QUANTIZING MIXTURE-WEIGHTS IN A TIED-MIXTURE HMM

Sunil K. Gupta, Frank Soong and Raziel Haimi-Cohen
Bell Laboratories, Lucent Technologies
700 Mountain Avenue, Murray Hill, NJ 07974

ABSTRACT

In this paper, we describe new techniques to significantly reduce computational, storage and memory access requirements of a tied-mixture HMM based speech recognition system. Although continuous mixture HMMs offer improved recognition performance, we show that tied-mixture HMMs may offer significant advantage in complexity reduction for low-cost implementations. In particular, we consider two tasks: (a) connected digit recognition in car noise; and (b) sub-word modeling for command word recognition in a noisy office environment. We show that quantization of mixture weights can provide an almost three fold reduction in mixture-weight storage requirements without any significant loss in recognition performance. Furthermore, we show that by combining mixture-weight quantization with techniques such as VQ-Assist the computational and memory access requirements can be reduced by almost 60-80% without any degradation in recognition performance.

1 INTRODUCTION

Recently, there has been increased interest in voice dialing using a portable/car phone in hands-free mode [1, 2, 3, 4]. The problem is quite difficult due to the very noisy conditions in a car. Furthermore, in order to meet the low-cost and low-power requirements of portable consumer products, a fixed-point implementation with low storage and low computational complexity is mandatory.

In a previous paper [1], we addressed the task of speaker-independent recognition in a car-noise environment using continuous-density hidden Markov models. Although continuous-mixture HMMs offer the highest recognition accuracy, they have relatively high computational cost. In this paper, we investigate the application of tied-mixture HMMs [5, 6] to achieve lower computational complexity. In a low-cost, fixed-point implementation on a Digital Signal Processor (DSP), the reference models are usually stored in slow off-chip memory. This results in a significant overhead in memory access during the computation of kernel/state likelihoods. Several approaches, such as VQ-Assist and beam search [7] have been successfully applied to reduce both the computational cost and the memory accesses for Gaussian kernel parameters by a factor of three to nine. In a tied-mixture HMM based speech recognition system, all states in all the models share the same set of Gaussian kernels. The state observation likelihood for state $j, 1 \leq j \leq N$ in log-domain is defined as

$$\log b_j(\vec{\mathbf{o}}_t) = \sum_{s=1}^{S} \log \left[\sum_{m=1}^{M_s} c_{jms} \cdot \mathcal{N}[\vec{\mathbf{o}}_{ts}; \vec{\boldsymbol{\mu}}_{ms}, \vec{\boldsymbol{\sigma}}^2_{ms}] \right], \quad (1)$$

where S is the number of streams into which the feature vector is divided, M_s is the number of mixture components for stream s, $\mathcal{N}[\vec{\mathbf{o}}_{ts}; \vec{\boldsymbol{\mu}}_{ms}, \vec{\boldsymbol{\sigma}}^2_{ms}]$ is the likelihood of Gaussian kernel m and stream s, and c_{jms} is the corresponding weight for state j. Since the total number of Gaussian kernels is considerably less compared to continuous mixture models, the computational cost for kernel likelihoods is reduced. Furthermore, for each additional HMM state, there is a significantly larger increase in the memory and computational requirements for continuous mixture HMMs than for tied-mixture HMMs.

We note that after the kernel likelihoods are computed, we need $M = \sum_{s=1}^{S} M_s$ multiplications, $M - 1$ additions, and S logarithmic operations for each state-likelihood computation. It is clear that the number of mixture component weights for each state in a tied-mixture HMM is quite high (544 in our example). In sections 3.2 and 3.3, we show methods to dramatically reduce the storage requirements of mixture weights and as a consequence significantly decrease the operations and storage access requirements for evaluation of state likelihoods.

We address these performance/complexity issues within the context of two recognition tasks: (a) connected-digit recognition in car-noise; and (b) command-word recognition based upon sub-word models.

In section 2, we describe our recognition tasks and the baseline performance. We describe methods to reduce the complexity of a tied-mixture HMM in section 3. In section 4, we conclude with a discussion of our results.

2 RECOGNITION TASKS

2.1 Connected-Digits

For training, we use TI-DIGITS to which car-noise is added [1]. The test database consists of 7 female and 8 male speakers. The subjects spoke digit strings of lengths 1, 3, 4, 7, and 10 in a noisy Chevrolet Celebrity car driven at a speed of 30 MPH. A first-order gradient (FOG) microphone was mounted on the visor at the driver side. The average string length of the database is 5.9 digits. The windows were kept closed during the data collection with the fan running on medium setting. Each 20-ms non-overlapping frame of 8-bit $\mu-$law samples is converted into 16-bit linear format, high-pass filtered (100 Hz - 300 Hz transition band) and 25 features (12 LPC-cepstral coefficients, 12

delta-cepstral coefficients and 1 delta-energy) are extracted. We use whole-word digit models with 8-states. The background model is a single-state model. We use gender dependent models, 2-level CMS, and constrained-length decoding [1] in our experiments. Table 1 compares the recognition performance on car-data between the continuous-density HMMs (8-Gaussian components per state) and tied-mixture HMMs.

HMM Type	Digit Acc. (%)	String Acc. (%)
Continuous-Density	96.1	83.4
Tied-Mixtures	95.5	81.0

Table 1. Recognition performance on digits database for continuous-density HMMs and tied-mixture HMMs.

2.2 Sub-word Models Based Recognition

For many applications, the vocabulary may need to be changed depending upon the current task. In order to avoid the need to collect a database for each new vocabulary word, whole word models can be obtained by concatenating a sequence of sub-words (or phonemes) based upon transcriptions obtained from a text-to-speech (TTS) system. For training the forty-one sub-word models and a background model in our system, we have used 12140 phrases collected over the telephone network. A test database of eight voice-dialing command words ("Redial Last Number", "Voice Name List", 'Cancel Command", "Operator", "Send Number", "Cancel", "Yes", and "No") was collected in an actual office environment using a separate handset. We obtained 431 utterances in the test database. Each context-independent sub-word unit is a 3-state HMM. The training and recognition setup was the same as for the connected digits with the following exceptions: consecutive 20-ms frames overlap by 10 ms, and gender-independent models are used.

3 COMPLEXITY REDUCTION

In an HMM based speech recognition system, a significant portion of the computing power available on a DSP chip may be needed to access the reference models due to slow external model memory. In this section, we investigate several techniques to reduce the total models storage as well as access requirements for tied-mixture HMMs based recognition system, and their impact on recognition performance. From Eq. (1), we note that there are two ways to achieve a reduction in complexity.

1 Reducing the number of mixture component weights.

2 Reducing the total number of kernels used in state likelihood computations.

We will call the first scheme "static" reduction in complexity, based upon the mixture weight values derived from the training data while the second scheme which we call "dynamic" reduction of complexity is based upon the likelihood values of the current input observation.

3.1 Kernel and State Likelihoods

In this section, we simply investigate the effect on performance when only the top few kernel likelihood values are used in computing the state observation likelihood. Assume, without loss of generality that the kernel likelihoods $L_1, L_2, \cdots, L_{M_s}$ are arranged in descending order. Then, we select M_s^L with $M_s^L < M_s$ and truncate the sum to include only the top M_s^L likelihood values. Table 2 shows the recognition performance as a function of M_s^L for connected digit task and command-word recognition task using sub-word models.

M_1^L, M_2^L, M_3^L	Connected Digits Digit Acc. (%)	Command Words Word Acc. (%)
Baseline	95.5	92.8
24, 24, 16	95.1	93.2
10, 10, 10	95.0	92.8

Table 2. Recognition performance on digits database as a function of the number of Gaussian kernel likelihoods included in the computation of the state observation likelihood.

Note that for an isolated word recognition task such as command-word recognition, very few mixture components are required to obtain performance close to baseline performance. For the more complex connected digit recognition task in a highly noisy environment, many more mixture components must be retained to maintain good recognition performance

The above approach requires the evaluation of all Gaussian kernel likelihoods in order to select the top M_s^L likelihoods. In a practical implementation, kernels may be grouped into several clusters, each represented by a centroid. Only those kernels for which the cluster centroid is "close" to the input feature vector [7] would be evaluated. This can provide significant savings in likelihood computations and in memory access cost for the models.

3.2 Reduction in Mixture Weights

In a tied-mixture hidden Markov model based system, a significant number of mixture component weights are close to zero. This may be the result of a kernel being "too far" from the feature space for a state or due to a lack of enough training data for the state. Let M_{js}^z represent the number of non-zero mixture component weights for stream s in state j. For connected digits and command word recognition, we find that 40-50% of the mixture component weights are zero. To store only non-zero mixture weights, we need to also store the index of the corresponding kernel, in addition to its value. Furthermore, for each stream and each state, the number of non-zero weights M_{js}^z needs to be stored. The overall storage requirement for mixture component weights, M_w for model w is:

$$M_w = \sum_{s=1}^{S}\sum_{j=1}^{N} M_{js}^z \text{ weights } + \left[\sum_{s=1}^{S}\sum_{j=1}^{N}(1 + M_{js}^z)\right] \text{ indices,} \quad (2)$$

where N is the number of states in the model. In the following sections, we describe several approaches to reducing M_w.

A simple approach to reducing memory requirements for mixture weights is to only use $M_{js}^{'}$, with $M_{js}^{'} \leq M_{js}^z$ mixture components. Since the kernel likelihoods are indexed, we may assume that the weights c_{jms} are arranged in decreasing order for each state. The weight c_{jms} in Eq. (1) is

replaced by $\hat{c}_{jms}$ such that

$$\hat{c}_{jms} = \begin{cases} c_{jms} / \sum_{m=1}^{M_{js}^{'}} c_{jms}, & m \leq M_{js}^{'} \\ 0, & m > M_{js}^{'}. \end{cases} \quad (3)$$

The storage required for mixture component weights is

$$M_w = \left[\sum_{s=1}^{S} \sum_{j=1}^{N} M_{js}^{'} \right] \text{ weights} + \left[\sum_{s=1}^{S} \sum_{j=1}^{N} (1 + M_{js}^{'}) \right] \text{ indices}, \quad (4)$$

Since $M_{js}^{'} \leq M_{js}^{z}$, the above scheme clearly reduces the memory requirements for mixture weights. However, as shown in table 3, there is a significant penalty in recognition performance when $M_{js}^{'}$ is reduced.

$M_{j1}^{'}, M_{j2}^{'}, M_{j3}^{'}$	Accuracy (%)
Baseline	92.8
100, 100, 16	91.6
60, 60, 16	82.1
40, 40, 16	76.1
24, 24, 16	62.2

Table 3. Recognition performance on command words database for different number of included mixture component weights.

Note that the dynamic range of kernel likelihood values is significantly larger compared to the dynamic range of mixture weights. Therefore, if the mixture weight for a Gaussian kernel with a high observation likelihood for an input feature vector is artificially set to zero, it can lead to severe error in state-likelihood estimation.

In an alternative approach, different number of mixture components are selected in the state-likelihood computation such that the sum of mixture weights is greater than or equal to a specified threshold. That is,

$$min M_{js}^{'}, \; such \; that \; \sum_{m=1}^{M_{js}^{'}} c_{jms} \geq \eta \quad (5)$$

Table 4 compares the performance for different values of the threshold η. This method can provide a reduction of

η	$M_{j}^{'} = \sum_{s=1}^{S} M_{js}^{'}$	Accuracy (%)
Baseline	351.0	92.8
0.999	246.2	92.8
0.995	225.7	90.0
0.98	179.7	84.4
0.95	132.0	79.3

Table 4. Recognition performance on command words database for different value of the threshold η.

30% in storage compared to storing all non-zero component weights without loss of accuracy if we use a very conservative η. However, it is clear that relatively large number of mixture component weights must still be retained to maintain good performance. Instead of artificially setting the weights to zero, we can assign a constant non-zero value to those weights that are outside the criterion specified in Eq. (5). The weights c_{jms} in Eq. 1 are replaced by $\hat{c}_{jms}$,

$$\hat{c}_{jms} = \begin{cases} c_{jms}, & m \leq M_{js}^{'} \\ \frac{1}{(M_{js}^{z} - M_{js}^{'})} \sum_{m=M_{js}^{'}+1}^{M_{js}^{z}} c_{jms}, & M_{js}^{'} < m \leq M_{js}^{z} \\ 0, & m > M_{js}^{z}. \end{cases} \quad (6)$$

Here, $M_{js}^{'}$ is the number of mixture component weights that are represented exactly. The storage for mixture weights per reference model is given as

$$M_w = \sum_{s=1}^{S} \sum_{j=1}^{N} \left[1 + M_{js}^{'} \right] \text{ weights} + \sum_{s=1}^{S} \sum_{j=1}^{N} \left[M_{js}^{z} + 2 \right] \text{ indices}. \quad (7)$$

Table 5 shows that $M_{js}^{'}$ can be reduced significantly with considerable savings in storage needs while maintaining good performance.

η	$M_{j}^{'} = \sum_{s=1}^{S} M_{js}^{'}$	Accuracy 1-best (2-best) (%)
Baseline	351.0	92.8 (97.2)
0.90	83.0	92.1 (96.5)
0.80	77.6	90.0 (96.3)

Table 5. Recognition performance on command words database when threshold on sum of mixture weights is used to decide weights that are included and the remaining non-zero weights are replaced by an average value.

As an alternative to using the criterion in Eq. (5), we consider another criterion to selecting $M_{js}^{'}$ based upon the value of mixture weight. In other words,

$$\text{Select } M_{js}^{'}, \text{ such that } c_{jms} < \psi \text{ for } m > M_{js}^{'}, \quad (8)$$

The recognition performance as a function of the threshold ψ is shown in table 6. Note that for similar performance, storage requirements are much lower when criterion in Eq. (8) is used.

ψ	$M_{j}^{'} = \sum_{s=1}^{S} M_{js}^{'}$	Accuracy 1-best (2-best) (%)
Baseline	351.0	92.8 (97.2)
0.008	75.8	92.3 (96.7)
0.02	37.6	93.0 (96.7)
0.05	13.1	91.9 (96.1)

Table 6. Recognition performance on command words database as a function of ψ. The remaining non-zero weights are replaced by an average value.

We have seen in table 2 that relatively few kernels are required for each state to maintain performance identical to the baseline performance. When this is combined with mixture component reduction based upon a threshold on the mixture weights, a large reduction in computations as well as in model storage and model accesses per input frame can be obtained as shown in table 7.

ψ	$M_{j1}^L, M_{j2}^L, M_{j3}^L$	$M_j^{'} = \sum_{s=1}^{S} M_{js}^{'}$	Acc.
Baseline	256, 256, 32	351.0	92.8%
0.008	24, 24, 16	75.8	92.8 %
0.05	24, 24, 16	13.1	92.6 %

Table 7. Recognition performance on office command words database when the number of mixture components is reduced based upon the mixture weight as well as the kernel likelihood.

3.3 Quantization of Mixture weights

In Eq. (6), we replace all mixture weights below a threshold by their average. In this section, we consider representing all the mixture weights using a codebook. This offers the advantage that a sufficiently small codebook can be stored in a faster internal memory to reduce memory access costs while the indices are stored in external memory.

Table 8 shows the recognition accuracy for the connected digits car-database as a function of the codebook size. The codebook is designed using LBG [8] algorithm using L1 distance measure on the mixture weights. Also shown in the table is the average number of mixture weights $\hat{M}_{js}^z$ per state and stream that are mapped to non-zero codewords.

Codebook Size	$\hat{M}_{j1}^z, \hat{M}_{j2}^z, \hat{M}_{j3}^z$	Digit Accuracy (%)
Baseline	113.6, 85.8, 28.2	95.5
256	103.2, 76.3, 27.4	95.2
128	101.7, 74.7, 27.3	95.3
64	78.8, 56.5, 25.5	95.2
32	66.1, 45.2, 24.1	95.1
16	61.9, 41.9, 23.7	95.1

Table 8. Recognition performance on digits database as a function of the size of the mixture weight codebook.

Firstly, note that the number of mixture weights that are mapped to a non-zero codeword decreases substantially for smaller codebooks which results in reduced storage and computational requirements for kernel indices. Furthermore, the degradation in accuracy is relatively small even for codebook sizes down to 16.

In table 9, we present the recognition results when mixture weight quantization is used along with truncating the sum in Eq. (1) to include the top few kernel likelihoods. The results are shown for a mixture weight codebook of 128 values. A comparison of table 2 and table 9 shows that there is no additional degradation in recognition performance due to quantization of mixture weights. Furthermore, table 9 shows that we need the top 40, 40, and 24 kernels for three streams, respectively to maintain performance.

4 DISCUSSION

In this paper, we have addressed the issue of complexity reduction in a low-cost HMM based speech recognition application, In particular, we showed that tied-mixture hidden Markov models can provide significant advantage in memory and computational cost over continuous mixture HMMs. We presented new methods, such as mixture weight quantization, to represent the mixture component weights in a tied-mixture HMM and maintain good performance

$M_{j1}^L, M_{j2}^L, M_{j3}^L$	Digit/String Accuracy (%)
Baseline	95.5 / 81.0
256, 256, 32	95.2 / 80.5
40, 40, 24	95.2 / 80.3
24, 24, 16	95.1 / 79.8
10, 10, 10	95.0 / 79.5

Table 9. Recognition performance on digits database as a function of number of kernel likelihoods included in computing state observation likelihood when a mixture weight codebook of size 128 is used.

while achieving reduction in storage and access costs significantly. For small vocabulary isolated word recognition tasks, tied-mixture HMMs also offer performance comparable to continuous-mixture models at a much lower complexity. For the more difficult task of connected-digit recognition, there is approximately 10% loss in digit recognition accuracy with tied-mixtures HMMs.

The choice between continuous and tied-mixture HMMs for a low-cost application should depend upon the vocabulary size, task complexity, and available memory and CPU resources.

REFERENCES

[1] S. K. Gupta, F. K. Soong, and R. Haimi-Cohen, "High-accuracy connected-digit recognition for mobile applications," in *Proc. Int. Conf. Acoust. Speech Sign. Process.*, vol. I, pp. 57–60, 1996.

[2] P. Lockwood and J. Boudy, "Experiments with a non-linear spectral subtractor (NSS), hidden Markov models and the projection, for robust speech recognition in cars," in *Proceedings Eurospeech*, pp. 79–82, 1991.

[3] E. Erzin, A. E. Çetin, and Y. Yardimci, "Subband analysis for robust recognition in the presence of car noise," in *Proc. Int. Conf. Acoust. Speech Sign. Process.*, vol. 1, pp. 417–420, 1995.

[4] R. Yang and P. Haavisto, "Noise compensation for speech recognition in noise environments," in *Proc. Int. Conf. Acoust. Speech Sign. Process.*, vol. 1, pp. 433–436, 1995.

[5] X. D. Huang and M. A. Jack, "Semi-continuous hidden Markov models for speech signals," *Computer Speech and Language*, vol. 3, pp. 239–251, 1989.

[6] J. R. Bellegarda and D. Nahamoo, "Tied mixture continuous parameter modeling for speech recognition," *IEEE Trans. Acoustics Speech Signal Process.*, vol. 38, no. 12, pp. 2033–2045, 1990.

[7] E. Boccheiri, "A study of the beam-search algorithm for large vocabulary continuous speech recognition and methods for improved efficiency," in *Proceedings Eurospeech*, pp. 1521–1524, 1993.

[8] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Comm.*, vol. COM-26, pp. 702–710, April 1980.