Mixture Component Clustering for Efficient Speaker Verification

Richard D. McClanahan[1]*, Phillip L. De Leon*[2]

[1]Sandia National Laboratories, Albuquerque, N.M., U.S.A.
[2]New Mexico State University, Klipsch School of Elect. and Comp. Eng., Las Cruces, N.M., U.S.A.
rmcclan@sandia.gov, pdeleon@nmsu.edu

Abstract

In speaker verification (SV) systems based on a support vector machine (SVM) using Gaussian mixture model (GMM) supervectors, a large portion of the test-stage computational load is the calculation of the *a posteriori* probabilities of the feature vectors for the given universal background model (UBM). Furthermore, the calculation of the sufficient statistics for the mean also contributes substantially to computational load. In this paper, we propose several methods to cluster the GMM-UBM mixture components in order to reduce the computational load and speed up the verification. In the adaptation stage, we compare the feature vectors to the clusters and calculate the *a posteriori* probabilities and update the statistics exclusively for mixture components belonging to appropriate clusters. Our results, demomstrate that (on average) we can, reduce the number of *a posteriori* probability calculations by a factor up to $2.8\times$ without loss in accuracy.

Index Terms: speaker recognition, clustering methods

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, 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. This task must therefore perform a $1{:}N$ classification–one for each of the known speakers [1].

Recent research has examined ways to reduce the required computation of SR systems without sacrificing accuracy—which will always be an 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 a 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.

A number of different methods have been proposed to reduce the computation in GMM-UBM SV systems where the computational bottleneck is in the number of *a posteriori* probability calculations required in the log likelihood ratio. For example, a method was proposed to reduce the number of features being evaluated by eliminating those with substantial redudancy [2]. Depending on the corpus and distance measured, the authors were able to achieve little to no performance degradation with a frame rate reduced by a factor of four. Other researchers have used hash tables and were able to reduce the computational burden by creating a shortlist between a smaller hash GMM and the full component size GMM [3]. The authors achieved a processing reduction factor of about $6\times$ with "no noticeable performance degradation." The approach in [4] 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 paper is on how to reduce the computational load in the SV test-stage of a state-of-the-art, support vector machine (SVM) using Gaussian mixture model (GMM) supervectors system [5]. We propose several different methods of creating a hash GMM. The first method is similar to the GS1 hash of [3] 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 [3, 4] 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 rather than the GMM-UBM system.

The remainder of the paper is organized as follows. In Section 2, we review the SV system based on SVM using GMM supervectors. Next, in Section 3, we present several methods of clustering the GMM-UBM component densities to create a hash GMM. In Section 4, we present the results that we have obtained using these various methods. Finally, we conclude our paper in Section 5.

2. Speaker Verification Based on SVM Using GMM Supervectors

In this section, we briefly review the SVM using GMM supervectors SV system described in [5].

2.1. SV System Training

Training the SVM SV system is achieved in four steps. The first step consists of constructing a GMM-UBM using feature vectors, $\mathbf{X}$, from a large collection of background speakers. The feature vectors are usually composed of mel-frequency cepstral coefficients (MFCCs) and delta MFCCs. 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 component densities in the GMM-UBM.

In the second step, feature vectors are extracted from a target speaker's utterance and used to MAP-adapt the mean vec-

tors of the GMM-UBM. It is assumed that we have several utterances available for each target speaker. The MAP-adapted model is denoted $\lambda_{s,u} = \{w_i, \boldsymbol{\mu}_{s,u,i}, \boldsymbol{\Sigma}_i\}$ where $\boldsymbol{\mu}_{s,u,i}$ is the MAP-adapted mean vector for the i-th component density from utterance u of speaker s.

In the third step, the mean vectors $\boldsymbol{\mu}_{s,u,i}$ are then diagonally-scaled according to

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

and stacked to form a GMM supervector for a speaker's given utterance

$$\mathbf{m}_{s,u} = \begin{bmatrix} \mathbf{m}_{s,u,1} \\ \vdots \\ \mathbf{m}_{s,u,M} \end{bmatrix}. \quad (2)$$

The fourth training step involves training a SVM to discriminate the target speaker. The SVM is trained using a linear kernel [6], with weight and bias parameters a_n and b. The supervectors for the target speaker are labeled $+1$ whereas the supervectors of all other background speakers are labeled -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.

2.2. SV System Testing

In the SV test stage, we are given a speech utterance and an identity claim C. We must decide to accept or reject the claim. To do so, we extract T feature vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$ from the utterance and form a test supervector, $\mathbf{m}_{\text{test}}$ following the same procedure as in steps 2 (MAP adaptation) and 3 (supervector) of the training stage. The supervector is evaluated against the model SVM by computing

$$y(\mathbf{X}) = \sum_{n \in \mathcal{S}} a_{C,n} l_{C,n} \mathbf{m}_{\text{test}}^T \mathbf{m}_{C,n} + b_C \quad (3)$$

where $l_{C,n}$ denotes the labels associated with the support vectors and the claim is accepted if $y(\mathbf{X}) \geq 0$ otherwise it is rejected.

2.3. MAP-Adaptation

In the test-stage, MAP-adaptation begins with calculating the alignment of the training vectors into the UBM by computing the likelihood function

$$\Pr(i|\mathbf{x}_t) = \frac{w_i p_i(\mathbf{x}_t)}{\sum_{j=1}^{M} w_j p_j(\mathbf{x}_t)} \quad (4)$$

where

$$p_i(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_i|^{1/2}} e^{-(1/2)(\mathbf{x}-\boldsymbol{\mu}_\mathbf{i})^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_\mathbf{i})}. \quad (5)$$

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

$$n_i = \sum_{t=1}^{T} \Pr(i|\mathbf{x}_t) \quad (6)$$

and

$$E_i(\mathbf{x}) = \frac{1}{n_i} \sum_{t=1}^{T} \Pr(i|\mathbf{x}_t)\mathbf{x}_t. \quad (7)$$

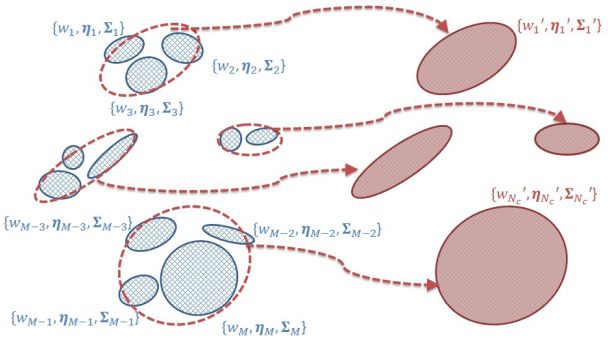

Figure 1: Clustering the components of a GMM-UBM with a large number of components (blue) into a hash GMM with a smaller number of components (red)

These sufficient statistics are then used to adapt the UBM to reflect the characteristics of the hypothesized speaker. Finally, the test supervector is formed, (3) is computed, and an accept/reject decision is made.

3. GMM Component Density Clustering

Within the test stage, a significant portion of the computational load is required for the calculation of likelihoods in (4) and sufficient statistics in (6) and (7). Clearly, one way to reduce computation time is to not perform the calculations for the entire set of mixture components since many of these evaluate to near zero. Effectively, we only calculate the terms in (4), (6), and (7) for a subset of mixture components, $\mathcal{C}$, that are properly chosen.

We next describe several methods for clustering the component densities of the GMM-UBM to create smaller hash GMMs as shown in Fig. 1. In the illustration, the components of the original GMM-UBM $\{w_i, \boldsymbol{\eta}_i, \boldsymbol{\Sigma}_i\}$ are combined to form new component densities $\{w_i', \boldsymbol{\eta}_i', \boldsymbol{\Sigma}_i'\}$ within the hash GMM. A shortlist is created that maps components of the hash GMM to clusters of components in the GMM-UBM. A small number of clusters in the GMM-UBM is chosen based on the scoring of the feature vectors into the hash GMM. Only the components in these clusters are used to calculate (4), (6), and (7).

3.1. Mapping to Smaller Hash GMMs using KL Divergence

Ideally, we would like the hash GMM to capture similar features as in the GMM-UBM. In our first method, we propose training a hash GMM using the same features vectors and methodology used to train the UBM except that the hash GMM has fewer component densities than the UBM. For instance, if our GMM-UBM consisted of 1024 component densities, we would train a hash GMM with perhaps 32 mixture components using the expectation maximization method. The components in the GMM-UBM are then mapped to components within the hash GMM using the symmetrized KL divergence. For multivariate normal distributions, this symmetrized KL divergence can be expressed as

$$d(f,g) = \frac{1}{2}\text{trace}\Big[\left(\boldsymbol{\Sigma}_f^{-1} + \boldsymbol{\Sigma}_g^{-1}\right)\left(\boldsymbol{\mu}_f + \boldsymbol{\mu}_g\right)\left(\boldsymbol{\mu}_f + \boldsymbol{\mu}_g\right)^T + \boldsymbol{\Sigma}_f\boldsymbol{\Sigma}_g^{-1} + \boldsymbol{\Sigma}_f^{-1}\boldsymbol{\Sigma}_g - 2\mathbf{I}\Big] \quad (8)$$

where μ_f, μ_g is the mean of f and g, respectively; $\boldsymbol{\Sigma}_f, \boldsymbol{\Sigma}_g$ is the covariance of f and g, respectively; and $\mathbf{I}$ is the identity matrix. Thus, a shortlist can be generated that maps components

from the hash GMM into clusters of components of the GMM-UBM in which the mapping is determined by minimizing (8) between the components in the hash GMM and the components in the GMM-UBM.

3.2. k-means Clustering using Divergence

A second method we propose for creating a hash GMM uses k-means clustering with symmetric KL as the distance measure to cluster the component densities of the UBM. Only the component densities of the GMM-UBM are needed to perform the clustering. In the assignment step of the k-means algorithm, components of the GMM-UBM are assigned to cluster centroids using the symmetric KL divergence.

In the update step of k-means, we calculate the centroids as expectation centroids [7] using the N components that have been assigned to the c-th centroid with

$$\begin{aligned}\boldsymbol{\mu}_c &= \frac{1}{N}\sum_{n=1}^{N} E\left[\mathbf{x}_n\right] \\ &= \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\mu}_c \end{aligned} \tag{9}$$

and

$$\begin{aligned}\boldsymbol{\Sigma}_c &= \frac{1}{N}\sum_{n=1}^{N} E\left[\left(\mathbf{x}_n - \boldsymbol{\mu}_c\right)\left(\mathbf{x}_n - \boldsymbol{\mu}_c\right)^T\right] \\ &= \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\Sigma}_n + \boldsymbol{\mu}_n\boldsymbol{\mu}_n^T - \boldsymbol{\mu}_c\boldsymbol{\mu}_c^T. \end{aligned} \tag{10}$$

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

$$w_i' = \sum_{n=1}^{N} w_n. \tag{11}$$

After the k-means algorithm has terminated, we use the centroids as the components of our hash GMM with weights determined by (11). Finally the shortlist mapping hash GMM components to GMM-UBM components is determined by minimizing (8) between the components of the UBM and the hash GMM.

3.3. KL GMM Reduction

Our final proposal for creating the hash GMM uses a method of merging component densities in the GMM-UBM known as *GMM reduction*. Recently, Runnalls [8] proposed a KL-based approach to GMM reduction. His approach was to successively merge pairs of mixture components within a GMM, 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 the two GMMs, the author does present an upper bound on the divergence and it is this upper bound that he attempts to minimize. He shows that divergence of the mixture following the merge from the mixture before the merge is bounded by

$$B[(w_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), (w_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)] = \frac{1}{2}\left[(w_i + w_j)\log\det\boldsymbol{\Sigma}_{ij} - w_i\log\det\boldsymbol{\Sigma}_i - w_j\log\det\boldsymbol{\Sigma}_j\right] \tag{12}$$

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

$$w_{ij} = w_i + w_j \tag{13}$$

$$\boldsymbol{\mu}_{ij} = w_{i|ij}\boldsymbol{\mu}_i + w_{j|ij}\boldsymbol{\mu}_j \tag{14}$$

$$\begin{aligned}\boldsymbol{\Sigma}_{ij} &= w_{i|ij}\boldsymbol{\Sigma}_i + w_{j|ij}\boldsymbol{\Sigma}_j \\ &= +w_{i|ij}w_{j|ij}\left(\boldsymbol{\mu}_i + \boldsymbol{\mu}_j\right)\left(\boldsymbol{\mu}_i + \boldsymbol{\mu}_j\right)^T\end{aligned} \tag{15}$$

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.

4. Experiments and Results

We have performed our experiments using the NIST2002 speaker recognition evaluation (SRE) corpus, single speaker cellular data. The UBM was trained by using data from both the Switchboard II Phase 1 corpus and the Switchboard Cellular Part 2 corpus. Each training file within NIST2002 was segmented into 10 utterances 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 cepstral vectors were processed with RASTA filtering. Delta-cepstral coefficients were then calculated using a 5 sample window length. These 19 delta-cepstral coefficients were concatenated to the cepstral vector to generate a 38-dimensional vector. Finally these cepstral/delta-cepstral vectors were processed with feature warping to generate our feature vectors.

In order to evaluate the efficiency of the proposed clustering methods, we chose as our metric the average number of *a posteriori* probability calculations in (4) versus EER. This is somewhat conservative in that it accurately reflects the reduction in the calculations of (4) for both hash and UBM but it does not reflect the total reduction the calculation of sufficient statistics (6) and (7). In order to include this further reduction, it would be necessary to determine the cost of basic operations such as addition, multiplication, and division which can be hardware/implementation specific.

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

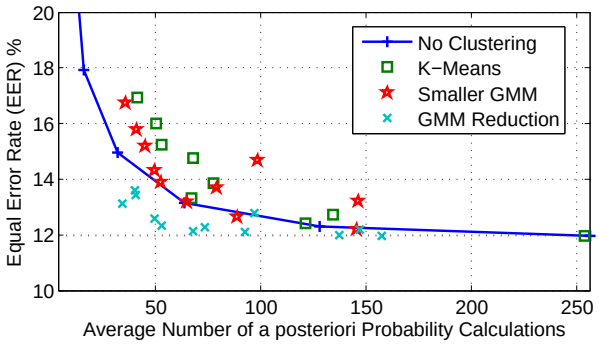

Figure 2: Results for Clustering a 256 Component UBM in both Training and Testing

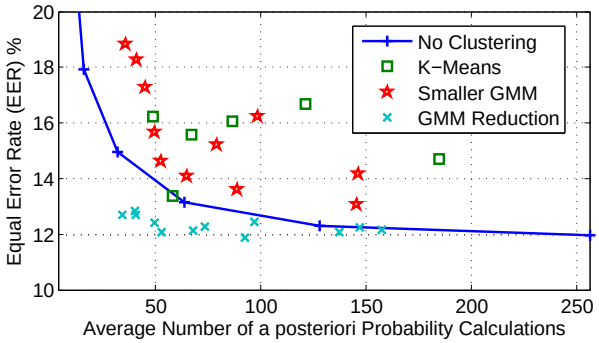

Figure 3: Results for Clustering a 256 Component UBM in Testing Only

achieved with this system to a GMM-UBM system with 32 mixture components.

In our experiments we simulated clusters of 4, 8, 16, and 32. Further, we simulated scenarios of 1, 2, and 4 as the maximum number of clusters chosen. Finally, for all cases we implemented the clustering during the testing stage but simulated the training stage with and without clustering.

Figs. 2 and 3 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. Clearly, for the majority of cases, the clustering methods did not perform as well as simply implementing the reduced size GMM-UBM. The exception was implementing the clustering using the GMM reduction method as described by Runnalls.

In general, performance is degraded when clustering is used only for testing (as opposed to training and testing). In particular, the k-means based clustering method 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, 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%. This improvement is very minimal and does not necessarily mean that we would generally achieve improved performance. The reason we are able to achieve no degradation is that perhaps the clustering procedure is successfully selecting all the components needed to perform the MAP adaptation reasonably well.

In addition to evaluating our method with a 256 component UBM we also evaluated it with a 1024 component UBM. In this case, 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 with respect to the 1024 component UBM.

5. Conclusion

In this paper, we presented a method for reducing the computational load of the support vector machine (SVM) using GMM supervectors 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 fewer *a poseteriori* probability calculations 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.

Future work could include intelligently adjusting the number of clusters to adapt based on the likelihood scores of the clusters. Further, it might also be worthwhile to explore potential gains of using a tree-based structuring algorithm.

6. References

[1] D. A. Reynolds, "An Overview of Automatic Speaker Recognition Technology," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 4, May 2002, pp. 4072–4075.

[2] G. Sarkar and G. Saha, "Analysis of Distance Measures for Pre-Quantization before Feature Extraction in Automatic Speaker Recognition," in *IEEE India Conference (INDICON)*, Dec. 2009, pp. 1 –4.

[3] R. Auckenthaler and J. S. Mason, "Gaussian Selection Applied to Text-Independent Speaker Verification," in *Proc. IEEE Speaker and Language Recognition Workshop (Odyssey)*, 2001, pp. 83–88.

[4] B. Xiang and T. Berger, "Efficient Text-Independent Speaker Verification with Structural Gaussian Mixture Models and Neural Network," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 5, pp. 447 – 456, Sep. 2003.

[5] W. Campbell, D. Sturim, and D. Reynolds, "Support Vector Machines using GMM Supervectors for Speaker Verification," *IEEE Signal Process. Lett.*, vol. 13, no. 5, pp. 308 – 311, May 2006.

[6] S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy, "SVM and Kernel Methods Matlab Toolbox," Perception Systemes et Information, INSA de Rouen, Rouen, France, 2005.

[7] T. Myrvoll and F. Soong, "Optimal Clustering of Multivariate Normal Distributions using Divergence and its Application to HMM Adaptation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 1, Apr. 2003, pp. 552–555.

[8] A. Runnalls, "Kullback-Leibler Approach to Gaussian Mixture Reduction," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 43, no. 3, pp. 989–999, Jul. 2007.