The Use of Subvector Quantization and Discrete Densities for Fast GMM Computation for Speaker Verification

Guoli Ye and *Brian Mak*

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
{yeguoli,mak}@cse.ust.hk

Abstract

Last year, we showed that the computation of a GMM-UBM-based speaker verification (SV) system may be sped up by 30 times by using a high-density discrete model (HDDM) on the NIST 2002 evaluation task. The speedup was obtained using a special case of the product-code vector quantization in which each dimension is scalar-quantized in the construction of the discrete model. However, the speedup resulted in a drop of an absolute 1.5% in equal-error rate (EER). In this paper, our previous work is generalized to the use of subvector quantization (SVQ) in the construction of HDDM. For the same NIST 2002 SV task, the use of SVQ leads to an overall speedup by a factor of 8–25 with no significant loss in EER performance.

Index Terms: speaker verification, high-density discrete model, subvector quantization, split vector quantization

1. Introduction

Most state-of-the-art speaker verification (SV) systems use Gaussian mixture model (GMM) to represent the universal background model (UBM) and the speaker models (SM). For an SV system that employs log-likelihood ratio between SM and UBM to make the decision, its computational efficiency is largely determined by the GMM computation. The most common method to speedup GMM computation in such SV systems is to find out from the UBM the top N Gaussian components that give the highest likelihoods for an evaluating speech frame, and then compute the GMM log-likelihoods of the UBM and SM using only those N components [1, 2]. For large UBM and small N, it is claimed that the method reduces the number of Gaussian evaluations almost by half, with negligible effect on verification accuracy.

In our previous work [3], we investigated another way to adjust the acoustic resolution of a GMM to achieve fast GMM computation. We employed scalar quantization (SQ) and converted a continuous-density GMM to a high-density discrete model (HDDM). Because of the use of per-dimension SQ, a full-space codeword can be determined quickly in $O(d)$ time where d is the dimension of the full-space feature vectors, whereas the use of discrete density model reduces the subsequent GMM log-likelihood computation to a simple table-lookup. As a result, compared with GMM, HDDM gives a speedup of 30 times in the NIST 2002 SV task; however, the EER drops by an absolute 1.5%.

This research is supported by the Research Grants Council of the Hong Kong SAR under the grant numbers DAG05/06.EG43, HKUST617507, and HKUST617008.

In this paper, we will show that performance of HDDM can be improved to that of the baseline GMM-UBM system while maintaining its fast speed by using more codewords and subvector quantization[1] (SVQ). It is well-known that given the same number of bits, VQ is more efficient than SQ in the sense that it results in smaller quantization error. However, it requires more space to store a VQ codebook and a longer time to find a VQ codeword. A good trade-off between time and space requirement can be obtained by SVQ which is well proven in the work of LPC coding [4], subspace distribution clustering hidden Markov modeling [7], discrete mixture hidden Markov modeling [6], and so forth. We call the new HDDM, *subvector-quantized high-density discrete model* (SVQ-HDDM), and the old HDDM proposed in our previous work [3] that employs SQ will be called *scalar-quantized high-density discrete model* (SQ-HDDM) hereafter.

The major drawback of SVQ-HDDM is its larger model size compared to GMM. However, we will show later in the experiments that the model size, although larger, is acceptable for many real SV applications depending on its actual operation.

2. Subvector-Quantized High-Density Discrete Model (SVQ-HDDM)

Before the construction of an SVQ-HDDM, each d-dimensional acoustic vector x_t is first partitioned into L subvectors,

$$x_t = [x_{1t}, x_{2t}, \ldots, x_{Lt}]. \quad (1)$$

Subvectors of each partition are then vector-quantized to create an SVQ codebook for the partition. Full-space VQ codewords can then be constructed as the products of SVQ codewords, one from each of the L partitions. Let's denote the relation as $VQ(x_t) \equiv SVQ_1(x_{1t}) : SVQ_2(x_{2t}) : \cdots : SVQ_L(x_{Lt})$, where $VQ(\cdot)$ and $SVQ_i(\cdot)$ represent the VQ codeword given the full-space acoustic vector and the SVQ codeword in the ith partition given the acoustic subvector in that partition respectively. While an SVQ codeword (or bin) represents a multi-dimensional Voronoi cell in the space of one of the L partitions, a VQ codeword (or bin) constructed above is a d-dimensional convex polytope. Suppose the SVQ codebook of the ith partition consists of n_i SVQ codewords, and n_{max} is the maximum codebook size among the SVQ codebooks of the L partitions. Then there will be $\prod_{i=1}^{L} n_i$ VQ bins in the full acoustic space. For instance, if each d-dimensional acoustic vector is

[1] The technique is more commonly called "*split vector quantization*" [4] in the signal coding community. On the other hand, it has been called "*subvector quantization*" in various acoustic modeling methods such as the *discrete mixture hidden Markov modeling* [5, 6]. We decide to use the latter name.

partitioned into 6 subvectors (i.e., $L = 6$), and each partition is vector-quantized using 3 bits (i.e., there are $2^3 = 8$ SVQ bins for each partition), then there will be $8^6 = 262,144$ VQ bins in the full-space.

Although SVQ is employed, it is only used to efficiently index a VQ codeword in the original d-dimensional acoustic space through the combinatorial effect of per-partition SVQ codewords. Finally, an SVQ-HDDM is a *single* discrete density function indexed by the VQ codewords constructed from SVQ codewords in the way described above.

2.1. Conversion of a GMM to an SVQ-HDDM

Let the probability density function (pdf) of a continuous-density GMM with diagonal covariances be

$$p(\mathbf{x}_t) = \sum_{m=1}^{M} \left(c_m \mathcal{N}(\mathbf{x}_t; \mu_m, \sigma_m^2) \right) , \quad (2)$$

where M is the number of mixture components, and c_m, μ_m, and σ_m^2 are the mixture weight, mean vector, and variance vector of the mth Gaussian component. Since each Gaussian component is assumed to have diagonal covariance, it can be rewritten as the product of any number of subvector Gaussians. Thus, if we partition the acoustic vectors as in Eqn.(1), then we have

$$p(\mathbf{x}_t) = \sum_{m=1}^{M} \left(c_m \prod_{i=1}^{L} \mathcal{N}(\mathbf{x}_{it}; \mu_{im}, \sigma_{im}^2) \right) , \quad (3)$$

where x_{it}, μ_{im} and σ_{im}^2 are the ith subvector of x_t, μ_m and σ_m^2 respectively.

The conversion of a GMM pdf to a probability mass function (pmf) of the corresponding SVQ-HDDM consists in finding the probability of each VQ bin. The computation boils down to integrating a GMM pdf over the convex polytope representing each VQ codeword. If the VQ codeword h is constructed from L SVQ codewords $h_i, i = 1, 2, \ldots, L$, then we have $h \equiv h_1 : h_2 : \cdots : h_L$. Let's also denote the convex polytope of h by $\Omega(VQ_h)$ and the convex polytope of h_i by $\Omega(SVQ_{h_i})$, $i = 1, \ldots, L$. Hence, the probability of the VQ codeword h is given by

$$\begin{aligned} & P(\mathbf{x}_t \in \Omega(VQ_h)) \\ = & \sum_{m=1}^{M} \left(c_m \int_{\Omega(VQ_h)} \mathcal{N}(\mathbf{x}_t; \mu_m, \sigma_m^2) \right) \\ = & \sum_{m=1}^{M} \left(c_m \prod_{i=1}^{L} \int_{\Omega(SVQ_{h_i})} \mathcal{N}(\mathbf{x}_{it}; \mu_{im}, \sigma_{im}^2) \right) . \quad (4) \end{aligned}$$

There is no closed-form solution for the integrals over the convex polytope of each SVQ codeword $\Omega(SVQ_{h_i})$. In this paper, Matlab was used to compute the integrals. Although Matlab provides a few numerical routines that evaluate double or triple integration, we developed our own integration program using Matlab's single integration routines, *quad* and *quadgk*, to evaluate integrals of any order. *quad* is fast but imprecise, whereas *quadgk* can be very precise but runs too slow. Our new program tries to obtain a good trade-off between their integration speed and precision. Essentially, we only used *quadgk* for the integration of the innermost dimension, and *quad* for the integration of the remaining dimensions.

2.2. Time Complexity

The time complexity of finding a VQ codeword from its constituent SVQ codewords is $O(n_{max}L)$, and that of finding an SVQ-HDDM probability is $O(1)$ as it is simply a table lookup. As a consequence, the cost of SVQ-HDDM likelihood computation is largely determined by the search of codewords. Such search can be very fast provided the maximum SVQ codebook size n_{max} is sufficiently small.

Scalar-quantized HDDM (SQ-HDDM) and traditional VQ-based discrete model (VQDM) are special cases of SVQ-HDDM. An SQ-HDDM is an SVQ-HDDM in which each partition consists of a single dimension (i.e., $L = d$), so that SVQ in SVQ-HDDM is reduced to SQ in SQ-HDDM. On the other hand, a VQDM is equivalent to an SVQ-HDDM with only one partition (i.e., $L = 1$). Given a fixed number of bits, a VQDM is more accurate than an SQ-HDDM because VQ yields smaller quantization error than SQ. However, an SQ-HDDM finds codewords much faster than a VQDM since the SQ codebook size is much smaller. By carefully choosing the number of partitions, SVQ-HDDM may obtain a good trade-off between the accuracy of VQDM and the speed of SQ-HDDM.

2.3. Practical Issues

We discuss some practical issues of SVQ-HDDM below.

- *Partition of a feature vector into subvectors*
  In theory, we may partition the feature vectors in any manner. For simplicity, we partition the vector into subvectors of consecutive dimensions.
- *Subvector dimension, w*
  There is a trade-off on the subvector dimension. Given a fixed number of bits, larger subvector dimension will minimize the quantization error at the expense of more storage space and slower search time for SVQ codewords. In addition, The computation of the integrals in Eqn.(4) grows non-linearly with the subvector dimension. Here, subvector dimensions are limited to 2 and 3. Furthermore, again for simplicity, subvectors of all different partitions have the same dimension. i.e., $wL = d$.
- *Multiple-stream SVQ-HDDM*
  Multiple-stream SVQ-HDDM is introduced to get an SVQ-HDDM of reasonable size and resolution. As will be seen later, 24-dimensional acoustic vectors consisting of 12 static and 12 dynamic MFCCs were used in our experiments. If each feature vector is partitioned into 12 2-dimensional subvectors which are subvector-quantized with 4 bits per partition (i.e., 16 SVQ bins per partition), then there will be $(2^4)^{12} = 256$ trillion VQ bins! However, if the feature vectors are split into two 12-dimensional independent streams, and each stream is modeled independently by an SVQ-HDDM again with a subvector dimension of two, then the number of VQ bins is greatly reduced to $2 \times (2^4)^6 = 32$ million.
  To obtain a K-stream SVQ-HDDM, a GMM is first approximated by a K-stream GMM before the conversion method described in Section 2.1 is applied. The major shortcoming is the loss of correlation among features across the streams. Here only two streams are used.
- *Model size*
  In general, the model size of SVQ-HDDM is larger than its parent GMM as it stores all discrete probability values. Table 1 shows the model sizes of SQ-HDDMs and SVQ-HDDMs of varied resolutions used in this paper.

Table 1: 2-stream HDDMs of varied resolutions. (w = subvector dimension. Probabilities are assumed 4-byte floating point numbers. The model size is in megabytes (MB).)

Model	w	Bit Allocation/Partition	Size
SQ-HDDM-a	1	(222222221111, 111111111111)	4
SQ-HDDM-b	1	(222222221111, 222222221111)	8
SQ-HDDM-c	1	(222222222111, 222222222111)	16
SQ-HDDM-d	1	(222222222211, 222222222211)	32
SQ-HDDM-e	1	(222222222221, 222222222221)	64
SQ-HDDM-f	1	(222222222222, 222222222222)	128
SVQ-HDDM-a	2	(444422, 222222)	4
SVQ-HDDM-b	2	(444422, 444422)	8
SVQ-HDDM-c	2	(444432, 444432)	16
SVQ-HDDM-d	2	(444442, 444442)	32
SVQ-HDDM-e	2	(444443, 444443)	64
SVQ-HDDM-f	2	(444444, 444444)	128
SVQ-HDDM-g	3	(6666, 6666)	128

3. Experimental Evaluation

NIST SRE 2001 and 2002 were used in this work. Specifically, all of the 1006 male utterances (from $>$ 120 speakers) in NIST 2001 were used for creating the GMM-UBM, and speaker models were created by MAP adaptation [2] on the GMM-UBM for each of the 139 male speakers in NIST 2002. Each adaptation utterance is about 2 minutes long but half of the contents is silence. Each of the 1442 male verification utterances in NIST 2002, with an average duration of 16.3s, was scored against 11 hypothesized speakers. This amounts to 1,232 speaker trials and 14,630 impostor attempts.

The features used were 12 MFCCs plus their first derivatives, leading to 24-dimensional acoustic vectors. Cepstral mean normalization was applied to the MFCCs, followed by feature warping. Speaker verification is based on the log-likelihood ratio between a speaker model and the UBM.

Table 2: EER performance of 1-stream or 2-stream GMMs of varied number of mixtures, M. (K = #streams; all/top5 = GMM likelihood computation using all/top 5 Gaussians components.)

M	K = 1, all	K = 2, all	K = 1, top5
1024	11.11	**11.68**	11.84
512	**10.66**	12.08	**11.43**
256	11.45	12.32	11.88
128	11.72	12.50	12.37
64	12.52	13.30	13.20

3.1. SV Performance of Baseline GMMs

1-stream GMMs with 64–1024 Gaussian components were trained and converted to 2-stream GMMs; their SV performance is shown in Table 2. The GMM likelihoods were computed using either only the top-5 Gaussian components — which is the standard speedup method employed by the SV community — or all Gaussian components.

The effect of loss of correlation in multi-stream modeling can readily be seen in the performance of 1-stream GMMs and their 2-stream counterparts: as expected, 1-stream GMMs have better EER performance than their 2-stream counterparts (10.66% vs. 11.68%). The use of the top-5 Gaussian components in GMM likelihood computation actually results in significant performance degradation (10.66% vs. 11.43%). But since it speeds up the calculation by almost 50% and is the common practice, we continued to use it for all SV experiments using GMMs. On the other hand, that the performance degradation due to the heuristic use of top-5 Gaussians in GMM likelihood computation is comparable to that due to the use of two independent feature streams (11.68% vs. 11.43%) can be a good news for the proposed high-density discrete modeling (HDDM) because HDDM *always* uses *all* Gaussian components in a GMM.

Table 3: SV performance of various SQ-HDDMs and SVQ-HDDMs converted from a 2-stream GMMs with 1024 components.

Model Index	SQ-HDDM		SVQ-HDDM	
	EER	minDCF	EER	minDCF
a	**12.97**	**0.0671**	12.60	0.0661
b	12.93	0.0662	12.43	0.0662
c	12.47	0.0654	12.20	0.0651
d	12.39	0.0649	12.14	0.0649
e	12.16	0.0644	11.79	0.0640
f	12.11	0.0635	**11.60**	**0.0631**
g	—	—	**11.48**	**0.0618**

3.2. SQ-HDDM vs. SVQ-HDDM

For a given number of bits, the model size of an HDDM is the same regardless of the model order of the GMM they are converted from, the number of partitions, and the subvector dimension. Thus, one is free to choose the best GMM to convert to HDDM. Based on the results shown in Table 2, the 2-stream GMM with 1024 components has the lowest EER, and thus was chosen and converted to both SQ-HDDMs and SVQ-HDDMs of varied configurations indexed from 'a' to 'g' as shown in Table 1.

Table 3 summarizes the SV performance of the various SQ-HDDMs and SVQ-HDDMs. Among all the HDDMs, SQ-HDDM-a, having an EER of 12.97% was reported in [3], will serve as our HDDM baseline for the ensuing discussion. The results clearly show that

- SV performance improves with increasing codebook size (or equivalently, the model size of HDDMs, or the number of bits).
- For the same model size, an SVQ-HDDM performs better than an SQ-HDDM.
- SVQ-HDDMs with fewer partitions (or equivalently, larger subvector dimension) are preferred.
- Here we used at most 24 bits for each stream to limit the size of the generated HDDMs. Within the limit, none of the generated SQ-HDDMs perform as well as the parent 2-stream GMM from which they were converted. On the contrary, SVQ-HDDM-f and SVQ-HDDM-g both perform better than their parent 2-stream GMM (11.60% and 11.48% vs. 11.68%). This shows the superiority of SVQ over SQ in reducing the quantization error.
- The performance of SVQ-HDDM-g actually matches that of the parent 1-stream GMM if the top-5 Gaussian speedup heuristic is used.

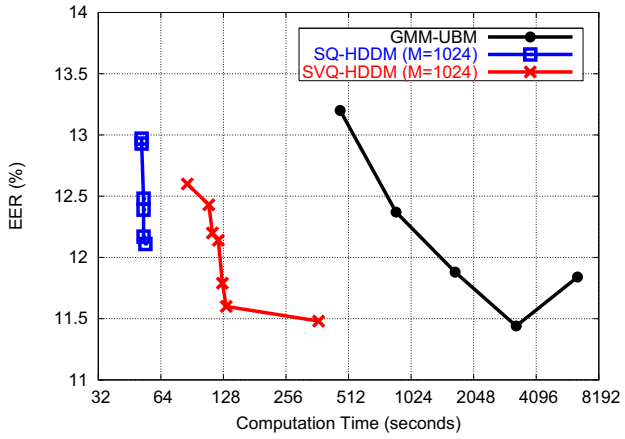

Figure 1: SV operating characteristics of various models when the UBM and speaker models are pre-loaded to the memory.

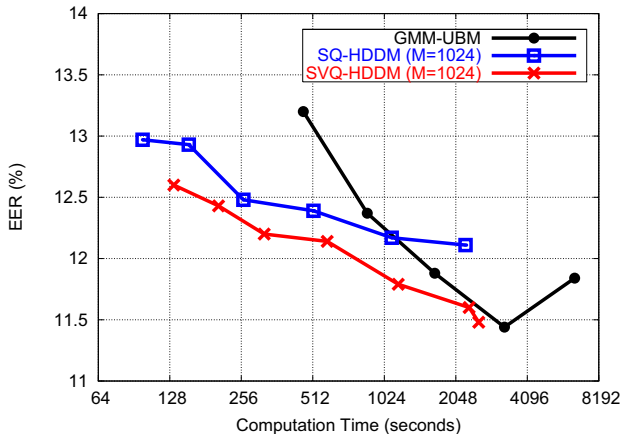

Figure 2: SV operating characteristics of various models when the UBM and speaker models are loaded on-the-fly.

3.3. Operating Characteristics of Various Models

There can be two operation modes for an SV application depending on whether the models are pre-loaded in memory. Model pre-loading is preferred if there is sufficient memory or when the number of speaker models is small. Thus, we measured[2] and reported the operating characteristics of various GMMs, SQ-HDDMs, and SVQ-HDDMs on the NIST 2002 SV task in two figures, depending on whether models were pre-loaded in Fig. 1 and Fig. 2. In both cases, the performance of the following models are compared:

- GMMs with varied number of Gaussian components.
- SQ-HDDMs and SVQ-HDDMs of varied resolutions (indexed from 'a' to 'f') as shown in Table 1 and they were converted from the GMMs with 1024 Gaussian components.

From Fig. 1, we find that if all models are pre-loaded onto the memory, for the same EER

- SQ-HDDM is faster SVQ-HDDM which is in turn faster than GMM. Since the speed of finding the HDDM likelihood is the same for both SQ-HDDM and SVQ-HDDM which is simply a table-lookup, SQ-HDDM is faster because the search for SQ codewords is faster. However, SQ-HDDM cannot achieve the best performance of its parent 1-stream or 2-stream GMM.
- On the other hand, SVQ-HDDM can achieve almost the best performance of its parent 1-stream GMM at a much faster speed.
- In general, for the same EER, SVQ-HDDM is 8-25 times faster than GMM.

On the other hand, if speaker models are loaded on-the-fly, Fig. 2 shows that the computational advantage of HDDM (both SQ-HDDM and SVQ-HDDM) over GMM is largely offset by the memory loading time of its much bigger model. However, the memory loading time is affected greatly by the hardware. We believe that using better server-grade hardware may improve the operating characteristics of HDDMs in Fig. 2.

4. Conclusions

Last year, we proposed to speed up GMM computations in speaker verification tasks by using scalar quantization (SQ) in the construction of high-density discrete models (HDDM) from the parent GMM. In this paper, we generalize the quantization scheme to subvector quantization (SVQ). Empirical results from NIST 2002 SV evaluation show that for the same model size, SVQ produces HDDMs that are more accurate than SQ. SVQ-HDDM can also achieve the same performance of its parent GMM when sufficient number of bits is employed while SQ-HDDMs cannot; yet, for the same EER performance, SVQ-HDDMs are only 2 times slower than SQ-HDDMs . In summary, compared with its parent GMM, SVQ-HDDM may provide a significantly speedup of 8–25 times when the speaker models are pre-loaded in memory.

5. References

[1] J. McLaughlin, D. A. Reynolds, and T. Gleason, "A study of computation speed-ups of the GMM-UBM speaker recognition system," in *Proc. of Interspeech*, 1999, pp. 1215–1218.

[2] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Processing*, vol. 10, pp. 19–41, 2000.

[3] G. Ye, B. Mak, and M. W. Mak, "Fast GMM computation for speaker verification using scalar quantization and discrete densities," in *Proc. of Interspeech*, 2009.

[4] K. K. Paliwal and B. S. Atal, "Efficient vector quantization of LPC parameters at 24 bits/frame," *IEEE Trans. on SAP*, vol. 1, no. 1, pp. 3–14, January 1993.

[5] S. Takahashi, K. Aikawa, and S. Sagayama, "Discrete mixture HMM," in *Proc. of ICASSP*, 1997, pp. 971–974.

[6] V. Digalakis, S. Tsakalidis, C. Harizakis, and L. Neumeyer, "Efficient speech recognition using subvector quantization and discrete-mixture HMMs," *Computer Speech and Language*, vol. 14, pp. 33–46, 2000.

[7] E. Bocchieri and B. Mak, "Subspace distribution clustering hidden Markov model," *IEEE Trans. on SAP*, vol. 9, no. 3, pp. 264–275, March 2001.

[2] All experiments were performed on a Linux machine that runs on the Intel CPU, Core 2 Duo E8400 @ 3.00GHz with 4GB RAM. The computation time was collected over *all* 1442×11 trials.