with fewer component densities–for which case the average number of likelihood calculations is simply equal to the number of component densities. Clearly, for the majority of cases, the clustering methods did not perform as well as simply implementing the reduced size GMM-UBM since the EERs are greater. The exception was implementing the clustering using the GMM reduction method as described by Runnalls [24].

The results for clustering the 256 component UBM are presented in Tables 1-5. Of most interest are the results achieved using the reduction method proposed by Runnalls. These results are tabulated in Table 5. The table lists the number of clusters implemented, the number top clusters to use, N_{max}, reduction factor of the average *a posteriori* probability calculations calculated per feature vector, the degradation (increase) in EER as a relative percent when clustering was done only in testing, and the degradation of EER when clustering was done in both training and testing. The results demonstrate that substantial reduction in the number of likelihood calculations has been achieved with minimal degradation in EER.

In general performance is degraded when clustering is used only for testing (as opposed to training and testing). In particular, the clustering based on *a posteriori* probabilities of training feature vectors and k-means based clustering methods incurred substantial degradations when deviating from matched processing. Of particular interest is that the Runnalls GMM reduction method appears to be robust to the case of unmatched processing. In fact,

Clusters	**Nmax**	**Computational Reduction Factor Cluster Test**	**EER Rise % Cluster Test**	**Computational Reduction Factor Cluster Test and Train**	**EER Rise % Cluster Test and Train**
2	1	1.9758	10.3641	2.0665	7.0028
4	1	1.9617	9.2437	2.7817	13.4454
8	1	3.1829	15.1261	2.6082	15.4062
16	1	3.1785	21.5686	3.1697	21.7367
32	1	3.98	30.7563	3.2322	19.6078
4	2	1.9417	7.563	1.8523	4.6499
8	2	1.9745	5.042	1.916	3.3613
16	2	2.3476	5.042	2.323	8.6835
32	2	2.2482	7.563	2.4141	11.4846
8	4	1.7874	6.7227	1.6975	1.9328
16	4	1.7301	1.9608	1.7267	3.3613
32	4	2.0882	4.4818	2.1155	4.3697

Table 1. Results for Clustering based on *a posteriori* Probabilities of Training Vectors with a UBM of 256 Components

Clusters	Nmax	Computational Reduction Factor Cluster Test	EER Rise % Cluster Test	Computational Reduction Factor Cluster Test and Train	EER Rise % Cluster Test and Train
2	1	42.6255	311.3725	1.0094	0
4	1	6.0123	256.3025	7.0151	129.972
8	1	8.1028	193.5574	6.0281	143.4174
16	1	7.6179	81.7927	7.1523	115.1261
32	1	6.3894	82.7731	6.1599	41.4566
4	2	1.387	22.9692	2.1146	3.6415
8	2	4.2699	88.6835	3.7647	23.2493
16	2	4.3996	11.7647	4.8229	27.451
32	2	5.2319	35.5742	5.0645	33.6134
8	4	2.1142	39.4958	1.9086	6.4426
16	4	2.9512	34.1737	3.2988	15.6863
32	4	3.8173	30.2521	3.8101	11.2045

Table 2. Results for K-Means Clustering with a UBM of 256 Components

Clusters	Nmax	Computational Reduction Factor Cluster Test	EER Rise % Cluster Test	Computational Reduction Factor Cluster Test and Train	EER Rise % Cluster Test and Train
2	1	2.0317	61.0644	2.0317	16.1345
4	1	12.1905	283.3333	12.8	112.8571
8	1	12.7956	298.3193	17.0641	82.0728
16	1	13.5646	291.8768	11.1824	91.5966
32	1	7.4007	271.0924	7.4184	85.1541
4	2	7.3143	246.7787	9.1429	81.7927
8	2	9.8459	266.1064	9.8448	86.8067
16	2	8.0477	240.056	8.5105	56.3025
32	2	6.8004	235.8543	6.7869	67.7871
8	4	3.2405	120.6723	3.1605	26.0504
16	4	5.4849	176.1905	5.8123	50.1401
32	4	5.7428	203.0812	6.0007	65.2661

Table 3. Results for Normalized Mean Clustering with a UBM of 256 Components

Clusters	Nmax	Computational Reduction Factor	EER Rise % Cluster Test	EER Rise % Cluster Test and Train
4	1	2.5947	35.5742	22.6891
8	1	5.6497	44.2577	26.8908
16	1	7.1419	57.423	39.7759
32	1	6.2446	52.6611	31.9328
4	2	1.7502	18.4874	10.3361
8	2	3.2335	27.1709	14.2857
16	2	4.8742	22.1289	15.9664
32	2	5.161	30.8123	19.6078
8	4	1.76	9.1597	1.8487
16	4	2.8839	13.7255	5.6022
32	4	3.9471	17.619	10.084

Table 4. Results for Smaller UBM Mapping with a UBM of 256 Components

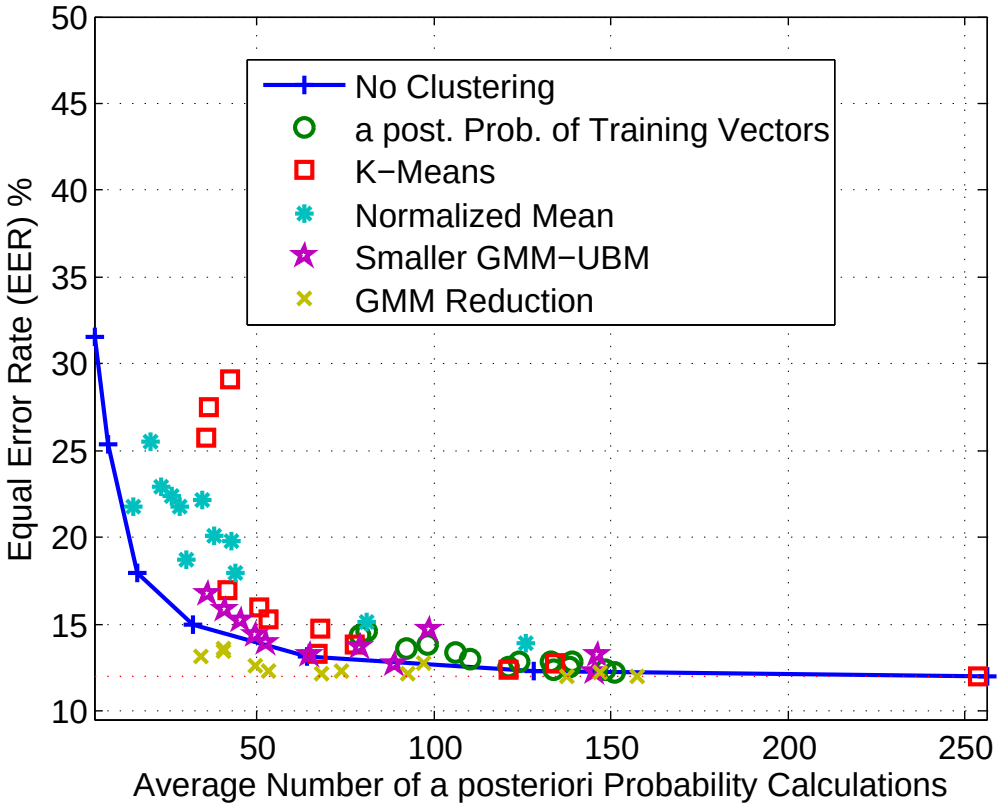


Figure 9. Results for Applying Clustering Method to a 256 Component UBM for both Training and Testing

as shown in Table 5, when we used 16 clusters and processed the 4 clusters with maximum likelihood, we were able to achieve an EER of 11.87% with on average 92.4 *a posteriori* probability calculations. This actually surpassed our baseline EER of 11.97% with a 256 component UBM. So for this particular case, not only were we able to reduce the average number of likelihood calculations by a factor of 2.77 but we also improved our EER by 0.84%.

Results for 1024 Component GMM-UBM

In addition to evaluating our method with a 256 component UBM we also evaluated it with a 1024 component UBM. It should be noted that without clustering, in our testing conditions we actually achieved lower EER with 256 components rather than 1024 in our baseline system. This is shown in Fig. 11 in which the blue solid line represents implementing the standard SV system without any clustering with different number of component densities. It can be seen in the figure that EER is lowest at 256 component densities then EER increases with increasing number of components at 512 and 1024 components.

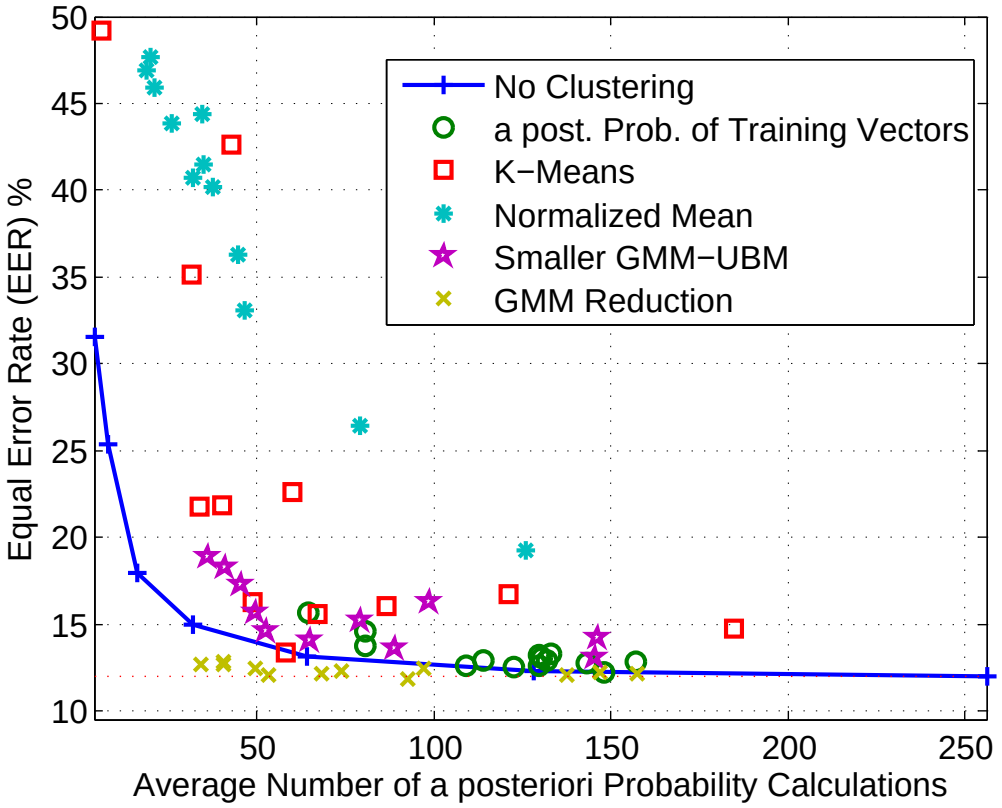


Figure 10. Results for Applying Clustering Method to a 256 Component UBM for Testing Only

Essentially, this means that if we are clustering a 1024 component UBM we are already at a disadvantage when the system is compared to the 256 component baseline. In this case, we are not able to match the performance of a smaller UBM with the clustered method, neither for matched processing nor for unmatched processing.

Though clustering a suboptimal 1024 component UBM does not allow us to match the performance of the 256 component UBM, we can still analyze the degradation with respect to the 1024 component UBM. Again, the GMM reduction method proposed by Runnalls [24] appears to provide the best results as shown in Fig. 11 and Fig. 12. The Runnalls method works well for both matched and unmatched processing. Starting with the 1024 component UBM we are able to achieve a factor of 5 reduction with no loss and a factor of 10 reduction with less than 2.4% loss in relative performance.

Clusters	**Nmax**	**Computational Reduction Factor**	**EER Rise % Cluster Test**	**EER Rise % Cluster Test and Train**
2	1	1.7438	2.2409	1.9608
4	1	2.6391	3.9216	6.7227
8	1	6.3162	7.2829	13.7255
16	1	7.466	6.1625	9.5238
32	1	6.3008	6.1625	12.3249
4	2	1.6267	1.6807	0
8	2	3.4736	2.521	2.521
16	2	4.8215	0.84034	2.9972
32	2	5.1547	3.8095	5.042
8	4	1.8643	0.84034	0.28011
16	4	2.7702	-0.84034	1.1204
32	4	3.7584	1.3725	1.3725

Table 5. Results for Runnalls GMM Reduction with a UBM of 256 Components

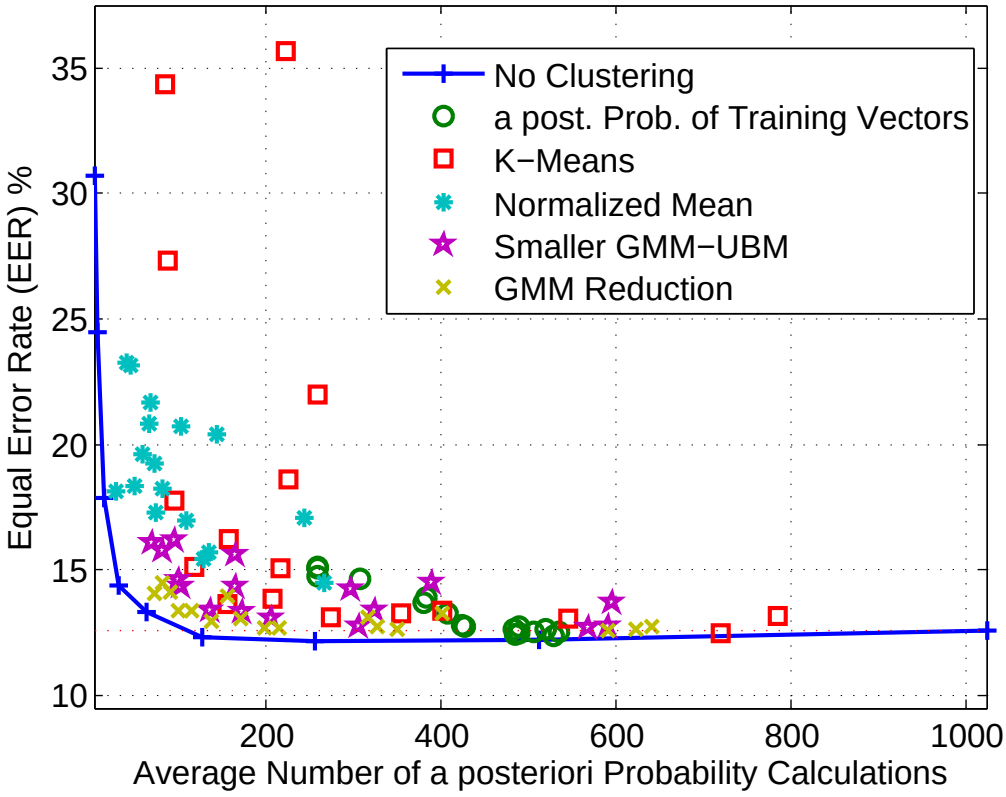


Figure 11. Results for Applying Clustering Method to a 1024 Component UBM for both Training and Testing

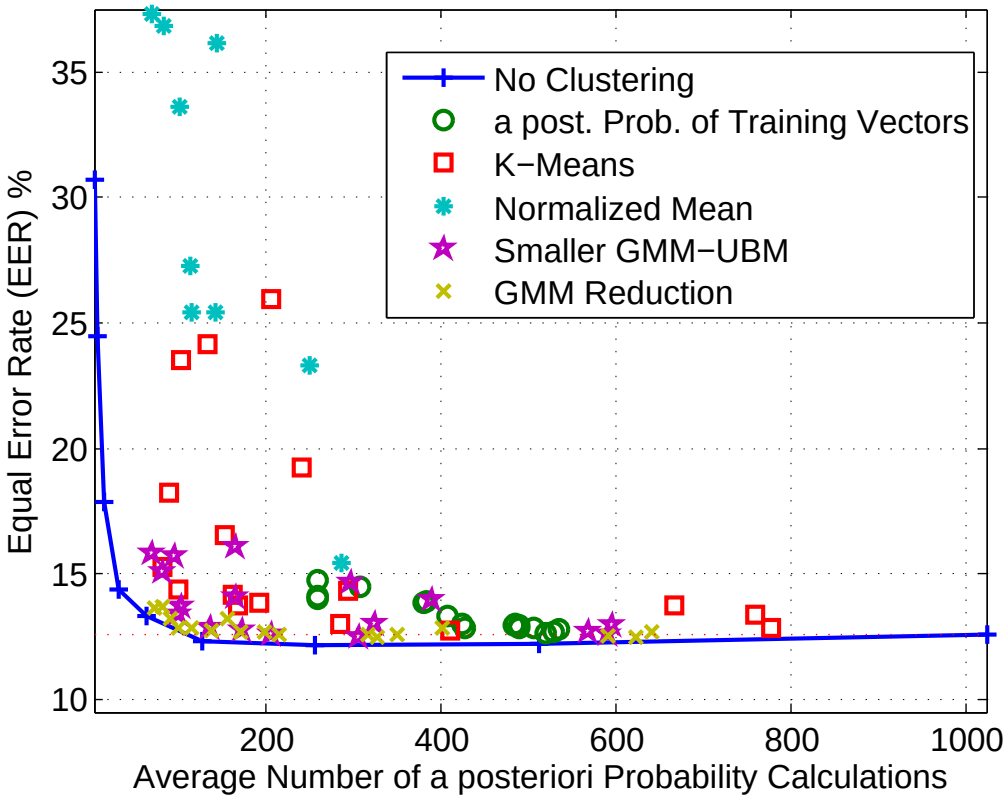


Figure 12. Results for Applying Clustering Method to a 1024 Component UBM for Testing Only

Clusters	Nmax	Computational Reduction Factor Cluster Test	EER Rise % Cluster Test	Computational Reduction Factor Cluster Test and Train	EER Rise % Cluster Test and Train
4	1	2.6898	10.3733	2.6898	8.8
8	1	2.6676	10.6667	2.6676	10.6667
16	1	3.3268	15.2	3.3268	16.5333
32	1	3.9454	17.52	3.9454	19.7333
64	1	3.9368	12.5333	3.9368	20.2667
4	2	2.1201	2.9067	2.1201	0.53333
8	2	2.0906	2.1067	2.0906	-0.8
16	2	2.3938	2.1333	2.3938	1.6
32	2	2.5098	6.1333	2.5098	5.4933
64	2	3.954	11.68	3.954	17.3333
8	4	2.1132	3.4667	2.1132	-1.0667
16	4	2.0201	2.1333	2.0201	-0.26667
32	4	2.0926	2.9333	2.0926	1.3333
64	4	2.4105	3.4667	2.4105	1.8667
16	8	1.911	1.8667	1.911	-0.26667
32	8	1.9354	0.8	1.9354	-1.3333
64	8	1.9698	0.53333	1.9698	0.53333

Table 6. Results for Clustering based on *a posteriori* Probabilities of Training Vectors with a UBM of 1024 Components

Clusters	Nmax	Computational Reduction Factor Cluster Test	EER Rise % Cluster Test	Computational Reduction Factor Cluster Test and Train	EER Rise % Cluster Test and Train
4	1	7.6311	92.2667	3.9522	74.9333
8	1	4.9513	106.4	4.5882	183.7333
16	1	9.9557	87.2	10.7367	41.3333
32	1	12.4062	21.52	11.5402	117.6
64	1	11.4068	45.0667	11.9784	173.3333
4	2	1.3488	6.6667	1.3029	4.72
8	2	4.2554	53.2	4.5345	48.2667
16	2	6.6737	31.4667	4.7177	19.7333
32	2	6.3231	12.8	6.4615	29.0133
64	2	10.28	14.4	8.6699	20.2667
8	4	1.5335	9.2533	1.8739	4
16	4	3.4882	13.8667	2.8853	5.8133
32	4	5.2993	10.4	4.9198	10.2667
64	4	6.1011	9.6	6.569	8.5333
16	8	1.3167	2.4	1.4207	-0.53333
32	8	2.4953	1.6	2.5455	6.6133
64	8	3.6003	3.4667	3.7348	4.5333

Table 7. Results for K-Means Clustering with a UBM of 1024 Components

Clusters	Nmax	Computational Reduction Factor Cluster Test	EER Rise % Cluster Test	Computational Reduction Factor Cluster Test and Train	EER Rise % Cluster Test and Train
4	1	15.2836	223.7333	15.2836	65.6
8	1	17.6552	223.2	22.2609	84
16	1	41.9241	289.3333	35.3093	44.2667
32	1	24.4105	269.0667	25.0387	85.0667
64	1	15.3869	274.9333	14.9298	72.2667
4	2	7.1111	187.4667	7.1111	62.4
8	2	7.1608	102.1067	9.8462	64.8
16	2	17.0667	231.1467	17.3556	56
32	2	20.1216	232.8	20.121	46.1333
64	2	13.8695	237.76	14.127	53.0667
8	4	4.0797	85.6	4.1967	36
16	4	10.1386	167.6	7.9642	22.8
32	4	14.7336	196.8	13.8362	37.6
64	4	12.1724	193.2533	12.5053	45.0667
16	8	3.5643	22.9067	3.8352	15.4667
32	8	8.8614	102.1333	7.5829	25.0667
64	8	9.0329	117.0667	9.3759	34.9333

Table 8. Results for Normalized Mean Clustering with a UBM of 1024 Components

Clusters	Nmax	Computational Reduction Factor	EER Rise % Cluster Test	EER Rise % Cluster Test and Train
4	1	2.6269	10.9333	15.4667
8	1	6.1905	28	24
16	1	10.6756	25.0667	28.8
32	1	14.7624	25.6	27.9733
64	1	12.4148	20	25.28
4	2	1.7165	2.9333	9.0667
8	2	3.4452	16.6933	13.0667
16	2	6.1693	11.7333	14.1333
32	2	9.9497	8.8	14.08
64	2	10.209	6.6667	16
8	4	1.8027	0.8	0.8
16	4	3.1623	3.4667	6.6667
32	4	5.939	1.6	6.1333
64	4	7.5256	2.2667	6.4
16	8	1.7298	0.26667	1.3333
32	8	3.3493	-0.53333	1.6
64	8	4.9724	0.26667	3.7333

Table 9. Results for Smaller UBM Mapping with a UBM of 1024 Components

Clusters	Nmax	Computational Reduction Factor	EER Rise % Cluster Test	EER Rise % Cluster Test and Train
4	1	2.5442	2.1067	5.52
8	1	6.5393	5.3333	10.9333
16	1	11.1469	5.3333	12.2667
32	1	14.1382	8.5333	12
64	1	12.531	8.8	15.36
4	2	1.5961	0.8	1.3333
8	2	3.2246	0.26667	4.2667
16	2	5.9666	0.96	3.7333
32	2	8.8987	2.4	6.6667
64	2	10.2276	2.4	6.4
8	4	1.7317	-0.26667	0.26667
16	4	3.1251	-0.53333	1.6
32	4	5.169	0.8	1.0667
64	4	7.399	1.3333	2.9333
16	8	1.6445	-0.53333	0.53333
32	8	2.9248	0	0.53333
64	8	4.7554	0.18667	1.0667

Table 10. Results for Runnalls GMM Reduction with a UBM of 1024 Components

5 Conclusions and Future Work

5.1 Conclusions

In this dissertation proposal, we presented a method for reducing the computational load of the GMM-UBM SV system by clustering the component densities of the GMM-UBM. We compared the results to systems based on GMM-UBMs with a reduced number of component densities. We have shown that in some cases we were able to achieve lower EER with less computational burden than systems with smaller GMMs by clustering a larger GMM-UBM. Further we have demonstrated that we can achieve promising results even in the case when clustering is not performed in the training stage but only in the testing stage. This opens the possibility that we might be able to throttle the amount of processing that we are performing in real time by adjusting the number of clusters, N_{max}, that we adapt.

5.2 Future Work

Additional research to carryout along these lines include the addition of the mixture weights in the divergence calculation used in the k-means clustering algorithm. In such a system, components with low weight would be more likely to be merged into clusters [16]. Second, in our analysis, we fixed the number of clusters for adaptation, improved gains might be achieved by intelligently adjusting the number of clusters to adapt based on the likelihood scores of the clusters. For example, for some feature vectors only one or two hash GMM components might have a significant probability whereas with others perhaps more components would be significant. On a feature vector by feature vector basis a decision could be made as to how many clusters to use.

A third possibility for improvements is to use a tree based hash GMM. Currently the hash GMM in this research has used a single layer GMM that maps directly to the GMM-UBM. The components are not equally distributed among the clusters of the hash GMM so it might be worthwhile to perform clustering with the hash clusters which contain a significant number of component members.

Finally, a fourth potential area of research is in the feature warping algorithm. Feature warping helps us achieve improved EER but it comes at a significant computational burden. It would be worthwhile to determine whether there might be other ways of achieving similar effects but with reduced computation cost.

References

[1] V.R. Apsingekar and P.L. De Leon. Speaker model clustering for efficient speaker identification in large population applications. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(4):848 –853, may 2009.

[2] Roland Auckenthaler and John S. Mason. Gaussian selection applied to text-independent speaker verification. In *In Proc. Speaker Odyssey 2001*, pages 83–88, 2001.

[3] J. Benesty, M.M. Sondhi, and Y. Huang. *Springer handbook of speech processing*. Springer Handbook Of Series. Springer, 2008.

[4] C.M. Bishop. *Pattern recognition and machine learning*. Springer, 2009.

[5] W.M. Campbell, D.E. Sturim, and D.A. Reynolds. Support vector machines using gmm supervectors for speaker verification. *Signal Processing Letters, IEEE*, 13(5):308 – 311, may 2006.

[6] S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy. Svm and kernel methods matlab toolbox. Perception Systmes et Information, INSA de Rouen, Rouen, France, 2005.

[7] D.G. Childers, D.P. Skinner, and R.C. Kemerait. The cepstrum: A guide to processing. *Proceedings of the IEEE*, 65(10):1428 – 1443, oct. 1977.

[8] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357 – 366, aug 1980.

[9] P.L. De Leon, M Pucher, J. Yamagishi, I. Hernaez, and I Saratxaga. Evaluation of speaker verification security and detection of synthetic speech. *in review IEEE Trans. Audio, Speech, and Language Proc.*, oct 2011.

[10] David Graff, Alexandra Canavan, and Zipperlen. *Switchboard-2 Phase I*. Linguistic Data Consortium, Phildelphia, 1998.

[11] David Graff, Kevin Walker, and David Miller. *Switboard Cellular Part 2 Audio*. Linguistic Data Consortium, Phildelphia, 2004.

[12] H. Hermansky and N. Morgan. Rasta processing of speech. *Speech and Audio Processing, IEEE Transactions on*, 2(4):578 –589, oct 1994.

[13] T. Kinnunen, E. Karpov, and P. Franti. Real-time speaker identification and verification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(1):277 – 288, jan. 2006.

[14] Alvin Martin and Mark Przybocki. *2002 NIST Speaker Recognition Evaluation*. Linguistic Data Consortium, Phildelphia, 2004.

[15] T.A. Myrvoll and F.K. Soong. Optimal clustering of multivariate normal distributions using divergence and its application to hmm adaptation. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, volume 1, pages I–552 – I–555 vol.1, april 2003.

[16] A. Ogawa and S. Takahashi. Weighted distance measures for efficient reduction of gaussian mixture components in hmm-based acoustic model. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 4173 –4176, 31 2008-april 4 2008.

[17] Jason Pelecanos and Sridha Sridharan. Feature warping for robust speaker verification. In *In Proc. Speaker Odyssey 2001*, 2001.

[18] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.

[19] L.R. Rabiner and R.W. Schafer. *Theory and Applications of Digital Speech Processing*. Pearson, 2010.

[20] Douglas A. Reynolds. An overview of automatic speaker recognition technology. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 4, pages IV–4072 –IV–4075, may 2002.

[21] Douglas A. Reynolds. Universal background models. In *Encyclopedia of Biometrics*, number v. 1. Springer, 2008.

[22] Douglas A. Reynolds. Gaussian mixture models. In *Encyclopedia of Biometrics*, number v. 1. Springer, 2009.

[23] Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn. Speaker verification using adapted gaussian mixture models. In *Digital Signal Processing*, page 2000, 2000.

[24] A.R. Runnalls. Kullback-leibler approach to gaussian mixture reduction. *Aerospace and Electronic Systems, IEEE Transactions on*, 43(3):989 –999, july 2007.

[25] G. Sarkar and G. Saha. Analysis of distance measures for pre-quantization before feature extraction in automatic speaker recognition. In *India Conference (INDICON), 2009 Annual IEEE*, pages 1 –4, dec. 2009.

[26] Gourav Sarkar and Goutam Saha. Real time implementation of speaker identification system with frame picking algorithm. *Procedia Computer Science*, 2(0):173 – 180, 2010. Proceedings of the International Conference and Exhibition on Biometrics Technology.

[27] Bing Xiang and T. Berger. Efficient text-independent speaker verification with structural gaussian mixture models and neural network. *Speech and Audio Processing, IEEE Transactions on*, 11(5):447 – 456, sept. 2003.

DISTRIBUTION:

1 Prof. Phillip L. De Leon
New Mexico State University
Klipsch School of Elect & Comp Eng
Box 30001, Department 3-O
Las Cruces, New Mexico 88003-8001

1	MS 0429	Tedd Rohwer, 02120
1	MS 0509	Dahlon Chu, 05330
1	MS 0509	Robert Spulak, 05331
1	MS 0519	Greg Haschke, 05332
5	MS 0348	Richard McClanahan, 05332
1	MS 0519	Richard Ormesher, 05332
1	MS 0519	Silpan Patel, 05332
1	MS 0519	Kenneth Plummer, 05332
1	MS 0529	David Wiegandt, 05333
1	MS 0529	Joel Darnold, 05335
1	MS 0348	Doug Mangum, 05350
1	MS 0348	John Moser, 05352
1	MS 0529	David Steele, 05932
1	MS 0899	RIM-Reports Management, 9532 (electronic copy)

Sandia National Laboratories