Subvector-quantized High-density Discrete Hidden Markov Model and its Re-estimation

Guoli YE and Brian MAK
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
Email: {yeguoli, mak}@cse.ust.hk

***Abstract*—We investigated two methods to improve the performance of high-density discrete hidden Markov model (HDDHMM). HDDHMM employs discrete densities with a very large codebook consisting of thousands to tens of thousands of vector quantization (VQ) codewords which are constructed as the product of per-dimension scalar quantization (SQ) codewords. Although the subsequent HDDHMM is fast in decoding, it is not accurate enough. In this paper, making use of the fact that, for a fixed number of bits, VQ is more efficient than SQ, subvector quantization (SVQ) was investigated to improve the quantization efficiency while keeping the (time and space) complexity of the quantizer sufficiently low. Model parameters of the resulting SVQ-HDDHMM were further re-estimated. For the Wall Street Journal 5K-vocabulary task, it is found that the proposed SVQ-HDDHMM could be a better model (both in terms of recognition time and error rate) than conventional continuous-density HMM for practical deployment.**

I. Introduction

Continuous-density hidden Markov model (CDHMM) using Gaussian mixture model (GMM) states is the dominant acoustic model employed in most state-of-the-art automatic speech recognition (ASR) systems. Nevertheless, with the advance in semiconductor technologies (e.g., large solid-state RAM space is becoming affordable), one may ponder: "*How could we improve the acoustic models in ASR if we were given very large amount of memory?*" It prompted a re-visit of the use of discrete density in HMM as in the work of *discrete mixture HMM* (DMHMM) [1], [2] and *high-density discrete HMM* (HDDHMM) [3].

Compared with CDHMM, the use of discrete density in HMM has the following attractive properties:

- the state distribution is non-parametric: in theory, it can model any distribution if there are enough training data.
- it is fast in decoding: to find the state probability is simply a table lookup.

However, even given sufficient memory storage, traditional discrete HMM (DHMM) could not afford a large codebook size. The major reason is the lack of training data, which is usually not sufficient to train the discrete density of a large codebook. As a result, usually 256 to 1024 codewords are used. Such a small codebook will induce larger quantization error that is unacceptable for complicated tasks which require high accuracy.

Unlike traditional DHMM which usually has at most 1024 codewords, HDDHMM has a codebook size in the order of thousands to tens of thousands. It takes only $O(d)$ time to find the codeword for a new acoustic vector in HDDHMM, where d is the dimension of the full-space acoustic vectors. This is made possible by constructing full-space vector quantization (VQ) codewords as the product of per-dimension scalar quantization (SQ) codewords. Moreover, the state likelihood computation is reduced to a simple table-lookup in HDDHMM. Thus, what is equivalent to the evaluation of GMM likelihood at each CDHMM state almost takes no time in HDDHMM. However, the computational advantage of HDDHMM is partly offset by three shortcomings:

1) For the conventional 39-dimensional MFCC acoustic vector (comprised of the static, delta, and delta delta MFCCs), even a 1-bit SQ for each dimension will lead to a full-space VQ codebook size of $2^{39} = 549,755,813,888$! Thus, it is necessary to split an acoustic vector into multiple independent streams for modeling. The use of multiple independent streams cannot model correlations among features across streams and results in poorer recognition performance.
2) As the computer memory is not really infinite in reality, there is always a limit to the number of quantization bits, which, in turn, limits the resolution of the HDDHMM and its performance.
3) Given the huge parameter set of an HDDHMM, it is our estimate that even an HDDHMM of modest size may require several thousands of hours of speech for its training. In the past, we avoided directly training an HDDHMM and proposed a conversion method to convert a K-stream CDHMM to a K-stream HDDHMM. However, our experience [3] on the Wall Street Journal 5K-vocabulary task shows that the conversion of a 4-stream CDHMM system to a 4-stream HDDHMM system resulted in significant increase of an absolute 2% in the word error rate (WER).

Although we cannot do much with the first shortcoming, we proposed to tackle the last two shortcomings as follows.

1) To generalize the quantization scheme in the genera-

This work was supported by the Research Grants Council of the Hong Kong SAR under the grant number DAG05/06.EG43, HKUST617507, and HKUST617008.

tion of HDDHMM from SQ to *subvector quantization* (SVQ). It is well-known that for a fixed number of bits, VQ is more efficient than SQ (in the sense that it results in smaller quantization error) but at the price of higher time and space complexities. SVQ may be used to strike a balance between efficiency and complexity, and has been used successfully in the work of LPC coding [4], SDCHMM [5], and DMHMM [2].
To emphasize the difference in the quantization scheme, HDDHMM that employs SVQ will be called *SVQ-HDDHMM* and our previous HDDHMM [3] which employs SQ will be renamed as *SQ-HDDHMM*.
2) To investigate an indirect training method for SVQ-HDDHMM via an intermediate SVQ-DMHMM[1].

II. Subvector-quantized High-density HMM (SVQ-HDDHMM)

Before the construction of an SVQ-HDDHMM, each d-dimensional acoustic vector $\mathbf{x}_t$ is partitioned into L subvectors, $\mathbf{x}_t = [\mathbf{x}_{1t}, \mathbf{x}_{2t}, \ldots, \mathbf{x}_{Lt}]$. Subvectors of each partition are 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

$$VQ(\mathbf{x}_t) \equiv SVQ_1(\mathbf{x}_{1t}) : SVQ_2(\mathbf{x}_{2t}) : \cdots : SVQ_L(\mathbf{x}_{Lt}),$$

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 L SVQ codebooks. Then there will be $\prod_{i=1}^{L} n_i$ VQ bins in the full acoustic space. For instance, if each acoustic vector is partitioned into 3 subvectors (i.e., $L = 3$) and each partition is vector-quantized using 4 bits, then there will be $(2^4)^3 = 4,096$ 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, each stream of an SVQ-HDDHMM state is a *single* discrete density function indexed by the products of SVQ codewords in the way described above.

A. Finding a VQ Codeword

Given a new acoustic vector, it will first be split into L subvectors. The subvector in the ith partition is then compared with the corresponding SVQ codebook to find its SVQ codeword in $O(n_i)$ time. Thus, for the same cookbook size, SVQ-HDDHMM takes $\sum_{i=1}^{L} n_i$ comparisons to find a VQ codeword, which is significantly smaller than the $\prod_{i=1}^{L} n_i$ comparisons in conventional DHMM.

[1]The development of DMHMM started with the use of SQ [1] and was later improved with the use of SVQ [2]. We will also call the two kinds of DMHMM as *SQ-DMHMM* and *SVQ-DMHMM* respectively.

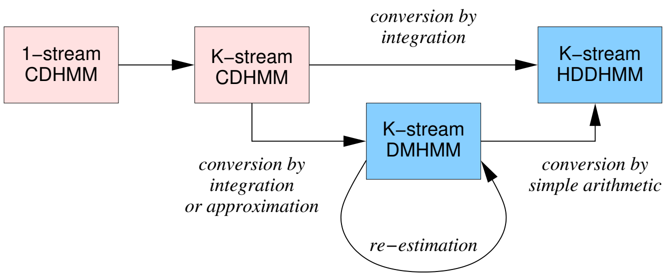
1–stream CDHMM
K–stream CDHMM
conversion by integration
K–stream HDDHMM
K–stream DMHMM
conversion by integration or approximation
conversion by simple arithmetic
re–estimation

Fig. 1. Two indirect methods for the parameter estimation of SVQ-HDDHMM

B. Parameter Estimation of SVQ-HDDHMM

Since SVQ-HDDHMM may have thousands to tens of thousands of full-space VQ codewords per state, it is a great challenge to directly estimate its parameters. Fig.1 provides a schematic view of two methods for the indirect estimation of SVQ-HDDHMM parameters.

1) Method I: Conversion from a GMM: The simple conversion algorithm used in the construction of SQ-HDDHMM [3] is modified to construct SVQ-HDDHMM in three steps as shown on the upper path in Fig.1.

STEP 1: A conventional 1-stream CDHMM with diagonal-covariance GMM states is first trained.

STEP 2: It is converted to a K-stream CDHMM simply by splitting the diagonal-covariance Gaussians according to the stream definition. The K-stream CDHMM is then re-estimated using the Baum-Welch algorithm. The probability density function (pdf) of the kth stream of state j is given by

$$b_j^{(k)}(\mathbf{x}_t^{(k)}) = \sum_{m=1}^{M} c_{jm}^{(k)} N(\mathbf{x}_t^{(k)}; \mu_{jm}^{(k)}, \sigma_{jm}^{2^{(k)}}) \ ,$$

where $\mathbf{x}_t^{(k)}$ is the acoustic vector of the kth stream; $\mu_{jm}^{(k)}$, $\sigma_{jm}^{2^{(k)}}$, and $c_{jm}^{(k)}$ are the mean vector, variance vector, and mixture weight of the mth Gaussian component in the kth stream of state j; M is the number of mixture components in that state.

STEP 3: Each GMM pdf is converted to a probability mass function (pmf) of the corresponding SVQ-HDDHMM by finding the probability of each VQ bin. The computation boils down to integrating a GMM pdf over the convex polytope representing each VQ codeword. Suppose the VQ codeword of $\mathbf{x}_t^{(k)}$ in the kth stream is $h^{(k)}$, which is constructed from $L^{(k)}$ SVQ codewords $h_i^{(k)}, i = 1, 2, \ldots, L^{(k)}$, then we have $h^{(k)} \equiv h_1^{(k)} : h_2^{(k)} : \cdots : h_{L^{(k)}}^{(k)}$. Let's also denote the convex polytope of $h^{(k)}$ by $\Omega(VQ_{h^{(k)}})$ and the convex polytope of $h_i^{(k)}$ by $\Omega(SVQ_{h_i^{(k)}})$. Hence, the probability of the VQ codeword of $\mathbf{x}_t$ in state j is given by

$$\begin{aligned}
& P_j(\mathbf{x}_t \in \Omega(VQ(\mathbf{x}_t))) \\
= & \prod_{k=1}^{K} P_j^{(k)}(\mathbf{x}_t^{(k)} \in \Omega(VQ_{h^{(k)}})) \\
= & \prod_{k=1}^{K} \left[\sum_{m=1}^{M} \left(c_{jm}^{(k)} \int_{\Omega(VQ_{h^{(k)}})} \mathcal{N}(\mathbf{x}_t^{(k)}; \mu_{jm}^{(k)}, \sigma_{jm}^{2(k)}) \right) \right] \\
= & \prod_{k=1}^{K} \left[\sum_{m=1}^{M} \left(c_{jm}^{(k)} \prod_{i=1}^{L} P_{jmi}^{(k)}(\mathbf{x}_{it}^{(k)}) \right) \right] , \qquad (1)
\end{aligned}$$

where $\mathbf{x}_{it}^{(k)}$, $\mu_{jmi}^{(k)}$, and $\sigma_{jmi}^{2(k)}$ are the subvector of $\mathbf{x}_t^{(k)}$ $\mu_{jm}^{(k)}$, and $\sigma_{jm}^{2(k)}$ in the ith partition of the kth stream, and

$$P_{jmi}^{(k)}(\mathbf{x}_{it}^{(k)}) = \int_{\Omega(SVQ_{h_i^{(k)}})} \mathcal{N}(\mathbf{x}_{it}^{(k)}; \mu_{jmi}^{(k)}, \sigma_{jmi}^{2(k)}) \ . \qquad (2)$$

There is no closed-form solution for the integrals in Eqn.(2). We developed our own numerical integration program to evaluate the integrals using the single integration routines, *quad* and *quadgk*, provided by Matlab.

2) Method II: Re-estimation via a DMHMM: (A) SVQ-DMHMM as an SVQ-HDDHMM

Eqn.(1) shows that a K-stream SVQ-DMHMM can be treated as a K-stream SVQ-HDDHMM[2], and can always be converted to the latter by simple arithmetic. SVQ-HDDHMM is more efficient in state likelihood computation while SVQ-DMHMM has fewer model parameters. To see that, for a stream with L subvector partitions, if each partition is vector-quantized to n_i bins, the number of parameters in an SVQ-HDDHMM state is $\prod_{i=1}^{L} n_i$; the similar figure for an SVQ-DMHMM state is $M \times (1 + \sum_{i=1}^{L} n_i)$. Thus, the number of model parameters in an SVQ-DMHMM is greatly reduced, which makes the training of SVQ-DMHMM feasible.

(B) Model training

Following the lower training path in Fig.1, the training procedure of a K-stream SVQ-HDDHMM via a K-stream SVQ-DMHMM is described as follows.

STEP 1 and 2: Same as those in the conversion method in Section 2.2.1.

STEP 3: Convert the K-stream CDHMM to an initial K-stream SVQ-DMHMM using methods discussed in (C).

STEP 4: Re-estimate the model parameters of the K-stream SVQ-DMHMM as described in [2].

STEP 5: Convert the re-estimated K-stream SVQ-DMHMM to an equivalent K-stream SVQ-HDDHMM using Eqn.(1).

(C) Initialization of SVQ-DMHMM

We investigated two ways to compute the quantity, $P_{jmi}^{(k)}(\mathbf{x}_{it}^{(k)})$ of Eqn.(2), for the initial K-stream SVQ-DMHMM:

- by integration as shown in Eqn.(2).
- by centroid approximation as used in [2]:

$$P_{jmi}^{(k)}(SVQ(\mathbf{x}_{it}^{(k)})) = \frac{\mathcal{N}(h_i^{(k)}; \mu_{jmi}^{(k)}, \sigma_{jmi}^{2(k)})}{\sum_{c_i^{(k)}} \mathcal{N}(c_i^{(k)}; \mu_{jmi}^{(k)}, \sigma_{jmi}^{2(k)})} \qquad (3)$$

  where $c_i^{(k)}$ is any codeword in the ith subvector partition of the kth stream.

In general, model initialization by integration can be quite accurate but is computationally very expensive especially when the subvector dimension is high. On the contrary, initializing the model by the centroid approximation method is very fast though the resulting model is not accurate. Insofar as the initialization is followed by re-estimation iterations, the second method may be used when the subvector dimension is high.

(D) Model smoothing

Even though SVQ-DMHMM already has many fewer model parameters than its SVQ-HDDHMM counterpart, its number of model parameters is usually still much larger than the conventional CDHMM. Consequently, its model parameters may not be robustly trained. Here, the simple smoothing technique in [2] was adopted which interpolates the newly estimated value of a parameter with its old value in the last iteration. That is, if $P_{jmi,old}^{(k)}(\cdot)$ and $P_{jmi,new}^{(k)}(\cdot)$ are the discrete density of the ith subvector partition of the mth mixture in state j of the kth stream in the previous and current re-estimation iterations respectively, then the smoothed density, $P_{jmi}^{(k)}(\cdot)$, is computed as

$$P_{jmi}^{(k)}(\cdot) = \alpha P_{jmi,new}^{(k)}(\cdot) + (1-\alpha) P_{jmi,old}^{(k)}(\cdot) \, ,$$

where $\alpha \in [0.0, 1.0]$ is a smoothing coefficient determined empirically using some development data.

III. Experimental Evaluation

The proposed SVQ-HDDHMM was evaluated on the Wall Street Journal 5K-vocabulary task (WSJ0). All experiments were run on a Linux machine that runs on the Intel CPU, Core 2 Duo E8400@3.00GHz with 4GB RAM.

The conventional 39-dimensional MFCC vectors were extracted at every 10 ms over a window of 25 ms. The parent CDHMM system consisted of 15,449 cross-word triphones and 3,130 tied states (plus 2 untied state). Evaluation was performed on the standard nov'92 5K non-verbalized test set which consists of 330 utterances from 8 speakers. A trigram language model with a perplexity of 57 was used in decoding. A subset of the development data set of WSJ0 was used to tune the decoding parameters and the optimal smoothing coefficient α, as well as to decide when to stop HMM training. Finally, the HTK software was modified for HDDHMM training and decoding.

A. Stream Definition, Bit Allocation Schemes, and Subvector Partitions

4-stream SVQ-DMHMMs and 4-stream SVQ-HDDHMMs were constructed according to the stream definition depicted

[2]In general, the converse is *not* true: a K-stream SVQ-HDDHMM may not be converted to an equivalent K-stream SVQ-DMHMM.

TABLE I
DEFINITION OF THE 4 STREAMS AND BIT ALLOCATION SCHEMES.

Bit Alloc	Stream 1: 12 MFCCs	Stream 2: 12 ΔMFCCs	Stream 3: 12 $\Delta\Delta$MFCCs	Stream 4: 3 energies
b1	222211111111	222211111111	222211111111	555
b2	222121111111	222121111111	222121111111	555
b3	221122111111	221122111111	221122111111	555
b4	221121112111	221121112111	221121112111	555

TABLE II
BASELINE CDHMM PERFORMANCE ON WSJ0

Model	WER on Test Data
CDHMM-1stream	4.46
CDHMM-4stream	5.23

in Table I. The table also shows four different schemes of bit allocation per dimension investigated in this paper; all of them have the same total numbers of bits for each stream, which are 16, 16, 16, and 15 respectively. Basically, we tried to put more bits to the MFCCs of lower indices, which makes use of the fact that MFCCs of lower indices are more important for recognition task. In addition, the following two subvector partitions (or dimensions w) were attempted:

- $w = 3$: {3333, 3333, 3333, 111}.
- $w = 4$: {444, 444, 444, 111}.

For example, in the case of $w = 3$, each of the first three streams was split into four 3-dimensional partitions, while the last stream was split into three 1-dimensional partitions.

B. Baseline CDHMM Performance

There are two baseline results in Table II: one from the 1-stream CDHMM system and the other from the 4-stream CDHMM system. As expected, there is a small but significant degradation in the recognition performance with the use of multiple independent streams, probably due to the fact that the multi-stream CDHMM cannot model the correlation among features from different streams.

C. Comparison of Different Quantization Methods

From Table III, we observe that

- For a fixed number of bits, the construction of HDDHMM using SVQ is more effective than if SQ is used.
- The conversion of a 4-stream CDHMM to a 4-stream SVQ-HDDHMM using numerical integration is effective. The converted 4-stream SVQ-HDDHMM has very similar WER as its parent 4-stream CDHMM (5.64% vs. 5.23%).
- The proposed re-estimation method via an intermediate SVQ-DMHMM works well and further improves the model's WER from 5.64% to 5.45%, which is now very close to the WER of 4-stream CDHMM baseline performance 5.23%.
- If possible, it is better to initialize an SVQ-DMHMM by integration instead of using centroid approximation. Fortunately, after model re-estimation, SVQ-HDDHMM initialized by the centroid approximation method will improve. This can be useful if we larger subvector dimensions are used.

TABLE III
COMPARISON BETWEEN SQ AND SVQ USING BIT ALLOCATION SCHEME B1 AND SUBVECTOR DIMENSION $w = 3$.

SQ / SVQ	Initialization	WER (Re-estimation?)	
		no	yes
SQ	integration	6.26	6.26
SVQ	integration	5.64	**5.45**
SVQ	centroid approx.	6.67	6.11

TABLE IV
COMPARISON OF DIFFERENT BIT ALLOCATION WITH SUBVECTOR DIMENSION $w = 4$.

Bit Allocation	WER on Dev	WER on Test
b1	8.37	6.31
b2	**7.97**	**6.00**
b3	8.25	6.54
b4	8.37	6.16

D. Effect of Bit Allocations and Subvector Dimension

The subvector dimension was increased to 4, and different bit allocations given in Table I were tried with no re-estimation after model conversion. Their effects are recorded in Table IV. In this case, since the subvector dimension is too large for numerical integration of the convex polytopes, models were initialized by the centroid approximation method. It is found that the bit allocation scheme, b2, gives the best performance.

E. Best Model

Based on the results of Table IV, the 4-stream SVQ-DMHMM constructed using the bit allocation scheme b2 and subvector dimension of $w = 4$ was re-estimated and then converted to a 4-stream SVQ-HDDHMM. The final model gives the best WAC performance of 5.31% which compares favorably with the 4-stream CDHMM baseline's WER of 5.23%.

F. Operating Characteristics

The operating characteristics of the following five models are compared in Fig. 2.

- 1-stream CDHMM baseline (model size: 15MB)
- 4-stream CDHMM baseline (model size: 16MB)
- conventional 4-stream DHMM with 512 codewords for each MFCC stream and 256 codewords for the energy stream (model size: 11MB)
- 4-stream SQ-HDDHMM using bit allocation scheme b1 (model size: 1370MB)
- 4-stream SVQ-HDDHMM using bit allocation scheme b2 and subvector dimension $w = 4$ (model size: 1370MB)

The results show that the best 4-stream SVQ-HDDHMM performs better than its parent 4-stream CDHMM, conventional discrete HMM and the best 4-stream SQ-HDDHMM. Even compared with the 1-stream CDHMM baseline, it provides better performance in operating conditions when computation time is of the greatest concern.

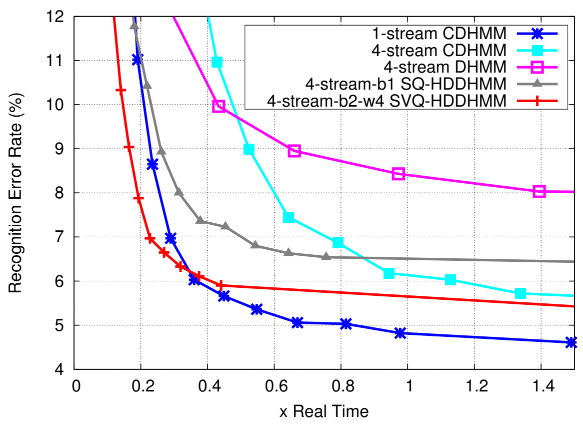

Fig. 2. The operating characteristics of various models

IV. Conclusions

Two methods are proposed to improve HDDHMM: (1) the use of subvector quantization (SVQ), and (2) parameter re-estimation via a discrete-mixture HMM. Evaluation on WSJ0 shows the effectiveness of both methods. Although the asymptotic accuracy of the new 4-stream SVQ-HDDHMM is still not as good as that of its parent 1-stream CDHMM, the performance gap is small. Furthermore, from the operating characteristics of the two models, SVQ-HDDHMM has better recognition performance under stringent computation time requirement.

References

[1] Satoshi Takahashi *et al.*, "Discrete mixture HMM," in *Proc. of ICASSP*, April 1997, vol. 2, pp. 971–974.

[2] 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.

[3] Brian Mak, S. K. Au Yeung, Y. P. Lai, and M. Siu, "High-density discrete HMM with the use of scalar quantization indexing," in *Proc. of Eurospeech*, Sept 2005.

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