Data-driven Gaussian Component Selection for Fast GMM-Based Speaker Verification

Ce Zhang¹, Rong Zheng¹, Bo Xu¹,²

¹Digital Content Technology Research Center, Institute of Automation
²National Lab of Pattern Recognition, Institute of Automation
Chinese Academy of Sciences, Beijing, China
{czhang, rzheng, xubo}@hitic.ia.ac.cn

Abstract

In this paper, a fast likelihood calculation of Gaussian mixture model (GMM) is presented, by means of dividing the acoustic space into disjoint subsets and then assigning the most relevant Gaussians to each of them. The data-driven approach is explored to select Gaussian component which guarantees that the loss, brought by pre-discarding most useless Gaussians, can be easily controlled by a manual set parameter. To avoid the rapid growth of the index table size, a two level index scheme is proposed. We adjust several set of parameters to validate our work which is expected to speed up the computation while maintaining the performance. The results of the experiments on the female part of the telephone condition of NIST SRE 2006 indicate that the speed can be improved up to 5 times over the GMM-UBM baseline system without performance loss.

Index Terms: speaker verification, fast GMM, GMM-UBM

1. Introduction

Gaussian mixture model (GMM) is among the most statistically mature method for generative modeling which is commonly used in speaker verification. In order to achieve high accuracy the GMMs usually consist of large number of mixture components, especially when the diagonal covariance matrices are used. The likelihood calculation, which is the essential step in the GMM-based speaker verification process, is also included in the channel compensation module, such as Factor Analysis [1] [2] [3], for collecting the sufficient statistics. It is proved that the likelihood calculation of GMMs dominates the time of speaker verification process (85% in our baseline system), and it is worth investigating a fast and effective calculation to improve the speed of verification.

The traditional methods for fast GMM calculation have been investigated in [4], [5], [6]. In [4], the authors focused on the acoustic resolution of the universal background model (UBM)-reduction of the number of mixtures, and the temporal resolution of the input speech-reduction of the number of vectors. The authors [5] used a combination of scalar quantization and discrete density techniques to convert a continuous-density GMM to various discrete densities. The most recently proposed speedup method [6] used a cluster GMM to reduce the computation of sufficient statistics for factor analysis.

In this work, we propose a data-driven method for the Gaussian component selection. Two important characteristics of this study can be presented as follows. First, it is evident that while increasing the amount of training data, our data-driven approach is more reliable and stable in the statistical view. Second, because of the soft decision strategy used for the acoustic space resolution, a particular component is allowed to belong to more than one center/class around it.

The outline of this paper is organized as follows. Section 2 gives a brief introduction to the mathematical form of GMM model. In section 3, we describe the one and two layer index and then explain the generation of the candidate Gaussian list. Finally a measure of the effectiveness called top n hit frequency is introduced to evaluate the proposed method. We present our experimental configuration and results in section 4 and 5, respectively.

2. GMM related background

GMM is a mixture model which is composed of $C$ multi-variate Gaussian densities. The likelihood function for a $D$-dimensional feature vector $x$ is defined as follows:

$$ p(x|\lambda) = \sum_{i=1}^{C} \omega_i p_i(x) $$ (1)

The mixture weights satisfy the constraint $\sum_{i=1}^{C} \omega_i = 1$. The individual multivariate Gaussian density, $p_i(x)$, is parameterized by a $D$-dimensional mean vector $\mu_i$ and a $D \times D$ covariance matrix $\Sigma_i$.

$$ p_i(x) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp\left\{-\frac{1}{2}(x - \mu_i)^{\top}\Sigma_i^{-1}(x - \mu_i)\right\} $$ (2)

Thus, the GMM probabilistic density is specified by a parameter $\lambda$ of the form $(\omega, \mu_i, \Sigma_i)$ where $i = 1, ..., C$.

Furthermore, let us introduce the posterior relating to a specified component $i$, that is:

$$ p(i|x) = \frac{\omega_i p_i(x)}{p(x|\lambda)} $$ (3)

where $p(i|x)$ represents the posterior probability of the event that the vector $x$ is accounted for by the mixture component $i$. 

---

Supported by the National Natural Science Foundation of China (Grant No. 90820303)
3. Gaussian selection

According to (1) we can conclude that two factors dominate the computations in the GMM-based speaker verification system: the number of Gaussians in the GMM and the number of feature vectors to be scored in a test utterance. We only focus on the first factor about reducing the Gaussian numbers in this paper. In [7], the authors pointed out that when a large GMM is evaluated for a feature vector, only a few of the mixtures contribute significantly to the likelihood value which can be approximated very well using only the top \( n \) best scoring mixture components. Thus, it is straightforward to consider how to directly find the top \( n \) components with much lower computational cost instead of calculating all the mixtures.

3.1. Acoustic space division

In speaker verification, the GMM represents a distribution over a large space while a single feature vector \( x \) will be near only a few components of that GMM. So we first divide the acoustic space into several separate sets, each of which is more compact in Euclidean space.

The first approach is to use K-means algorithm to find subspace centers, which is called one layer index. Given \( T \) training vectors \( X = \{x_1, ..., x_T\} \), we run K-means algorithm to obtain \( L \) centers \( \{m_1, ..., m_L\} \) and the corresponding \( L \) subsets of \( X \), \( \{X_1, ..., X_L\} \), where

\[
X_i = \{x_t | ||x_t - m_i|| = \min_k ||x_t - m_k||, 1 \leq t \leq T\}
\]  
and \( 1 \leq i \leq L \).

It is obvious that the within-cluster sum of squares (WCSS), denoted by \( \sum \sum_{x_t \in X_i} ||x_t - m_i||^2 \), will be smaller with a larger \( L \). We prefer a smaller WCSS which represents the compactness of each cluster, and the center as an index will be more representative and precise. On the other hand, too large \( L \) is unacceptable because the larger \( L \) will increase the size of index table. For the reason that the time for looking up table can not be neglected in our case, we propose a more effective method as follows.

In order to control the size of index table, we utilize the idea of multi-level index in the second approach. For simplicity we only use two layer index which is determined by \( L_1 \) and \( L_2 \) respectively. There are totally \( L_1 \times L_2 \) centers and the costs for looking up table decrease from \( O(L) \) to \( O(L_1 + L_2) \) where \( L_1 \times L_2 = L \). Notice that the optimal solution is obtained when \( L_1 = L_2 = \lfloor \sqrt{L} \rfloor \) if we want to obtain the same size of index as in the one layer case ([*]) returns the largest integer smaller than *.). Algorithm 1 shows the key steps of two layer division which is similar to the one layer case except for a secondary splitting.

3.2. Candidate Gaussians generation

After dividing the entire space into smaller ones, we view the center as index to the associated cluster. For each index \( i (1 \leq i \leq L) \), we first define the zeroth Baum-Welch statistics \( N_t \) (a vector of length \( C \)) for the one layer case whose elements are,

\[
N_i(c) = \sum_{x_t \in X_i} p(c|x_t)
\]

Secondly, let \( \Phi \) be the associated Gaussian list of the \( i \)th index. We sort \( \Phi \) to get the descending array \( \Phi(1), ..., \Phi(J) \) and

<table>
<thead>
<tr>
<th>Algorithm 1 Two layer space division</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Require:</strong> ( L_1 &gt; 0 ) and ( L_2 &gt; 0 ) and ( T &gt; L_1 \times L_2 )</td>
</tr>
<tr>
<td>1: Find ( L_1 ) centers ( m_{11}, ..., m_{1L_1} ) of ( X = {x_1, ..., x_T} )</td>
</tr>
<tr>
<td>2: for ( i = 1 ) to ( L_1 ) do</td>
</tr>
<tr>
<td>3: ( X_i \leftarrow {x_t</td>
</tr>
<tr>
<td>4: Find ( L_2 ) centers ( m_{i1}, ..., m_{iL_2} ) of ( X_i )</td>
</tr>
<tr>
<td>5: for ( j = 1 ) to ( L_2 ) do</td>
</tr>
<tr>
<td>6: ( X_{ij} \leftarrow {x_t</td>
</tr>
<tr>
<td>7: end for</td>
</tr>
<tr>
<td>8: end for</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 2 Candidate Gaussian list generation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Require:</strong> ( L_1 &gt; 0 ) and ( L_2 &gt; 0 ) and ( 0 &lt; \alpha \leq 1 )</td>
</tr>
<tr>
<td>1: for ( i = 1 ) to ( L_1 ) do</td>
</tr>
<tr>
<td>2: for ( j = 1 ) to ( L_2 ) do</td>
</tr>
<tr>
<td>3: for ( c = 1 ) to ( C ) do</td>
</tr>
<tr>
<td>4: ( N_{ij}(c) \leftarrow \sum_{x_t \in X_{ij}} p(c</td>
</tr>
<tr>
<td>5: end for</td>
</tr>
<tr>
<td>6: descending ( \Phi_{ij}(k_1), ..., \Phi_{ij}(k_C) )</td>
</tr>
<tr>
<td>7: Init: ( p \leftarrow 1 ), ( S \leftarrow 0 )</td>
</tr>
<tr>
<td>8: repeat</td>
</tr>
<tr>
<td>9: ( \Phi_{ij} \leftarrow \Phi_{ij} {k_p} )</td>
</tr>
<tr>
<td>10: ( S \leftarrow S + N_{ij}(k_p) )</td>
</tr>
<tr>
<td>11: ( p \leftarrow p + 1 )</td>
</tr>
<tr>
<td>12: until ( S \geq \alpha )</td>
</tr>
<tr>
<td>13: end for</td>
</tr>
<tr>
<td>14: end for</td>
</tr>
</tbody>
</table>

Note that the generation of \( K_i \) is independent of each other which means a specified component can be selected as candidate for several centers. It is clear that the value \( \overline{K} = \frac{1}{L} \sum_{i=1}^{L} K_i \) reflects the average number of remaining Gaussians. The smaller \( \overline{K} \) leads to more efficient calculation, and however it brings a higher risk of discarding useful components.

3.3. Top-\( n \) hit frequency

To evaluate the effectiveness of the Gaussian reduction method, we define the hit frequency of top \( n \) Gaussian which is denoted by \( \Gamma(n) \). Given an evaluation data set \( X = \{x_1, ..., x_T\} \), we calculate the posterior (3) for each feature vector and let \( \Theta_n(x_t) \) be the collection of top \( n \) Gaussian for \( x_t \). Then we look up the index table to find the nearest center \( i \) to \( x_t \) and
define an auxiliary indicator function,
\[ I_n(x_t) = \begin{cases} 1 & \text{if } \Theta_n(x_t) \subseteq \Phi_i \\ 0 & \text{otherwise} \end{cases} \tag{7} \]
where \( \Phi_i \) is the candidate Gaussian list of center \( i \) mentioned in section 3.1 and 3.2. Note that \( I_n(x_t) = 1 \) only if \( \Theta_n(x_t) \) is totally contained in \( \Phi_i \). Finally, the hit frequency of top \( n \) Gaussian can be formulated as
\[ \Gamma(n) = \frac{\sum_{t=1}^{T} I_n(x_t)}{T} \tag{8} \]

The significance of \( \Gamma(n) \) is that it provides a statistical prediction that the top \( n \) Gaussian of a new vector \( x \) will be totally contained in the Gaussian list \( \Phi_i \) with the probability \( \Gamma(n) \), where \( x \) belongs to the \( i \)-th center. We will analyze \( \Gamma(n) \) in detail in section 5.2.

4. Experiment setup

4.1. Data description

First, we train the UBM model with 1024 Gaussian components using 370 recordings of the female part of NIST SRE 2004 telephone data. We make use of the same data set as UBM for the acoustic space division and index generation (detailed in section 3), which is composed of 4, 463, 852 feature vectors.

To deal with the intersession variability, the feature domain intersession compensation (FDIC) [8] is employed with an eigen-channel matrix of rank 40 in both training and testing phase. The eigen-channel space is trained on Switchboard II Phase 2, Switchboard Cellular Parts 2, SRE 2004 and SRE 2005 telephone data including 9766 recordings from 738 female speakers and we only make use of those speakers for which 5 or more recordings are available.

The evaluation results of our experiments are reported on the female part of the telephone condition of NIST SRE 2006 corpus [9], which is composed of 461 target speakers and 2121 test utterances. Both training and test conversations have an average duration of 5 minutes and there are no cross-gender trials.

The verification decision scores are normalized using z-t norm. We use 200 female t-norm models and 500 female z-norm utterances, which are derived from NIST SRE 2002, 2004 and 2005 evaluation data.

4.2. Front-End processing

We extract the first 12 Mel frequency cepstrum coefficients (MFCC) together with a log energy feature using a 25 ms Hamming window and a 10 ms frame advance. Mean and variance normalization is used to remove the linear channel effects. Delta and double delta coefficients are then calculated using a 5 frames window and then we obtain a set of 39-dimensional feature vectors. We train bi-gauss model to prune out silence and low energy frames for Speech Activity Detection module and then the remained feature vectors are subjected to feature warping using a 3-second sliding window. Finally, FDIC is performed to deal with the channel variability.

4.3. GMM-UBM

We implement the most fundamental speaker verification system called GMM-UBM [7] to validate the feasibility of our proposed method for fast GMM calculation. Diagonal covariance matrix is used for the Gaussians of UBM and maximum a posteriori (MAP) estimation is used for the adaptation of speaker model from the UBM with a relevance factor 16. For each feature vector in the test phase, we first determine the top 8 scoring mixtures in the UBM and then score the vector against only the 8 components in the adapted speaker model. We select the top 8 Gaussian from all the components for the baseline system and from the corresponding candidate Gaussian list for the proposed method.

4.4. Software and hardware

The code is implemented with C and all the experiments are carried out on the machine Dell E5410, with Intel Xeon 2.33GHz CPU, and 3.99G RAM.

5. Results

5.1. The average number of remaining Gaussians

We implement the index size from 16 to 128 with one layer index and \( 16 \times 16, 32 \times 16, 32 \times 32 \) for two layer. The average number of Gaussians \( K \) as a function of index size \( L \) is plotted in Figure 1 with multiple curves according to different \( \alpha \). Two remarks are pointed out for this figures. First, increasing the confidence level \( \alpha \) result in increased average Gaussian number \( K \). It is consistent with (6) where \( K \) is an increasing function of the variable \( \alpha \). Second, for a fixed \( \alpha \), \( K \) monotonically decreases with the growth of index size \( L \). It can be explained by the fact that the larger \( L \) means more detailed division of the acoustic space and more compact subsets(smaller WCSS). As a result, the vectors belong to the same subset are prone to be aligned to the same Gaussians.

![Figure 1: The average number of Gaussians versus the index size for different confidence levels](image)

5.2. Top-n hit frequency

Figure 2 presents the hit frequency (8) of top \( n \) Gaussian according to different parameters. We choose 7 set of parameters by varying both \( L \) and \( \alpha \) as indicated with the circles in Figure 1, and calculate \( \Gamma(n) \), \( 1 \leq n \leq 20 \) using the evaluation data mentioned in section 4.1. The reason why we choose these set
of parameters is that we have to balance the trade-off between speed and performance due to different $K$. As the results in Figure 2 demonstrate, it is obvious that there is a turning point $n_o$ corresponding to different $\alpha$ where $\Gamma(n)$ remains constant before $n_o$ (e.g. $\Gamma(6) \equiv 0.86$, for $1 \leq n \leq n_0, 90$, where $n_0, 90 = 5$). In the region of $n > n_o$, $\Gamma(n)$ monotonically decreases with the increasing of $n$. This reveals that our method is expected to consistently capture the top $n_o$ Gaussian with the probability of 0.86. The difference between 0.86 and $\alpha$ is probably caused by the mismatch of training and test data. While the training set is sufficiently large, the constant value(0.86 here) will be close to $\alpha$.

Second, $\Gamma(n)$ decreases with the growth of $L$ for a fixed $\alpha$ and $n(n > n_o)$ because of the fact that a larger $L$ will lead to a smaller $K$ and as a result bring a higher risk of discarding useful components as mentioned in the last paragraph of section 3.2.

Finally, it seems that $\Gamma(n)$ is more sensitive to $\alpha$ than to $L$. We believe that the main function of $L$ is to control the average number of Gaussians $K$ while $\alpha$ influences both $K$ and $\Gamma(n)$.

![Figure 2: Hit frequency of top $n$ Gaussian.](image)

5.3. Comparative Results

We evaluate 6 configurations of the proposed method along with the baseline system by equal error rate (EER) and minimum value of NIST detection cost function (minDCF). The results including both performance and cost time (all the time are normalized by the time of baseline system) are presented in Table 1. Note that the minimum case of the last column is 0.15 in our experiment because the GMM calculation accounts for 85% of the total time of baseline system and we do nothing for the remaining 15% computation. As shown in the first three lines of Table 1, more than 3x speedup can be obtained with less than 0.1 absolute performance degradation if the one layer index is used. Furthermore, the usage of two layer strategy leads to 5x with nearly no performance loss.

<table>
<thead>
<tr>
<th>$L$</th>
<th>$\alpha$</th>
<th>EER(%)</th>
<th>minDCF</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>0.90</td>
<td>5.974</td>
<td>0.0272</td>
<td>0.28</td>
</tr>
<tr>
<td>128</td>
<td>0.90</td>
<td>5.998</td>
<td>0.0274</td>
<td>0.29</td>
</tr>
<tr>
<td>16×16</td>
<td>0.91</td>
<td>5.897</td>
<td>0.0271</td>
<td>0.31</td>
</tr>
<tr>
<td>16×16</td>
<td>0.92</td>
<td>5.877</td>
<td>0.0266</td>
<td>0.22</td>
</tr>
<tr>
<td>32×16</td>
<td>0.92</td>
<td>5.877</td>
<td>0.0269</td>
<td>0.21</td>
</tr>
<tr>
<td>baseline</td>
<td></td>
<td>5.877</td>
<td>0.0265</td>
<td>1</td>
</tr>
</tbody>
</table>

is required. We verify the feasibility of the proposed method by defining the hit frequency of top $n$ Gaussian, denoted as $\Gamma(n)$ and showing it as a function of $n$ for various combinations of $L$ and $\alpha$. Additionally, several configurations of $L$ and $\alpha$ are selected to compare the speed as well as performance in terms of EER and minDCF. The results show that nearly 5x speed improvement can be obtained without performance loss. We believe that the proposed method can be used in other large GMM related works, such as language recognition.

7. Acknowledgements

The authors would like to thank Jiaen Liang for a series of useful advice and discussion.

8. References


