# SUBSPACE DISTRIBUTION CLUSTERING FOR CONTINUOUS OBSERVATION DENSITY HIDDEN MARKOV MODELS

*Enrico Bocchieri and Brian Mak**
AT&T Labs-Research, 180 Park Ave, Florham Park, NJ 07932.
(*) Oregon Graduate Institute, 20000 NW Walker Rd, Portland OR, 97006.
enrico@research.att.com and mak@research.att.com

## ABSTRACT

This paper presents an efficient approximation of the Gaussian mixture state probability density functions of continuous observation density hidden Markov models ($CHMM$'s). In $CHMM$'s, the Gaussian mixtures carry a high computational cost, which amounts to a significant fraction (e.g. 30% to 70%) of the total computation. To achieve higher computation and memory efficiency, we approximate the Gaussian mixtures by (a) decomposition into functions defined on subspaces of the feature space, and (b) clustering the resulting subspace pdf's. Intuitively, when clustering in a subspace of *few dimensions*, even few function codewords can provide a small distortion. Therefore, we obtain significant reduction of the total computation (up to a factor of two), and memory savings (up to a factor of twelve), without significant changes of the $CHMMM$'s accuracy.

## 1. INTRODUCTION

Most of state-of-the-art speech recognition systems are based on hidden Markov models ($HMM$) technology. In particular, continuous observation density $HMM$'s ($CHMM$) have the highest speech recognition accuracy in many applications. However, the continuous state probability density functions (pdf) have a high computational cost, which amounts to a significant fraction (30% to 70%) of the total computation. Therefore techniques designed to speed up the state likelihood computations are important in practice.

In this paper we study an approximation of existing, highly accurate CHHM's using distribution clustering on subspaces of the frame feature space. We refer to our formulation as $SDCHMM$ (Subspace Distribution Clustering for $CHMM$'s). The proposed approximation of the $CHMM$ state pdf's consists of:

1) Decomposition into functions defined on subspaces (*streams*) of the feature space, and
2) Clustering (quantizing) the pdf's of the feature subspaces.

The clustering (or quantization) of pdf's allows for significant reduction of the number of model parameters. Computation and memory efficiency is improved by sharing few codeword functions among the state likelihood definitions. At the same time, the quantization in subspaces of low dimensionality gives small distortions and insignificant variations of recognition accuracy.

In this work, we derive the new $SDCHMM$'s from existing $CHMM$'s without retraining on speech data. The $SDCHMM$ formulation is presented in the next few sections. We then discuss the differences between $SDCHMM$ and other $HMM$ types based on *stream* distribution tying (tied mixtures) and *feature level* tying [1,2]. We then present the $SDCHMM$ recognition results on the ARPA-ATIS continuous speech recognition task.

## 2. $SDCHMM$ FORMULATION

We start from $CHMM$'s (possibly with *tied states*) with state density functions defined by Gaussian mixtures:

$$p_s(\mathbf{O}) = \sum_{m=1,M} c_{s,m}\ N(\mathbf{O},\mu_m,{\sigma^2}_m),$$
$$\sum_{m=1,M} c_{s,m} = 1.0 \qquad (CHMM) \qquad (1)$$

where $s$ is the $CHMM$ state index, and $N(\mathbf{O},\mu_m,{\sigma^2}_m)$ , $m=1,M$ is the set of all Gaussians of all state mixture models, with means $\mu_m$ and diagonal covariances ${\sigma^2}_m$. $\mathbf{O}$ is a frame feature vector of dimension $d$.

We divide the $d$ frame features into $K$ *streams* or subspaces, each of dimension $d_k$ so that $\sum_{k=1,K} d_k = d$. Since covariances are diagonal, each Gaussian in (1) can be expressed as the product of $K$ Gaussians (one for every stream):

$$N(\mathbf{O},\mu_m,{\sigma^2}_m) = \prod_{k=1,K} N(\mathbf{O}_k,\mu_{m,k},{\sigma^2}_{m,k}) \qquad (2)$$

then (1) becomes:

$$p_s(\mathbf{O}) = \sum_{m=1,M} c_{s,m} \prod_{k=1,K} N(\mathbf{O}_k,\mu_{m,k},{\sigma^2}_{m,k}) \qquad (3)$$

where ( $N(\mathbf{O}_k,\mu_{m,k},{\sigma^2}_{m,k})$, $m=1,M$ , $k=1,K$ ) are $K$ sets of $M$ Gaussians (one set per subspace).

We cluster the Gaussians of the $k^{th}$ ($k=1,K$) subspace, into a set of fewer *quantized* codeword Gaussians

$$N^{cw}(\mathbf{O}_k,\mu_{q,k},{\sigma^2}_{q,k}) \ , \ q=1,Q \ , \ Q \ll M \qquad (4)$$

We then approximate each original Gaussian with the closest *quantized* Gaussian:

$$N(.,\mu_{m,k},{\sigma^2}_{m,k}) \approx N^{quan}(.,\mu_{m,k},{\sigma^2}_{m,k}) = argmin_{q=1,Q}\ dist(N(.,\mu_{m,k},{\sigma^2}_{m,k}),N^{cw}(.,\mu_{q,k},{\sigma^2}_{q,k})) \qquad (5)$$

Note that the efficiency of the algorithms in (4) and (5) is not critical, because they are performed only once and the results stored before recognition. Therefore (3) becomes:

$$p_s(\mathbf{O}) \approx \sum_{m=1,M} (c_{s,m} \prod_{k=1,K} (N^{quan}(\mathbf{O}_k, \mu_{m,k}, {\sigma^2}_{m,k}))) \qquad (SDCHMM) \quad (6)$$

For efficiency in both $CHMM$ and $SDCHMM$, we also approximate the $\sum$ operator with the maximum operator. Therefore, the logarithm of (6) becomes:

$$ln(p_s(\mathbf{O})) \approx \max_{m=1,M} (ln(c_{s,m}) + \sum_{k=1,K} ln(N^{quan}(\mathbf{O}_k, \mu_{m,k}, {\sigma^2}_{m,k}))) \qquad (7)$$

In terms of computation and memory, (6) is more efficient than (3) and (1), because it uses a smaller number $Q \ll M$ of unique Gaussians for every stream. At the same time, the quantization in low dimensional feature spaces allows for a small distortion in (5) and a small loss of recognition accuracy.

## 3. STREAM DEFINITION

Our recognizer frontend is based on the mel-cepstrum analysis of the input speech sampled at 16kHz. Every 10 ms, the sampled waveform is analyzed to extract 12 mel frequency cepstrum coefficients (MFCC). For every input sentence, we subtract from all the MFCC vectors the average (per sentence) MFCC vector (Cepstrum Mean Subtraction). Every frame feature vector is made of $d = 39$ components, consisting of 12 MFCC's and of the frame energy in dB with their $1^{st}$ and $2^{nd}$ derivatives.

With $SDCHMM$'s, we need to divide the 39 feature components into subspaces or streams. We have actually tested different stream definitions as follows:

$K =$ 1 stream for the entire feature space.

$K =$ 4 streams, for the MFCC's, $1^{st}$ MFCC derivatives, $2^{nd}$ MFCC derivatives, and energy components (with $1^{st}$ and $2^{nd}$ derivatives), respectively.

$K =$ 13 streams, one for each of the 12 MFCC's or energy, with its $1^{st}$ and $2^{nd}$ derivatives.

$K =$ 20 streams, each (with the exception of one stream) containing two *strongly correlated* features.

$K =$ 39 streams, one for every feature component.

In the $K = 20$ stream case, we first estimate a grand correlation matrix for the 39 dimension feature space. The feature couples for each stream were iteratively selected by (1) picking the two features with the largest correlation, and (2) removing from the correlation matrix the rows and columns corresponding to the selected features. As a sanity check, we have experimentally verified that randomly generated feature couples do not perform as well.

## 4. GAUSSIAN CLUSTERING

To define the Gaussian codewords (4) of the $k^{th}$ stream, we cluster all the $k^{th}$ stream Gaussians (2) of the $CHMM$'s. The $CHMM$'s have been trained by segmental $k$-means, and, in the case of context dependent models, also by a bottom-up state-tying algorithm [3]. For each Gaussian, we store the mean vector, the diagonal covariance and the size of the feature ensemble used for parameter estimation.

For Gaussian clustering we apply an iterative bottom-up algorithm, with the original Gaussians as initial codewords. We then *merge* a Gaussian pair at a time, chosen to minimize the distortion increase given by merging the respective ensembles [3]. We stop the iteration when the number of codewords is reduced to the desired value. By keeping track of how the original Gaussians have been merged into codewords we also define their mapping (5) into the final Gaussian codewords.

In the particular case of $K = 39$ one-dimensional streams, we have obtained better experimental results with a simpler and faster Gaussian clustering procedure which we call *equal percentile clustering*:

a) Sort the Gaussians according to their mean value,

b) Put the Gaussians into *consecutive* disjoint sets, while adding up the respective ensemble sizes, so that the ensemble size of the various sets are roughly equal. Then estimate mean and variance of each set (codeword).

## 5. LIKELIHOOD COMPUTATION

Typically in a beam-search recognizer, the state log-likelihoods (7) are computed *on-demand*, i.e. only for the *active* states. However, to really take advantage of the $SDCHMM$ structure, we have implemented some changes in this strategy.

There are only few unique Gaussians $N^{quan}()$ in (7) ($Q$ for each stream). They are shared among many state densities and, for every input frame, the log-likelihood of almost every $N^{quan}(\mathbf{O}_k, ., .)$ is needed. For computation speed, it is important to pre-compute these values and to store them in a contiguous table for efficient access. Then the computation of (7) is performed *on-demand*, by accessing through pointers the stored values of $ln(N^{quan}(\mathbf{O}_k, ., .))$.

Another issue with the $SDCHMM$ likelihood computation concerns Gaussian Selection [4,5]. Gaussian Selection of the codewords (4) is not useful, because these are relatively few in number. However Gaussian Selection is still useful when applied to the $M$ $d$-dimensional Gaussians:

$$N(\mathbf{O}, \mu_m, {\sigma^2}_m) \approx \prod_{k=1,K} N^{quan}(\mathbf{O}_k, \mu_{m,k}, {\sigma^2}_{m,k}), \qquad m = 1, M \quad (8)$$

Gaussian Selection allows one to approximate some of the Gaussians on the left-hand-side of (8) to naught.

## 6. COMPARISON WITH OTHER $HMM$'S

Tied-mixture or semi-continuous $HMM$'s ( $TMHMM$ ) are computationally less expensive than $CHMM$'s. Since the $TMHMM$'s divide the frame feature space into *streams*, the $TMHMM$ formulation may appear similar to the $SDCHMM$'s (6). However, the $TMHMM$ formulation is different:

$$p_s(\mathbf{O}) = \prod_{k=1,K} \sum_{m=1,M_k} c_{s,m,k} \, N^{quan}(\mathbf{O}, \mu_{m,k}, {\sigma^2}_{m,k}) \qquad (TMHMM) \quad (9)$$

Note that while (1) and (3) do not assume statistical independence of the different streams, (9) does assume independence. In fact $SDCHMM$'s are designed as approximations of $CHMM$'s, while $TMHMM$'s are not.

In the particular case of $K = d$ streams of one-dimension, the $SDCHMM$ formulation (6) becomes equivalent to *feature-level* tying [1,2]. However we have obtained additional computation and memory savings with other stream definitions ($K = 13$, $K = 20$), that cannot be used with the formulation [1,2]. Let us consider the computation of the exponent (essentially the log-likelihood) of a $d$-dimensional Gaussian:

$$0.5 \sum_{i=1,d} \frac{(o_i - \mu_i)^2}{{\sigma_i}^2} \qquad (10)$$

The $K = d$ one-dimensional stream $SDCHMM$'s and the feature-level tying system [1,2], save computation because the addenda can be precomputed and shared among many Gaussians. However, still $d$ summations must be carried-out, which become the largest fraction of the log-likelihood computation. $SDCHMM$'s with $K < d$ streams reduce the computation further, because they add only $K$ terms, instead of $d$.

## 7. EXPERIMENTS

The new $SDCHMM$'s have been tested on the ARPA-ATIS spontaneous speech recognition task (official December 1994 test set). The $SDCHMM$ performance has been compared to two conventional $CHMM$ systems:

a) 49 context independent (CI) phone $CHMM$'s, with a total of 2302 Gaussians of 39 dimensions.

b) 9,863 context dependent (CD) tied-state triphone $CHMM$'s, and a total of 76,725 Gaussians.

Both use a 1,532 word lexicon, a word-class bigram language model, and frame-synchronous beam-search. The lexicons have tree and linear structures respectively.

We have converted the $CHMMM$'s to $SDCHMM$'s, with different numbers of streams and Gaussian codewords. Word error rates for various configurations (but with the same beam width) for the context independent case are shown in Figure 1. The baseline accuracy of the $CHMM$ system, corresponding to 2302 Gaussians in Figure 1, is 9.4%. For low dimensionality streams, i.e. 13, 20 and 39 stream cases, the number of Gaussian codewords (shown on the x-axis) can be substantially reduced to 64, 16 and 8 respectively, without significant changes of the word error rate.

Figure 2 shows the corresponding recognition times (including feature extraction, search and likelihood computation) on a 150 MHZ R4400 SGI workstation. The baseline $CHMM$ system, corresponding to 1-stream and 2302 Gaussians, takes a recognition time of 1.8 times real time. The 13-stream, 64-Gaussian $SDCHMM$, and the 20-stream, 32-Gaussian $SDCHMM$ speedup the recognition to real time. Note that the 39-stream case (feature-level tying), is not as efficient, for the reason explained in the previous Section.

Figure 3 shows the combined word error rate and recognition time for context dependent $SDCHMM$'s, in the 39-stream and 20-stream cases. The baseline context dependent $CHMM$'s give a 5.2% word error rate, at 21 times real time. The 20-stream, 64 Gaussian $SDCHMM$'s reduce the recognition time by a factor

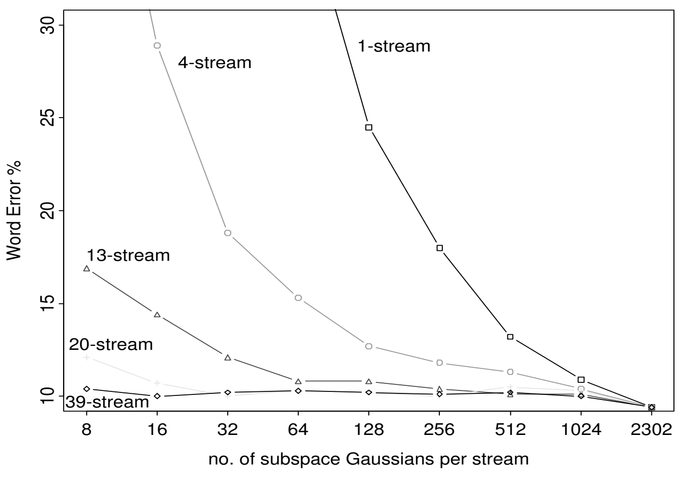

**Figure 1. Context independent $SDCHMM$ word error rates.**

of two, with only a 0.2% absolute word error rate increase. As in the context-independent case, the 20-stream $SCDHMM$'s are computationally more efficient than the 39-stream one-dimensional feature level tying system.

All results in Figures 1 through 3 were obtained without Gaussian Selection for the likelihood computation. Gaussian Selection [4] applied as in Section 5, further reduces the total recognition time of the fastest $SDCHMM$ configurations by 10 to 15%.

$SDCHMM$'s reduce the number of $CHMM$ Gaussian mean and variance parameters by a factor $\frac{M}{Q}$, that is about $70 \approx 2,302/32$ and $1,200 \approx \frac{76,725}{64}$ for our context independent and context dependent models respectively. After taking into account the mapping structure required to implement the $SDCHMM$'s, we reduce $CHMM$ memory by 12, 11, and 7, for $K = 13$, $K = 20$, and $K = 39$ streams (assuming one-byte per pointer). Therefore, the $SDCHMM$ formulations ($K = 13$, $K = 20$) appears more memory efficient than feature-level tying ($K = 39$).

## 8. CONCLUSION

The proposed $SDCHMM$'s provide computation and memory savings with respect to the widely used, highly accurate continuous density $HMM$'s ($CHMM$), without almost any loss in recognition accuracy.

In this study, the $SDCHMM$ have been derived as an approximation of existing $CHMM$'s, without retraining on speech data. We plan to investigate training of the $SDCHMM$'s directly from the speech data. We feel that, since $SDCHMM$ use a small number of Gaussian parameters, this method has further potential when few training data are available, as in speaker/environment adaptation.

## 9. REFERENCES

[1] S.Takahashi, S.Sagayama, "Four-Level Tied-Structure for Efficient Representation of Acoustic Modeling," Proc. ICASSP-95, pp. 520-523.

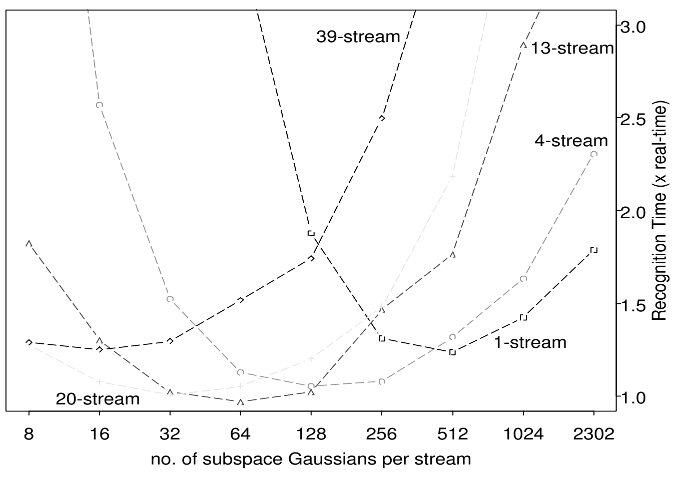

**Figure 2. Context independent *SDCHMM* recognition times.**

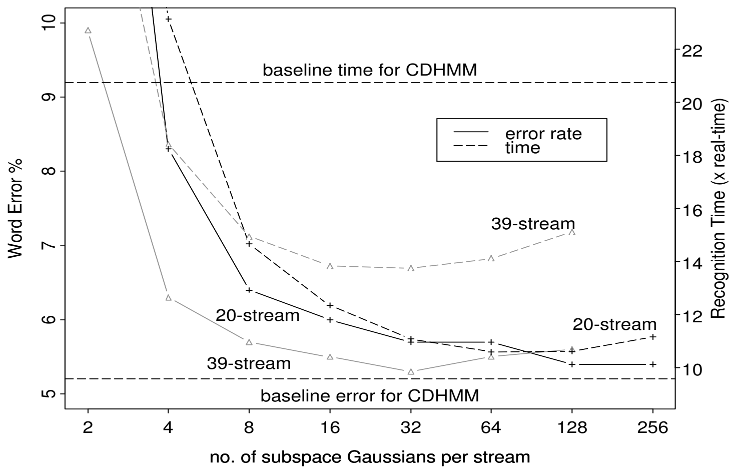

**Figure 3. Error rates and recognition times of context dependent *SDCHMM*'s.**

[2] S.Takahashi, S.Sagayama, "Effects of Variance Tying for Four-Level Tied Structure Phone Models," Proc ASI, March 95, pp 141-142.

[3] E.Bocchieri, G.Riccardi "State tying of triphone *HMM*'s for the 1994 AT&T ARPA ATIS Recognizer," Proc Eurospeech 95, pp 1499-1502.

[4] E.Bocchieri, "Vector Quantization For Efficient Computation Of Continuous Density Likelihoods", Proc. ICASSP-93, vol II, pp 692-695.

[5] K.Knill, M Gales, S.Young, "Use of Gaussian Selection In Large Vocabulary Continuous Speech Recognition Using *HMM*'S", Proc. ICSLP-96, vol 1, pp 470-473.