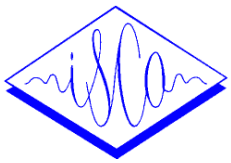

6th International Conference on Spoken Language Processing (ICSLP 2000)
Beijing, China
October 16-20, 2000

SPEECH RECOGNITION USING HMMS WITH QUANTIZED PARAMETERS

Marcel Vasilache

Speech and Audio Systems Laboratory, Nokia Research Center
P.O. Box 100, 33721 Tampere, Finland
E-mail: marcel.vasilache@nokia.com

ABSTRACT

In this paper we describe the structure and examine the performance of a recognition engine based on hidden Markov models (HMMs) with quantized parameters (qHMM). The main goal of qHMMs is to enable a low complexity implementation without sacrificing the classification performance. In the tests with a whole word digit dialler engine and a phoneme based isolated word recognizer we managed to preserve the performance of unquantized HMMs with qHMMs having as little as 5 bit for a mean component and 3 bit for a variance component.

1. INTRODUCTION

Since its introduction, the hidden Markov model has been the most successful and widely used tool in speech recognition. According to the type of the generating distributions HMMs can be divided into continuous and discrete models. Within each of these classes additional partitions are made. The discrete HMMs (DHMM) are usually differentiated according to the type of quantization on the feature space (e.g. scalar or vector quantization with single or multiple codebooks). For the continuous density models (CDHMM) this classification is done according to the type of the generating density (e.g. diagonal/full-covariance Gaussian or non-Gaussian).

Complexity is a key factor for most speech recognition engines which often leads to compromises in the choice of the acoustic modeling. DHMMs with moderate sizes for the quantization codebooks have the advantage of a low computational complexity requirement due to the table look-up implementation of the computation of B-probabilities. However, it is a common belief that CDHMMs can obtain a better performance when compared with DHMMs for a given recognition task. In CDHMMs case the B-probability computation starts to become an expensive operation and, sometimes, also memory limitations impose an upper bound for the total number of generating densities. In [1] the authors show that the performance gap between DHMMs and CDHMMs could be due to low rate quantization of the feature vectors. With a more elaborate scheme, split vector quantization and discrete-mixture state generating probability, the DHMMs are shown to have a similar performance as the reference CDHMMs. With pruning of B-probabilities these DHMMs are also less expensive in computations but they do require significantly more memory.

One aspect well known by the DSP engineer is the parameter representation when implementing the HMM based recognition engines. In many cases the standard IEEE floating point representations are replaced with fixed point implementations with smaller precision. By HMMs with quantized parameters we denote such cases where the parameters of the HMMs are represented in substantially less bits than the 32 bit/parameter of the IEEE floating point format. In the following we'll have a closer look on the implementation, performance and advantages of a parameter quantization scheme for HMMs.

2. CDHMM WITH QUANTIZED PARAMETERS

2.1. Structure

Parameter quantization is possible with all types of HMMs. Complex quantization schemes could partition the densities and split the feature vector into several quantization classes. For CDHMMs having densities with full covariance matrices an extra partitioning could be done using rotation classes. However, in this paper, we focus on CDHMMs based on Gaussian densities with diagonal covariance matrices. We selected for quantization only the mean and variance vectors. The remaining parameters; state transition probabilities and mixture weights were represented in standard 32 bit floating point format. It is possible to quantize them too since the recognition performance is less influenced [1] by these parameters and therefore has more robustness to quantization errors for them. However, the memory savings would be rather insignificant since their number is usually exceeded with orders of magnitude by mean and variance parameters.

In the classification process the various components of the feature vector are not of equal importance therefore an optimal bit allocation scheme should take this into account. However, due to simplicity and memory savings, we have used only two scalar quantizers, component-wise, for means and variances, respectively. This is possible only if using a normalization scheme for the feature vectors which, based on the estimated global mean and variance of the training set transforms the input into features with zero mean and unity variance components. It is known that the potential recognition performance is not influenced with such an approach since an invertible component-wise linear transform is used[2]. For CDHMMs with full covariance densities this can be extended to an invertible linear transform.

Typical distributions after normalization for means, variances and the scatter plot are shown in the figures at the end of the paper.

[1] In the cases when the feature vector dimension is large

[2] CDHMMs trained on the original data could be transformed for giving identical results on the normalized data.

2.2. Advantages

With the structure presented in the previous subsection qHMMs provide an intermediate model between discrete HMMs and CDHMMs. The major advantages of the model consist in; reduction of the memory footprint, reduction in the computational complexity as it will be shown next and similar performance to CDHMMs as it was found experimentally.

For states with mixtures of Gaussian densities the B-probability formula is:

$$b(\mathbf{x}) = \sum_{k=1}^{K} w_k \frac{1}{\prod_{i=1}^{n} \sqrt{2\pi\sigma_{ki}^2}} \exp\left(\sum_{i=1}^{N} -\frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2}\right) \quad (1)$$

where K represents the number of densities in the mixture and N is the dimension of the feature vector space.

Equation (1) in logarithmic domain becomes:

$$\log b(\mathbf{x}) = \log \sum_{k=1}^{K} \exp\left\{ \log\left(w_k \frac{1}{\prod_{i=1}^{N} \sqrt{2\pi\sigma_{ki}^2}}\right) - \sum \frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2} \right\} \quad (2)$$

In the last equation we notice that for each density there are two terms, a constant (i.e. the log of the mixture weight times the Gaussian normalization factor[3]) and the Mahalanobis distance to the feature vector $\mathbf{x}$.

In computing the B-probability the most costly operation is the computation of the Mahalanobis distance. The $\log \sum_k \exp$ operation requires approximatively $3(K-1)$ flops and $K-1$ table look-ups in an optimized implementation. This is usually negligible to the cost of computing K Mahalanobis distances if the feature vector dimension is large.

Computation of the Mahalanobis distance for one density requires one subtraction, two multiplications and one addition per feature vector component [4]. If each of these floating point operations have the same cost we have $4NK$ flops as the dominant cost for computing the state score. When we use quantization for the μ_{ki} and $1/\sigma_{ki}$ parameters we should also add the cost of table look-ups. However, this cost could be ignored because integer indexing operations can be performed in parallel with floating point operations by most modern processors.

When μ_{ki} and $1/2\sigma_{ki}^2$ are quantized it is noticeable that, for a given feature vector $\mathbf{x}$, the terms $\frac{(x_i-\mu_{ki})^2}{2\sigma_k^2}$ take a discrete set of values. The size of this set, for each component i, is equal to the product of the number of quantizer levels for means (M) and standard deviations (S). Precomputing these values requires $NM(S+2)$ flops for every frame. In addition to the codebooks we need the storage space for these NMS precomputed floating point values (i.e. one MS sized table for each feature vector component) [5]. With the help of this data the Mahalanobis distance computation reduces to summing up values from the tables above indexed with the joint mean and standard deviation quantizers indexes. The cost for one B-probability computation is now lowered to NK flops (ignoring again the costs of table indexing and the $\log \sum_k \exp$ operation).

The question to ask is $ND + NM(S+2) \lesseqgtr 4ND$ where D stands for the total number of densities we have to compute. This is equivalent to:

$$M(S+2) \lesseqgtr 3D \quad (3)$$

If the left size of equation (3) is smaller it is more efficient to use precomputed values, otherwise the direct computation path should be followed. With conventional CDHMMs the left side value is very big therefore no reductions in computation could be achieved with this approach [6]. However, for qHMMs it may often be the case that the $M(S+2)$ product is much smaller that the $3D$ term leading to substantial reductions in the number of operations. In the limit when $M(S+2) \ll 3D$ a near 4 times speed-up in the computation of B-probabilities is obtainable.

Alternative approaches for speeding up the computation of B-probabilities have been reported during the recent years e.g. [2], [3] and [4]. In [2] and [3] this is done by clustering densities to be computed using a vector quantizer. In [4] it is achieved by scalar quantizing the input features and, based on precomputed tables, discard the computations for part of the densities. However, all methods will increase the memory requirements. In the qHMM's case a deterministic computation speed-up is obtained at the same time with a reduction of the memory footprint. For both types of complexities there is a constant overhead given by the storage of the codebooks and the helper table computations. It is worth mentioning that, in some cases, the methods referred could also be applied to qHMMs. For instance, if the sizes of the density clusters are large enough, as required by equation (3), computations could be speeded-up even more. In addition, if also the feature vectors are quantized it could become feasible to have globally precomputed tables and thus no need to update them on a frame by frame basis. In general, to reduce overheads, small quantization rates are essential.

2.3. Training

In training two basic methods of parameter optimization are possible with respect to quantization:

- static
- dynamic

In the static approach, given the trained models, the quantization is performed as a final step. As inputs to he quantization process, in addition to the model parameters themselves, additional information could be given (e.g. total occupation probability for each parameter to be quantized if ML training was used or a similar measure in case of discriminative training).

In the dynamic approach the training iterations are intermixed with quantizer training and parameter quantization steps.

[3] If the given density is shared by several states it is necessary to store the log mixture weight separately.

[4] Standard deviations are stored in inverse form and include the $\frac{1}{\sqrt{2}}$ coefficient.

[5] If the feature vector size is very large there is the option of using only one such table and computing all B-probabilities at once, component by component

[6] The memory overhead is also prohibitive.

Both of the previously mentioned methods are empiric. In the correct approach a joint optimization procedure for quantizers and parameters is required.

3. EXPERIMENTS

3.1. Experimental Setup

In the initial experiments we used a speaker independent, continuous digit dialling engine for the German language with whole word HMMs. The vocabulary consisted in 11 digit models having a left to right structure with enforced duration constraints [5].

We used a conventional front-end based on FFT derived Mel cepstral coefficients, their first and second order time derivatives. Altogether there were 39 components in the feature vector.

In training there were about 52000 digits from 397 speakers. The test set had sentences of 4 digits and contained a total of 6240 digits from 229 speakers not included in the training set. The recognitions were done with unknown sentence length and the total error rate is given (substitutions plus deletions and insertions).

An additional test was done using a speaker independent, phoneme based isolated word recognition engine. Being a noise robust recognition task, the front-end had also a normalization scheme as presented in [6].

The vocabulary consisted of 120 names. In the test database there were 50 male and 32 female speakers with 240 utterances for each (every name in the vocabulary being spoken twice). The names were built using phoneme models, each one a CDHMM with three states and a left to right structure with no skips. These models were trained on a different database containing phoneme rich sentences.

For all test cases, the models had mixtures of Gaussian densities for the state generating distributions. The same number of densities was used for all states.

3.2. Experimental Results

Static Quantization

In the static quantization approach we used two Lloyd-Max quantizers for the mean and the inverse standard deviation parameters of the ML trained models.

Results for models with mixtures of 4, 8 and 12 densities per state are shown in Tables 1-3. All error rates are expressed in percentages. The bit rates for means and standard deviations are shown on lines and columns, respectively. The error rates of the original models were, respectively: 1.83, 1.60 and 1.55.

$\log_2(S)$ / $\log_2(M)$	*1*	*2*	*3*	*4*
2	3.14	3.03	3.01	2.98
3	2.36	2.20	2.08	2.13
4	2.42	2.10	2.04	2.04
5	2.21	2.05	2.00	1.89
6	2.12	1.97	1.89	1.88

Table 1: Digit error rates with quantization, 4 mixtures.

At first glance the main message is of a high robustness to quantization errors of the CDHMMs for this task. At a total rate of only 3 bits for both mean and standard deviation the error rate about doubles. With as little as 5+3 bits the performance is similar to the original models for 8 and 12 mixtures. For 4 mixtures an extra bit may be needed. Regarding complexity, at 5+3 quantization, the B-probability computation costs, respectively, 38%, 32.5% and 29.3% of the original models requirement.

$\log_2(S)$ / $\log_2(M)$	*1*	*2*	*3*	*4*
2	3.33	2.72	2.42	2.36
3	2.40	2.04	1.81	1.78
4	2.28	1.97	1.75	1.67
5	2.13	1.92	1.62	1.57
6	2.05	1.94	1.60	1.60

Table 2: Digit error rates with quantization, 8 mixtures.

$\log_2(S)$ / $\log_2(M)$	*1*	*2*	*3*	*4*
2	3.04	2.39	2.12	2.08
3	2.48	2.04	1.84	1.83
4	2.42	1.84	1.60	1.57
5	2.16	1.79	1.52	1.51
6	2.20	1.78	1.57	1.54

Table 3: Digit error rates with quantization, 12 mixtures.

Tables 4 and 5 contain the main results of the names recognition experiments. “clean” stands for noise-free recorded data. In “noise” we use various noise types and SNR ranges to corrupt the noise-free test data. Only the 5 bit mean and 3 bit standard deviation quantization rates was tested.

Method	*avg*	*male*	*female*
clean	**1.61**	2.0	1.0
clean qHMM	**1.61**	2.0	1.0
noise	**14.18**	15.0	12.9
noise qHMM	**14.66**	15.4	13.5

Table 4: Names error rates, 8 mixture models.

As with the digit recognition task, the recognition performance differences are minimal. The complexity figures were 39% and 32% relative to the original models.

Dynamic Quantization

Due to the increased complexity of the dynamic quantization only a few cases were tested. The procedure consisted in iterating a three step transformation:

1. ML re-estimation
2. quantizer training
3. quantization of the models

Ten such iterations were done. In ML training of the reference HMMs the same number of iterations was used and both procedures were started with identical initial models. It was observed that the convergence of the procedure was adequate, the

Method	*avg*	*male*	*female*
clean	**1.05**	1.4	0.5
clean qHMM.	**1.11**	1.5	0.5
noise	**11.72**	12.5	10.5
noise qHMM	**11.64**	12.3	10.6

Table 5: Names error rates, 16 mixture models.

likelihoods increasing steadily [7]. However, in the end, the results were not better than with the static quantization therefore the extra training complexity is not justified. This outcome is not surprising since there is a high utilization of only two basic quantizers.

4. CONCLUSION

CDHMMs with quantized parameters provide a valuable option when the computational complexity is a key design limitation. Even with a simple construction using only two scalar quantizers qHMMs provided impressive results at low quantization rates. With practically no recognition performance degradation substantial reductions in the computational complexity were achieved.

More complex quantization structures are possible, for instance with splitting the feature vector and partitioning densities into different quantization and/or rotation classes. However, for exploiting the advantages of qHMMs, an efficient training and low quantization rates are essential.

Acknowledgment

The author would like to thank David Bye from Nokia Mobile Phones for the discussions about implementation aspects of speech recognition engines which triggered the research presented in this paper.

5. REFERENCES

[1] Tsakalidis S., Digalakis V., and Neumeyer L. Efficient speech recognition using subvector quantization and discrete-mixture HMMs. In *ICASSP'99*, pages 569–572, 1999.

[2] Bocchieri E. Vector quantization for the efficient computation of continuous density likelihoods. In *ICASSP'93*, pages II–692–695, 1993.

[3] Suontausta J., Hakkinen J., and Viikki O. Fast decoding techniques for practical real-time speech recognition systems. In *IEEE Automatic Speech Recognition and Understanding Workshop*, pages 169–172, Keystone, USA, Dec. 1998.

[4] Sagayama S. and Takahashi S. On the use of scalar quantization for fast HMM computation. In *ICASSP'95*, pages 213–216, 1995.

[5] Laurila K. Noise robust speech recognition with state duration constraints. In *ICASSP'97*, pages 871–874, 1997.

[6] Viikki O., Bye D., and Laurila K. A recursive feature vector normalization approach for robust speech recognition in noise. In *ICASSP'98*, pages 733–736, Seattle, USA, May 1998.

[7] But being always under the corresponding values from the reference training.

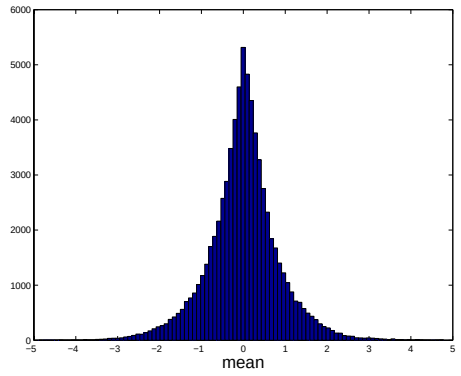

Figure 1: Histogram of the mean values.

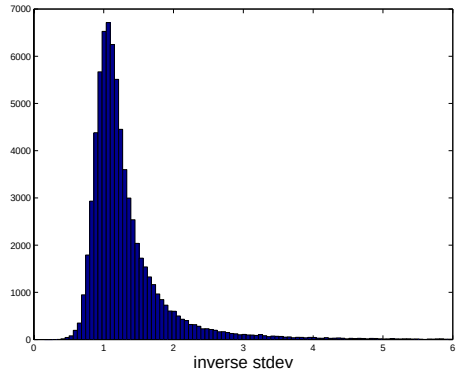

Figure 2: Histogram of the inverse standard deviation values.

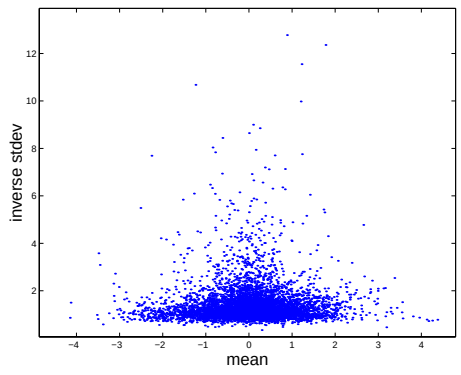

Figure 3: Scatter plot of means and inverse standard deviations.