SUB-VECTOR CLUSTERING TO IMPROVE MEMORY AND SPEED PERFORMANCE OF ACOUSTIC LIKELIHOOD COMPUTATION

M. Ravishankar, R. Bisiani and E. Thayer*
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA-15213, USA.
*Dept. of Computer Science, University of Milan, Italy
Tel. +1 412 268 3344, FAX: +1 412 268 5576, E-mail: rkm@cs.cmu.edu

ABSTRACT

We describe a *sub-vector clustering* technique to reduce the memory size and computational cost of continuous density hidden Markov models (CHMMs). Acoustic models in modern large-vocabulary, continuous speech recognition systems are typically CHMMs. Systems with 100,000 Gaussian distributions of 40-60 dimensions are common, needing several tens of MB of memory. Computing HMM state likelihoods is several tens of times slower than real time. We show that by clustering and quantizing the Gaussian distributions *a few dimensions at a time*, both computation and memory costs can be reduced several fold without significant loss of recognition accuracy. On the 1994 Wall Street Journal 20K test set, this technique reduced the acoustic model size by a factor of 9-10, and HMM state output likelihood computation time by a factor of 4-5.

1. INTRODUCTION

Acoustic models in most state-of-the-art speech recognition systems are based on fully continuous hidden Markov models or CHMMs ([1]). They are both computation and memory hungry. There are typically 1,000-10,000 distinct shared HMM states in such systems. Each state may be modelled by a mixture density consisting of 10-100 multi-dimensional Gaussian distributions. The dimensionality of these functions is usually in the range of 40-60. As a result, these acoustic models can require several tens of Mbytes of memory.

The computation of HMM state likelihoods during recognition is dominated by one function: in each frame evaluating the *distance* of the input speech vector from every one of the Gaussian means. Given the large number of Gaussians and their dimensionality, this computation can be many tens of times slower than real time.

A number of methods have been proposed for speeding up the state likelihood computation ([2,3,4,5]). They usually rely on hierarchically clustering the densities to form a series of smaller, coarser models. An initial search of the coarser models identifies a subset of the states to be evaluated with the fully detailed models. The remaining state likelihoods may be approximated using the coarser models. Some other methods exploit the fact that very few of the component densities that make up a state's mixture distribution actually determine its likelihood in a given frame. (The identity of these *active* densities, of course, changes from frame to frame.) The coarser models help identify the active densities quickly. The remaining need not be evaluated. By such methods, the computational load can be reduced several fold with little or no loss of recognition accuracy. However, memory requirements actually increase because of the additional, coarse models.

In this paper we describe a new clustering approach that reduces both computation and memory requirements of CHMM systems. The main problem with existing techniques is that they cluster entire vectors as a single unit to build the coarse models. Therefore, the resulting models have rather large quantization errors and cannot be used directly in the likelihood computations. The original, detailed models must still be retained. In contrast, the proposed method breaks up each vector into several sub-vectors, and clusters the fragments piecemeal. As the units being clustered are shorter in length, quantization errors are much smaller. Therefore, the quantized vectors can be directly used in state likelihood computation. The smaller size of the clustered models results in both reduced computation time and memory requirement.

The improvement in speed with this new method is comparable to existing techniques, with little or no loss in recognition accuracy. The memory requirement, on the other hand, can be reduced by an order of magnitude.

The rest of this paper is organised as follows. The proposed clustering technique is described in detail in Section 2. In Section 3 a number of experimental results are presented, with concluding remarks in Section 4.

2. SUB-VECTOR CLUSTERING

In this section we describe the proposed clustering technique in detail. This work has been carried out in the context of the Sphinx-3 speech recognition system ([6]). The speech input in each 10msec frame is a 39-dimensional feature vector. It is a concatenation of a 12-element *cepstrum* vector, signal power, and their first and second order differences in time.

The CHMM acoustic models in Sphinx-3 consist of a separate mixture Gaussian distribution for each shared state, or *senone*. The models considered in this work include 6000 senones, each of which is defined by a 16-

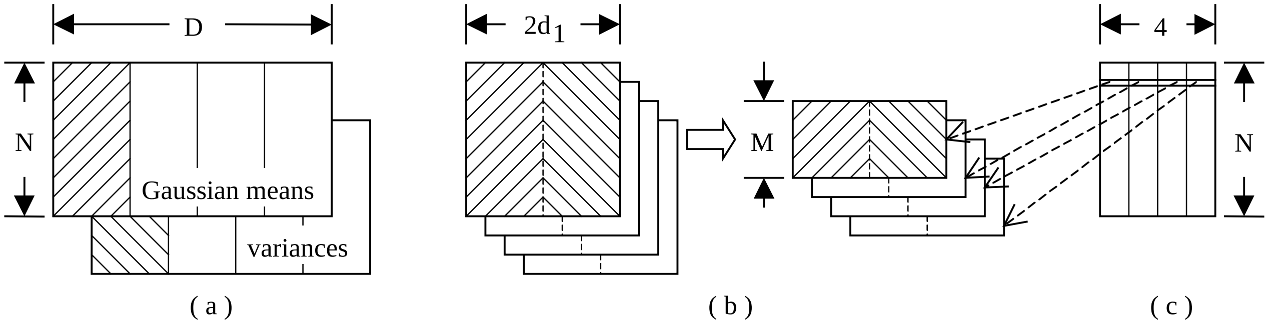

Figure 1: The clustering algorithm. (a) Original codebooks (b) Sub-array clustering (c) Cluster-index codebook**.**

component mixture Gaussian with diagonal covariances (henceforth simply referred to as *variances*). Thus, there are a total of 96,000 39-dimensional means and variances in the system.

The following discussion is based on the above model. Nevertheless, it should become quite obvious that the technique can be applied to other forms as well, including systems with multiple feature streams.

2.1. The Clustering Algorithm

As mentioned earlier, the main difference between conventional clustering schemes and the proposed one is in the choice of the unit to be clustered. We cluster only a *subset* of the dimensions of the mean and variance vectors at a time. In other words, the unit of clustering is a *sub-vector* within the full 39-dimensional feature vector. In the extreme case, each of the 39 dimensions may be clustered independently.

We first introduce some terminology. Let:

N = Total no. of Gaussian densities in the original acoustic model,

D = The feature dimensionality (39 in Sphinx-3),

$d_1, d_2, d_3 \ldots d_K$ = Some partition of D into K sub-vectors (*i.e.*, $d_1 + d_2 + d_3 + \ldots + d_K = D$),

M = Number of clustered Gaussian densities to be created for each d_i., $i = 1, 2, \ldots, K$.

Thus, the input to the algorithm are the $N \times D$ mean and variance arrays. We refer to them collectively as the *original codebook*.

The output of the algorithm is the following:

- A set of K quantized or *clustered codebooks*, mean and variance sub-arrays of size $M \times d_i$, $i = 1,2,\ldots,K$.
- A *cluster-index codebook* of size $N \times K$.

The two together replace the original codebook.

The clustering algorithm works as follows (see Figure 1 for an example with K=4):

1. Partition both the $N \times D$ mean and variance arrays *vertically* into K sub-arrays of size $d_1, d_2, d_3 \ldots d_K$.
2. Concatenate each of the K variance sub-arrays *horizontally* to its corresponding mean sub-array.
3. Cluster each such concatenated sub-array into M groups, and determine the centroid of each.
4. Split up the resulting centroid sub-arrays vertically back into *clustered codebook* means and variances.
5. Build the cluster-index codebook by replacing each mean and variance sub-vector in the original codebook with an index of its cluster centroid. (the dashed arrows in Figure 1(c)).

For the clustering in step 4 above, a straightforward *k*-means algorithm was used, see [7]. Thus, the procedure is basically vector quantization of concatenated mean and variance sub-vectors. As observed earlier, clustering shorter length sub-vectors keeps quantization errors low.

We note that the scalar mean and variance values in the original codebook have a lot of redundancy. It is the multivariate density as a whole that gives the system its discriminative capability. Hence, small amounts of quantization error are tolerated quite well by the system.

2.2. Reduction in Computation

Let us first see how the state likelihood computation is transformed as a result of the clustering. Originally, the D-dimensional input speech vector in each frame is compared to each of the N Gaussian distributions. This *Mahalanobis distance* is proportional to:

$$\frac{1}{(\sigma_1 \sigma_2 \cdots \sigma_D)} e^{-\frac{1}{2}\left[\sum_{i=1}^{D} \frac{(x_i - \mu_i)^2}{\sigma_i^2}\right]}$$

In Sphinx-3 this expression is actually evaluated in log-space. Hence, the exponentiation is avoided, and the total number of operations required is approximately $7ND$: three memory accesses and four floating point arithmetic operations for each of the D dimensions of N Gaussian distributions. (We have ignored array and loop-indexing operations since modern compilers and processors hide these costs through various techniques such as loop-unrolling and superscalar operation.)

After clustering, the computation becomes the following:

1. Break up the input speech vector into k sub-vectors of length $d_1, d_2, d_3 \ldots d_K$.
2. For each sub-vector, compute its *partial* Mahalanobis distance w.r.t. each of the M entries in the corresponding clustered codebook.

3. For each of the N vectors in the cluster-index codebook, use its K indices to look up the relevant partial distances computed above and sum them up.

The number of operations now has three components:
1. $7MD$: Computing partial Mahalanobis distances,
2. MK: Storing the results of above partial distances,
3. $3NK$: Computing the final results (two memory loads for an index and partial distance values, and a floating point accumulation),

i.e., $7MD+MK+3NK$. Hence, if $M<<N$ and $K<<D$, we can significantly reduce the computational load.

Clearly, there is a tradeoff between reducing M and reducing K; they cannot simultaneously be made arbitrarily small. The smaller we make K (i.e., the more dimensions we cluster as a unit), the larger must M be to avoid excessive quantization errors.

Secondly, for a given K, there are many different ways of partitioning D into $d_1, d_2, d_3 \ldots d_K$. Some of these may be better for clustering than others. This aspect is briefly discussed in Section 3 below.

2.3. Reduction in Memory Size

It is straightforward to figure out the comparative memory sizes of the two systems. The main data structures in the original case are the mean and variance arrays. Each requires ND 32-bit floating point values, or a total of $8ND$ bytes. In the Sphinx-3 system described earlier, with N=96000 and D=39, this amounts to about 30Mbytes.

The clustered system, similarly, requires $8MD$ bytes for the clustered codebook means and variances. The cluster-index codebook contains NK index values, which can be represented with 2-byte short integers. Finally, we need an additional $4MK$ bytes for storing the intermediate partial Mahalanobis distance floating point values while computing the output likelihoods. Hence, the total memory size is $8MD+2NK+4MK$. This limit is essentially achievable in practice. The actual reduction in memory size, of course, depends on how small M and K can be made without affecting recognition accuracy.

3. EXPERIMENTS

We applied the above codebook clustering technique to our fully continuous acoustic models trained from the SI-284 Wall Street Journal data ([8]). The form of these models has been described at the beginning of Section 2. In particular, N=96000, and D=39.

Several different cluster and partition sizes (*i.e.*, M and K) were tried in our experiments. The two values for K reported in this paper, 4 and 7, were chosen as follows. The 39-dimension feature vector space was separated into cepstrum, its first and second order differences, and the power dimensions, giving us four partitions. The 7 partitions were obtained by splitting all but the power dimensions further into two. Moreover, for a given value of K, we also varied the exact partition of D into $d_1, d_2, d_3 \ldots d_K$. We only report the best ones.

		Cluster size			
		2K	4K	6K	8K
No. of Partitions	7	13.3	12.9	12.7	13.1
	4	13.7	13.6	13.7	13.0

Table 1: Percentage word error rates on h1_et_94. (The baseline error rate is 12.7%.)

3.1. Recognition Accuracy

We evaluated the clustered models on the 1994 H1-C0 DARPA evaluation test set (also known as *h1_et_94*). This set consists of read-style Wall Street Journal and North American business news sentences. It was decoded with the standard 20K-word vocabulary and language model used in those evaluations.

Table 1 shows the performance (percentage word error rate) of the system for different choices of cluster size and no. of partitions; *i.e.*, M and K, respectively. As expected, the larger value of K performs better, because vectors being clustered are shorter, with correspondingly smaller quantization errors. It is, in fact, possible to match the baseline error rate with a suitably chosen value for M. In general, with K=7, the relative degradation in error rate is under 5% for the number of clusters shown. Interestingly, and contrary to expectations, the performance does not seem to improve monotonically with increasing M. We touch upon this phenomenon further in Section 3.4.

Experiments on the 1994 20K development test set (also known as *h1_dt_94*) showed similar results. The relative increase in word error rate was 0-2% with K=7, and 2-4% with K=4.

One drawback of the method is the rather restricted range of K that is useful. Smaller values of K begin to degrade recognition accuracy. On the other hand, larger values of K are less interesting because of the smaller theoretical limits on performance improvement.

3.2. Computation Time

The speedup of the clustered system over the baseline in computing the acoustic likelihoods is shown in Table 2. Table 2(a) shows the actual speedup attained on a Pentium-Pro PC workstation when both versions are implemented in a straightforward manner. In other words, we evaluate the acoustic likelihoods for all senones in a given frame before moving on to the next frame. The observed speedup is about 75% of the theoretical maximum discussed in Section 2.2.

Table 2(b) shows the speedup over a baseline system that has been optimized for memory cache performance using the technique of *blocking*. That is, instead of evaluating all the senones one frame at a time, we compute the output likelihood for a single senone in several frames as a block, and repeating this process for all the senones. The mixture Gaussian for the single senone is kept in cache longer, improving performance.

		Cluster size			
		2K	4K	6K	8K
No. of Partitions	7	7.3	6.3	5.1	4.1
	4	10.0	8.4	7.2	6.2

(a)

		Cluster size			
		2K	4K	6K	8K
No. of Partitions	7	5.1	4.4	3.6	2.9
	4	6.9	5.8	5.0	4.3

(b)

Table 2: Speedup (a) Over straightforward baseline (b) Over cache-optimized baseline implementation.

However, it is certainly feasible to apply blocking to the clustered system as well (though not quite so easily) and improve upon the figures shown in Table 2(b).

3.3. Memory Usage

Table 3 shows that with clustering, there is an order of magnitude reduction in memory requirement for the acoustic models. Thus, the 30MB of data in the baseline system can be easily compressed to within 4MB or less, depending on the configuration chosen.

3.4. Discussion

The *k*-means clustering algorithm we have chosen is one of the simplest. Also, it is applied in quite a coarse way to fairly long vectors. Even so, the decrease in recognition accuracy is surprisingly small and it points out that substantial redundancy is present in current CHMM representations. This also implies that the degree of sharing, and hence efficiency, could be increased even further with more sophisticated clustering algorithms. For example, the clustered structure may be directly incorporated in a forward-backward training algorithm.

We also saw in Table 2 that the error rate occasionally increases, in spite of increasing the cluster size, *M*. This could mean that the simple-minded *k*-means clustering algorithm, while capable of producing good models, is not *robust* enough to do so predictably. The need for robustness is another reason for employing the more sophisticated forward-backward algorithm to estimate the clustered codebooks.

4. CONCLUSION

We have shown that sub-vector clustering can be applied to Gaussian distributions of CHMMs to greatly reduce

		Cluster size			
		2K	4K	6K	8K
No. of Partitions	7	14.7	10.9	8.7	7.3
	4	20.8	14.2	10.8	8.7

Table 3: Ratio of acoustic model memory size in baseline system to that in clustered system.

both their memory size and the cost of computing state output likelihoods. Quantization errors introduced by such clustering can be kept under control by selecting the cluster and partition sizes suitably. On the 1994 Wall Street Journal 20K test sets, this technique reduces the acoustic model size by a factor of 9-10, and state likelihood computation time by a factor of 4-5. The relative degradation in word error rate is less than 2%.

ACKNOWLEDGEMENTS: We would like to thank Vipul Parikh, Paul Placeway, and other member of the CMU Sphinx speech group for their comments.

This research was sponsored by the department of the Navy, Naval Research Laboratory under Grant No. N00014-93-1-2005. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

REFERENCES

[1] Proceedings of the DARPA Speech Recognition Workshop, Chantilly, VA, Feb 2-5, 1997.

[2] Seide, F., “Fast Likelihood Computation for Continuous-Mixture Densities Using a Tree-Based Nearest Neighbor Search”, *Proc. Eurospeech,* Vol. II, pp. II--1079-1082, Sep. 1995.

[3] Beyerlein, P. and Ulrich, M., “Hamming Distance Approximation for a Fast Log-Likelihood Computation for Mixture Densities”, *Proc. Eurospeech*, Vol. II, pp. II--1083-1086, Sep. 1995.

[4] Komori, Y. *et al*, “An Efficient Output Probability Computation for Continuous HMM Using Rough and Detail Models”, *Proc. Eurospeech,* Vol. II, pp. II--1087-1090, Sep. 1995.

[5] Fritsch, J. *et al*, “Speeding up the Score Computation of HMM Speech Recognizers with the Bucket Voronoi Intersection Algorithm”, *Proc. Eurospeech*, Vol. II, pp. II--1091-1094, Sep. 1995.

[6] Placeway, P. *et al*, “The 1996 Hub-4 Sphinx-3 System”, *Proc. DARPA Speech Recognition Workshop*, Feb. 1997.

[7] Gray, R.M. “Vector Quantization”, Readings in Speech Recognition, Ed. Waibel&Lee, Morgan Kaufmann Publishers, CA.

[8] Kubala, F. “Design of the 1994 CSR Benchmark Tests”, *Proc. DARPA Spoken Language Systems Technology Workshop*, pp. 41-46, Jan. 1995.