Towards a More Efficient SVM Supervector Speaker Verification System using Gaussian Reduction and a Tree-Structured Hash

Richard D. McClanahan[1,2], *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

Speaker verification (SV) systems that employ *maximum a posteriori* (MAP) adaptation of a Gaussian mixture model (GMM) universal background model (UBM) incur a significant test-stage computational load in the calculation of *a posteriori* probabilities and sufficient statistics. We propose a multi-layered hash system employing a tree-structured GMM which uses Runnalls' GMM reduction technique. The proposed method is applied only to the test stage and does not require any modifications to the training stage or previously-trained speaker models. With the tree-structured hash system we are able to achieve a factor of $8\times$ reduction in test-stage computation with no degradation in accuracy. Furthermore, we can achieve computational reductions greater than $21\times$ with less than 7.5% relative degradation in accuracy.

Index Terms: speaker recognition, gmm reduction, clustering methods, tree structures

1. Introduction

In recent literature, there has been a significant amount of research aimed at improving the efficiency of speaker recognition systems (SR). In general, most of the recent research in efficient SR can be categorized by improvements in one of three layers: frame layer, Gaussian layer, and utterance layer [1]. Typically, within the frame layer, the algorithms focus on reducing the frame rate by using some method to choose MFCC feature vectors to discard. Within the Gaussian layer, the goal is to reduce the computational complexity required to score each input feature vector against the GMM-UBM or speaker dependent GMM [1]. Finally, there is a set of utterance/speaker level algorithms which includes reducing the search space of individual speakers in SR systems [2, 3], fast utterance scoring [4], or efficiently compensating for channel/scoring effects [5].

There exists a substantial amount of research in the realm of reducing the computational burden within the Gaussian layer which is the focus of this paper. Some of the earliest research was directed at determining the effects of simply using GMMs with fewer components. For example, McLaughlin *et al.* [6] studied the effects of varying model orders of as little as 16 components up to 2048 components achieving less that 1% absolute degradation with a $4\times$ processing gain.

Auckenthaler and Mason [7] used Gaussian selection to reduce the number of Gaussian component densities that each feature vector was scored against. In the paper, the authors described three different methods for performing Gaussian selection. These methods relied on creating a smaller model (GMM) called a hash model and a shortlist (or mapping) between the components of the hash model and the components of the original GMM-UBM. The authors achieved "small degradation" with a computational reduction by a factor of 16.

Another approach to reducing the number of Gaussian components involves using hierarchical Gaussian Mixture Models (HGMMs) such as was done by Xiang and Berger [8]. In their system, the researchers investigated HGMMs with up to four layers where each layer represents a GMM of varying resolution. Fast scoring is achieved by scoring each input feature vector against a lower resolution GMM, whose components are linked to components in the next higher resolution layer GMM. The structural UBM (a HGMM) is searched by iteratively following the highest scoring components and their children for each layer. The structural GMM (SGMM) for the target speaker is scored using only the path that was taken in the search down the SBM. Finally, a decision is made using a neural network that processes the scores from different layers achieving a $17\times$ processing reduction with 5% relative degradation.

A significantly different approach was taken by Ye and Mak who employed discrete densities for fast GMM computation. In [9], the authors examined using scalar quantization and discrete densities. Each dimension of the input feature vector is first scalar quantized to a codeword. These codewords are used to index into a codebook containing pre-computed discretized probabilities. The authors investigated using both a high-density discrete model (HDMM) and a discrete mixture model (DMM) where the former is a single discrete density and the latter is a mixture of discrete densities. The HDDM method achieved a speedup by a factor of 30 with a 1.5% absolute degradation in EER. In [10], Ye and Mak extended the work to use subvector quantization of the feature vectors.

In [11], Mohammadi and Saeidi used a scalar sorting function to select Gaussian components for likelihood calculations. In their work, the authors sorted the components of the GMM-UBM with the goal that "neighboring" feature vectors would provide almost neighboring values of s_t where $s_t = f(x_1, x_2, \ldots, x_D)$ is a scalar function. In their paper, the authors used a sorting function $f(\cdot)$ which was simply the sum of the input vector reducing computational costs by $3.5\times$ without degradation. In [12], Saeidi *et al.* extend upon this work by using a slightly more complicated sorting function and the implementation of two orthogonal functions.

A somewhat elegant approach in its simplicity is the Gaussian prediction method used by Tydlitát *et al.* [13]. In Gaussian prediction, given the best scoring component for the current feature vector, an attempt is made to predict a subset of Gaussian components against which the next feature vector will be scored. In their work, the authors reduced the number of components for scoring and adaption by a factor of four.

More recently, in [14], we presented a hash based system similar to [7] but using a powerful Gaussian-reduction approach to construct the hash. We demonstrated that by training a hash model and shortlist mapping using GMM reduction as described by Runnalls in [15] that we could achieve a factor of 2.8 reduction in *a posteriori* calculations without loss in accuracy. The hash model system allowed us to intelligently reduce the complexity of our statistical scoring of feature vectors. Within the hash based system the computational reduction was limited due to the fact that the hash model itself requires some computational burden.

In this paper we extend our use of Gaussian reduction to form tree-structured GMMs to further reduce the computation required for calculating the sufficient statistics. We show that using a tree-structure we are able to achieve significantly more reduction in computation while incurring a small degradation in recognition performance.

The remainder of the paper is organized as follows. In Section 2, we review training hash GMMs and shortlists using GMM reduction. Next, in Section 3, we describe extending the GMM reduction method to form tree-structures. In Section 4, we present the performance of the GMM-tree based system. Section 5 presents our conclusions and thoughts on future work.

2. Hash GMM System

2.1. Hash System Description

The goal of using a hash GMM with a shortlist mapping to components within the GMM-UBM is ultimately to reduce the total number of component likelihood calculations and the number of components used in the update of sufficient statistics. It should be noted that the hash GMM effectively represents a lower resolution version of the UBM. Figure 1 shows the principle of using a hash system. Individual input feature vectors, $\mathbf{x}_t$, are initially scored against the component densities of the hash GMM with model parameters $\{w_i^{\mathrm{h}}, \boldsymbol{\mu}_i^{\mathrm{h}}, \boldsymbol{\Sigma}_i^{\mathrm{h}}\}$ representing component weight, mean vector, and covariance matrix respectively and where the superscript denotes hash parameters. Based on these probabilistic scores, a subset of component densities within the GMM-UBM having the model parameters $\{w_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}$ are selected for scoring and adaptation. The components in the UBM are selected with the shortlist mapping.

The GMM-UBM is trained with M multivariate Gaussian component densities while the hash GMM is formed with N components where $N < M$. After scoring a feature vector against the hash model, the $B < N$ highest scoring hash component densities are identified and their shortlists are used to select the components within the UBM.

2.2. Hash GMM and Shortlist Training

The method we use to train our hash GMM is described in [14] and is based on GMM reduction [15]. This approach successively merges pairs of mixture components, replacing the pairs with a single Gaussian component. The criterion for selecting pairs to merge is based on minimizing the Kullback-Leibler (KL) divergence between the GMM pre-merge and the resulting GMM post-merge. Although a closed form solution does not exist for calculating the KL divergence between two GMMs, [15] presents an upper bound on the divergence between the pre-merger and post-merger GMMs and it is this upper bound that the algorithm attempts to minimize. Letting λ_i and λ_j represent the i^{th} and j^{th} component densities of a GMM, this KL

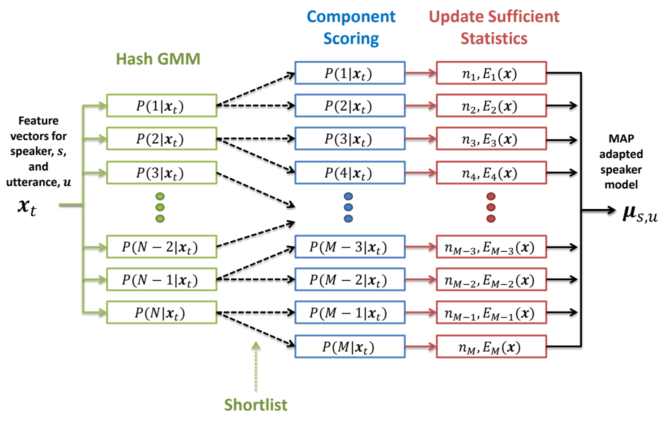

Figure 1: Single Layer Hash GMM System Trained using Gaussian Mixture Reduction [14]

divergence is bounded by [15]

$$\mathrm{KL}(\lambda_i||\lambda_j) \le \frac{1}{2}\left[(w_i + w_j)\log\det\left(\boldsymbol{\Sigma}_{ij}\right) - w_i \log\det\left(\boldsymbol{\Sigma}_i\right) - w_j \log\det\left(\boldsymbol{\Sigma}_j\right)\right] \quad (1)$$

where $\mathrm{KL}(\lambda_i||\lambda_j)$ is computed for every pair of component members with $i \neq j$ within the premerged GMM.

The two components that minimize $\mathrm{KL}(\lambda_i||\lambda_j)$ are selected for merger and are replaced by the moment-preserving merge [15]

$$\begin{aligned} w_{ij} &= w_i + w_j \\ \boldsymbol{\mu}_{ij} &= w_{i|ij}\boldsymbol{\mu}_i + w_{j|ij}\boldsymbol{\mu}_j \\ \boldsymbol{\Sigma}_{ij} &= w_{i|ij}\boldsymbol{\Sigma}_i + w_{j|ij}\boldsymbol{\Sigma}_j \\ &\quad + w_{i|ij}w_{j|ij}\left(\boldsymbol{\mu}_i + \boldsymbol{\mu}_j\right)\left(\boldsymbol{\mu}_i + \boldsymbol{\mu}_j\right)^T \end{aligned} \quad (2)$$

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

This 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 GMM reduction consists of the following two steps repeated $M - N$ times:

1. *Calculate bound*: Calculate the upper bound (1) on the divergence for all pairs of components i and j within the merged GMM from the last iteration.
2. *Merge Pairs*: Choose the two pairs of components with indices i and j that minimize the bound and merge the pair using the moment preserving merger of their weights, mean vectors, and covariance matrices (2).

3. GMM Tree-Structures

With a single layer hash system, the upper bound of the processing gain is $\frac{\sqrt{M}}{2}$–assuming uniformly-distributed cluster sizes. We can reduce the number of components in the UBM selected

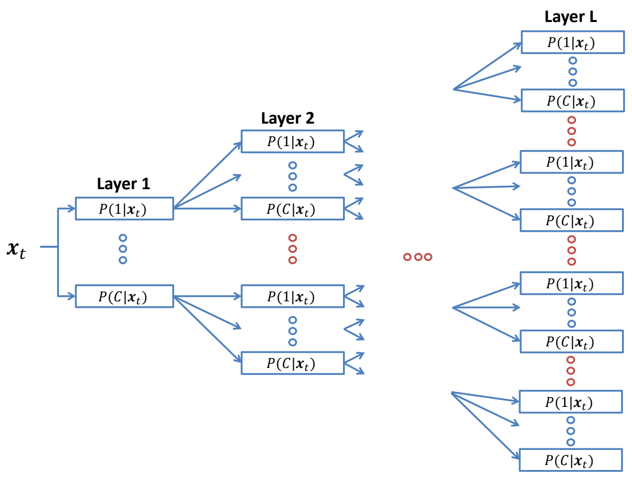

Figure 2: Tree-Structured Hash GMM System with each Layer Trained Using Gaussian Component Reduction of the Components of the Higher Resolution GMMs

for scoring by choosing a larger hash model. Though this decreases the number of UBM components for scoring, the effect is to increase the number of components in the hash for scoring. Within a single layer hash system, the best case processing gain can be achieved when the number of components within each cluster is the as same the number within the hash model.

Algorithm 1: Recursive Tree Training Algorithm

```
Input: GMM components $\{w_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}$ passed in from
       previous layer (or UBM components for top
       layer), $N$, and remaining number of layers $L_R$
Output: multi-layered GMM tree structure
Perform GMM reduction as in Section 2.2 using
$\{w_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}$ creating $C$ clusters ;
if $L_R = 0$ (last layer) then
    return GMM clusters from current layer ;
else
    $L_R = L_R - 1$ ;
    for $1 \leq$ cluster (node) $\leq C$ do
        if Number of components associated with
        current node and layer is less than $C$ then
            Continue to next node ;
        else
            Call Recursive Tree Training Algorithm
            with $\{w_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$, $C$, and $L_R$ where $k$
            indicates components associated with
            current node and layer ;
        end
    end
end
```

Extending our hash system to a multi-layered tree-structured system such as depicted in Fig. 2 allows us the opportunity to obtain even greater reduction in likelihood calculations. In building our tree structure, we have attempted to create a somewhat homogenous structure in which each parent node contains the same number of children C.

The tree forming algorithm, a recursive algorithm, which is performed once in training. is given in Algorithm 1. The algorithm is recursively called until L layers have been created.

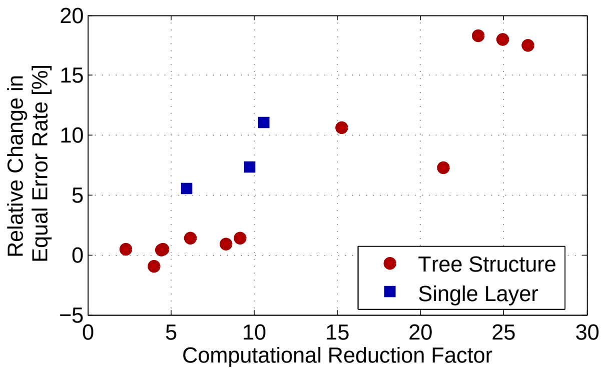

Figure 3: Single layer vs tree-structured hash. The tree-structured hash has higher computational reduction or less degradation in accuracy for fixed computational reduction.

4. Experiments and Results

We have performed our experiments using the NIST2005 speaker recognition evaluation (SRE) corpus [16]–eight-conversation training and one-conversation test. The UBM was trained using data from Switchboard II Phase 1 [17], Switchboard Cellular Part 2 [18], and the OGI National Cellular corpora [19]. Cohort speakers for the GMM supervector based SVM models were drawn from the Fisher English corpora [20, 21]. For front end processing, we calculated 13-dimensional MFCC feature vectors every 10 ms using a 20 ms Hamming window. The frequency content was limited to the range of 300-3140 Hz. These feature vectors were processed with RASTA filtering before delta-cepstral coefficients were computed to create 26-dimensional feature vectors that were $0/1$-normalized. These feature vectors were then used to adapt a 512-component gender-independent UBM.

With our baseline system we achieved an equal error rate (EER) of 10.71%. Because we are neither employing channel/session variability such as nuissance attribute project (NAP) nor are we applying score normalization such as zt-norm, we achieve degraded performance with respect to those systems that do. While we would expect our baseline system to perform better with these enhancements, we have chosen to forgo them in order to perform our simulations in a more timely manner.

In the training stage, supervectors for the cohort speakers as well as well as those calculated for the target speaker models were calculated using MAP adaptation without using a hash-tree front-end. During the test stage, different tree structures were evaluated by varying both the number of clusters per node and number of layers.

The relative change in EER as a function of the computational reduction factor of total *a posteriori* probability calculations performed–including both the UBM and tree nodes–is shown in Fig. 3 and Table 1. The results for a single hash layer system, such as in [14], are depicted by squares in Fig. 3 and rows with $L = 1$ in Table 1. The other results correspond to tree-structured GMM systems. There are two key observations that can be made with respect to the performance difference between a single layer and a tree-structure. First, with a tree structure we are able to achieve significantly more reduction in computation. Second, for comparable computational reductions, the

L	C	B	Reduction Factor	ΔEER %
9	2	1	24.93	17.97
8	2	1	23.47	18.30
4	4	1	26.46	17.51
4	4	2	4.53	0.45
3	**8**	**1**	**21.38**	**7.31**
3	8	2	8.31	0.92
3	**8**	**3**	**4.01**	**-0.94**
2	16	1	15.27	10.60
2	16	2	9.17	1.38
1	8	1	5.96	5.53
1	16	1	9.72	7.37
1	32	1	10.60	11.06

Table 1: Results of SV system using the proposed tree-structured hash. For the (3,8,3) parameters, there is a $4\times$ computational reduction with no loss in accuracy and for the (3,8,1) parameters there is a $21\times$ computational reduction with only a small reduction in accuracy.

tree-structure has significantly less degradation than the single layer hash system.

Analyzing the tree-structure results in Table 1 ($L > 1$), for the (3,8,3) parameters there is a $4\times$ computational reduction with no loss in accuracy (actually there is a small improvement in EER) while for the (3,8,1) parameters there is a $21\times$ computational reduction with a 7.31% relative loss in EER. To wit, the proposed method is applied only to the test stage and does not require any modifications to the training stage or previously-trained speaker models.

5. Conclusions and Future Work

We have proposed a tree-structured hash GMM system based on Runnalls' GMM reduction technique, for reducing test-stage computation in a SV system. As compared to to a single-layer hash, we are able to achieve greater reductions in computation for a fixed accuracy. For example, we are able to achieve a $4\times$ computational reduction with no loss in accuracy and as much as $21\times$ computational reduction with a small relative loss in EER. The proposed method is attractive from an implementation point-of-view in that it is applied only to the test stage and does not require any modifications to the training stage or previously-trained speaker models.

Our future work includes using the multi-layers of the tree-structure to form a multi-resolution supervector that use intra-layer *a posteriori* probabilities to MAP adapt the GMMs of the hash layers to create supervectors for each layer. Other work may include using a session variability compensation technique, such as nuisance attribute projection, to reduce the variability between normal supervectors and those created using a hashing system.

Other future work includes adapting the method discussed in this paper to calculate the sufficient statistics in a joint factor analysis [22] or total variability system [4].

6. References

[1] A. Chan, J. Sherwani, R. Mosur, and A. Rudnicky, "Four-layer categorization scheme of fast GMM computation techniques in large vocabulary continuous speech recognition systems," in *Proc. of ICSLP*, 2004.

[2] V. R. Apsingekar and P. L. D. Leon, "Speaker model clustering for efficient speaker identification in large population applications," *IEEE Trans. Audio, Speech, Language Process.*, vol. 17, no. 4, pp. 848–853, May 2009.

[3] A. K. Sarkar, S. Umesh, and J. F. Bonastre, "Computationally efficient speaker identification using fast-MLLR based anchor modeling," in *ICASSP*. IEEE, 2012, pp. 4357–4360.

[4] N. Dehak, R. Dehak, P. Kenny, N. Brummer, P. Ouellet, and P. Dumouchel, "Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification," in *Proc. Interspeech Conf*, 2009.

[5] T. Kinnunen, E. Karpov, and P. Fränti, "Efficient online cohort selection method for speaker verification," in *Proc. of ICSLP*, vol. 3, 2004, pp. 2401–2402.

[6] J. McLaughlin, D. A. Reynolds, and T. P. Gleason, "A study of computation speed-ups of the GMM-UBM speaker recognition system," in *EUROSPEECH*, 1999.

[7] R. Auckenthaler and J. S. Mason, "Gaussian selection applied to text-independent speaker verification," in *Proc. Speaker Odyssey*, 2001, pp. 83–88.

[8] B. Xiang and T. Berger, "Efficient text-independent speaker verification with structural gaussian mixture models and neural network," *IEEE Trans. on Speech and Audio Process.*, vol. 11, no. 5, pp. 447 – 456, sept. 2003.

[9] G. Ye, B. Mak, and M. Mak, "Fast GMM computation for speaker verification using scalar quantization and discrete densities," in *Proc. Interspeech Conf*, 2009, pp. 2327–2330.

[10] G. Ye and B. Mak, "The use of subvector quantization and discrete densities for fast GMM computation for speaker verification," in *Proc. Interspeech Conf*, 2010, pp. 1481–1484.

[11] H. R. S. Mohammadi and R. Saeidi, "Efficient implementation of gmm based speaker verification using sorted gaussian mixture model," *Proc. EUSIPCO06*, pp. 4–8, 2006.

[12] R. Saeidi *et al.*, "Joint frame and gaussian selection for text independent speaker verification," in *ICASSP 2010*, march 2010, pp. 4530 –4533.

[13] B. Tydlitát, J. Navratil, J. W. Pelecanos, and G. N. Ramaswamy, "Text-independent speaker verification in embedded environments," in *ICASSP*, vol. 4, april 2007, pp. IV–293 –IV–296.

[14] R. D. McClanahan and P. L. De Leon, "Mixture component clustering for efficient speaker verification," in *Proc. Interspeech Conf*, 2012.

[15] A. R. Runnalls, "Kullback-Leibler approach to gaussian mixture reduction," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 43, no. 3, pp. 989 –999, july 2007.

[16] N. M. I. Group, *2005 NIST Speaker Recognition Evaluation Test Data*. Linguistic Data Consortium, Philadelphia, 2011.

[17] D. Graff, A. Canavan, and G. Zipperlen, *Switchboard-2 Phase I*. Linguistic Data Consortium, Phildelphia, 1998.

[18] D. Graff, K. Walker, and D. Miller, *Switchboard Cellular Part 2 Audio*. Linguistic Data Consortium, Phildelphia, 2004.

[19] R. A. Cole *et al.*, *CSLU: National Cellular Telephone Speech Release 2.3*. Linguistic Data Consortium, Phildelphia, 2008.

[20] C. Cieri *et al.*, *Fisher English Training Speech Part 1 Speech*. Linguistic Data Consortium, Phildelphia, 2004.

[21] C. Cieri *et al.*, *Fisher English Training Part 2, Speech*. Linguistic Data Consortium, Phildelphia, 2005.

[22] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Trans. Audio, Speech, Language Process.*, vol. 15, no. 4, pp. 1435–1447, May 2007.