Comparison of Low Footprint Acoustic Modeling Techniques for Embedded ASR Systems

Jussi Leppänen and Imre Kiss

Multimedia Technologies Laboratory, Nokia Research Center
Tampere, Finland
{jussi.ar.leppanen, imre.kiss}@nokia.com

Abstract

In this paper we compare the performance of speech recognition systems based on hidden Markov models (HMM) with quantized parameters (qHMMs) and subspace distribution clustering hidden Markov models (SDCHMMs). Both of these HMM types provide similar performance as continuous density HMMs, but with significantly reduced memory requirements (approximately 90% less memory was needed to store the HMM densities). The experiments show that on a small vocabulary isolated word recognition task, SDCHMMs outperform qHMMs in clean conditions. However, when noisy test data is used and adaptation is enabled qHMMs outperform the SDCHMMs. In addition, the experiments show that as low as 3-bit feature quantization can be used with both qHMMs and SDCHMMs without sacrificing recognition performance.

1. Introduction

Voice User Interfaces are becoming a standard feature in personal mobile devices, such as mobile phones. Several devices in the market support voice input in the form of speaker dependent or speaker independent name dialing, while others are able to read aloud messages by using text-to-speech (TTS) technology.

For these embedded devices, the low footprint and low complexity implementation of the underlying ASR and TTS algorithms is especially important. One of the reasons for this is that even though memory and processing power are becoming more and more affordable, cost will always be an important factor for mass-market products.

In addition, Voice UI is only one of the rich set of features supplied in modern mobile devices, and in many cases several of these features need to be active at the same time.

In this paper we concentrate on low footprint acoustic modeling techniques for embedded ASR systems. In Sections 2 and 3 we describe the two main footprint reduction techniques (SDCHMMs and qHMMs). In Section 4 we compare the properties of these techniques, and continuous density HMMs. In Section 5 we compare these two techniques in terms of effect on recognition accuracy and the associated memory footprint. Conclusions are drawn in Section 5.

2. Subspace Distribution Clustering HMMs

Subspace distribution clustering HMMs (SDCHMMs) were introduced by Bocchieri and Mak in [1]. They were shown to provide computational and memory savings, without degrading recognition performance, when compared to continuous density HMMs. The basic idea is similar to density tying, but the tying is done on subspace distributions instead of the full-space distributions. The SDCHMM implementation used in this paper differs slightly from the one described in [1] and [2] and is explained briefly below.

SDCHMMs are trained by first dividing the continuous distributions of the original HMM model set into orthogonal subspaces and then clustering these subspace distributions into prototype distributions. The clustering is done separately for every subspace, so that as a result there is a separate prototype set (codebook) for each subspace. The parameters of the original distributions can then be replaced by indices to the codebooks.

The clustering of the continuous distributions is done using a binary, divisive k-means clustering algorithm. The algorithm starts with all distributions belonging to a subspace in a single cluster. At each iteration of the algorithm, every cluster is split into two and then k-means is run for a few iterations. The algorithm is run until the desired number of clusters (codebook size) is obtained. The distance measure used in the k-means algorithm is the Bhattacharya distance [3].

Before the clustering can be done, the subspaces have to be defined. This is done by finding the features that correlate the most and grouping them together to form the subspaces. The multiple correlation measure, for example, can be used to find the most correlated features [2]. In this paper, all experiments using SDCHMMs are done with single-feature subspaces (these were found to work best in our tests). Thus, subspace definition was not needed in our experiments.

3. Quantized HMMs

Quantized HMMs (qHMMs) were proposed in [4]. They were found to reduce memory consumption substantially while maintaining the high recognition accuracy of continuous density HMMs.

QHHMs are built starting from continuous density HMMs by scalar quantizing the mean and variance parameters. Here only two global quantizers are used, one for the mean values and one for the variance values. The use of one quantizer for all mean values and one for all variance values is potentially problematic as the dynamic range of the feature components varies from component to component. Applying a global normalization of the feature space such that zero mean and unity variance components are achieved, however, mitigates this problem. This is in contrast with the SDCHMM case, where there is a separate quantizer (or codebook) for each subspace and thus no normalization is required[1]. The actual quantization is done using a non-linear Lloyd-Max quantizer. The distance measure used during training of both the mean and variance quantizer was the Euclidian distance.

[1] The use of a single codebook for all features was also tried for SDCHMMs, but it resulted in reduced performance.

4. Comparisons

4.1. Codebooks

Figure 1 and Figure 2, below, show the mean vs. inverse standard deviation scatter plots of the 2nd cepstral coefficient with 4-bit qHMM and SDCHMM codebooks superimposed. The differences in the codebooks for SDCHMMs and qHMMs can be seen quite clearly. The SDCHMM codebook elements are nicely situated on top of the densities. For the qHMMs, the placement of the codebook elements does not appear to be very optimal. This is explained by the fact that the same codebook is used for all features and is optimal over all of them.

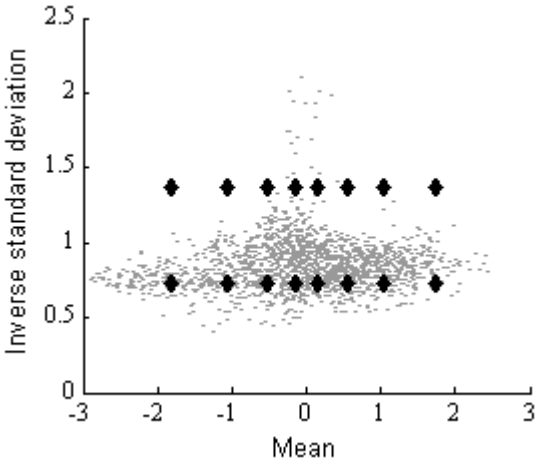

Figure 1 Scatter plot of the 2nd cepstral coefficient superimposed with the 4-bit qHMM codebook values.

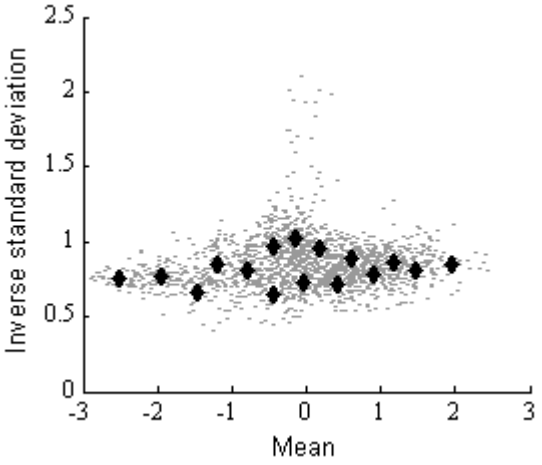

Figure 2 Scatter plot of the 2nd cepstral coefficient superimposed with the 4-bit SDCHMM codebook values.

4.2. Probability calculation and feature quantization

During decoding, the calculation of the state probabilities for mixture Gaussians in log-domain is done using the following formula:

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

where K is the number of densities and N is the feature vector dimension. For each density, there are two parts, a constant and the Mahalanobis distance to the feature vector x.

When using SDCHMMs or qHMMs, once a feature vector is obtained, the Mahalanobis distances can be calculated for the codebook elements. Once this is done, the actual probabilities for the densities can be calculated by summing up the appropriate distances for each density. In the qHMM case, the number of distance calculations is less than in the SDCHMM case because the same codebook is used for all features. For example, when using 39 element feature vectors and 4-bit codebooks, the number of distance calculations that are needed for qHMMs is 16, one for each codebook element. For SDCHMM the corresponding number is (16x39=) 624. This is quite many compared to the qHMM case, but still very efficient (the number of summations needed to calculate the probabilities for 3000 densities for example is still (39 x 3000=) 117000).

The probability calculation can be further speeded up by the use of feature quantization [6]. When the incoming feature vectors are quantized the Mahalanobis distances of the above equation can be calculated before the decoding begins. Here again the number of distance calculations is greater for SDCHMMs than for qHMMs.

4.3. Memory footprint

When using SDCHMMs or qHMMs, the mean and variance values of the densities are replaced by indices to the codebooks. The space needed to store the indices depends on how large the codebooks are. For example, 39 bytes are needed to store a single density when 8-bit codebooks and 39 element feature vectors are used. This is significantly lower than for the continuous case where (32bits x 2 x 39=) 312 bytes are needed (assuming that 32-bit floating point numbers are used for the mean and variance values). Since the densities take up the majority of space required for the whole model set, significant memory saving can be achieved. Storing the codebooks, however, requires memory, but this is usually insignificant compared to the memory required for the densities. This is the case also for SDCHMMs, where a codebook is required for every subspace. The memory required for the model sets used in this paper can be found in Table 2.

5. Experiments

5.1. Recognition system and acoustic model sets

The performance of qHMMs and SDCHMMs was tested on a small vocabulary isolated word task in several languages. The in-house test set used here was made up of approximately 40k

words from seven European languages: Finnish, Swedish, German, English, Danish, Icelandic and Norwegian. The size of vocabulary per language was approximately 120.

The front-end used in the experiments was based on FFT-derived Mel cepstral coefficients and their first and second order derivatives (39 coefficients in total). Recursive mean removal was applied on all components of the resulting feature vectors, and the variance of the energy component and its derivatives was normalized to unity [7].

The baseline acoustic model sets used for the tests contained standard 3-state monophone models with 8 and 16 densities per state. The model sets were trained on an in-house training set containing data from various European languages. Both sets contained a total of 75 multilingual phone models that were used to model the basic acoustic units of the seven European languages mentioned above. The number of densities in the 16-mixture set was 3568 and for the 8-mixture set there were 1784 densities. The recognition accuracies of the baseline sets can be seen in Table 1, labeled as BL_8 and BL_16.

For both of the baseline model sets, a pair of qHMM model sets was trained. One set using 5-bit quantization for the density means and 3-bit quantization for the variances, and the other set using 3 and 1 bits, respectively. The recognition accuracies for these models can be seen in Table 1, labeled as Q_x_y+z, where x is the number of densities per state, y the number of bits used in quantizing the means and z the number of bits used for quantizing the variances. The 5+3 and 3+1 quantizations used here were chosen for the reason that they allow convenient packing of the mean-variance pairs into bytes.

Similarly, four SDCHMM systems were also trained. These sets were trained based on the two baseline sets resulting in 16-mixture and 8-mixture models that have 4-bit and 6-bit codebooks. The recognition rates can be found in Table 1, labeled similarly to the qHMMs. There are no recognition rates for 8-bit codebooks because the number of densities in the model sets was not sufficient for training them.

Model set	Accuracy	Model set	Accuracy
BL_8	96.51%	BL_16	97.33%
Q_8_5+3	96.53%	Q_16_5+3	97.24%
Q_8_3+1	95.58%	Q_16_3+1	96.55%
SS_8_6b	96.38%	SS_16_6b	97.26%
SS_8_4b	96.41%	SS_16_4b	97.19%

Table 1 Word accuracies (clean speech, no adaptation).

From the above results, it can be seen, that using a 5+3 bit codebook for qHMMs and 6-bit codebooks for SDCHMMs only marginally degrades the performance. When reducing the codebook size to 4 bits, further degradation of the performance can be seen. However, this degradation is much smaller for the SDCHMMs than for the qHMMs. Note that, 16-mixture, 4-bit codebook SDCHMMs perform better than 8-mixture, 5+3-bit qHMMs, even though their memory footprint is comparable (see Section 5.2).

5.2. Memory figures

As mentioned before, significant saving in the memory footprint of the acoustic models can be achieved when using qHMMs or SDCHMMs. In Table 2, below, the memory required to store the densities of the various acoustic model sets used in these experiments is shown. For the baseline models, it is assumed that the mean and variance values are represented using 32 bit floating-point numbers. For the qHMMs and SDCHMMs, when 4-bit quantization is used, the memory needed is calculated by assuming that two 4-bit values are stored in a single byte. In the 6-bit SDCHMM model set, it is assumed that, the 6-bit values are each stored in a byte. While the memory needed for the densities is similar for qHMMs and SDCHMMs, the memory needed to store the codebooks is not. Because of the use of separate codebooks for each stream, the codebooks of the SDCHMMs require more memory. However, since the densities require much more memory the memory needed for the codebooks is not very significant.

Model set	Densities	Codebooks	Total
BL_8	557kB	-	557kB
BL_16	1.1MB	-	1.1MB
Q_8_3+1	35kB	20B	35kB
Q_8_5+3	70kB	80B	70kB
Q_16_3+1	70kB	20B	70kB
Q_16_5+3	140kB	80B	140kB
SS_8_4b	35kB	1kB	36kB
SS_8_6b	70kB	5kB	75kB
SS_16_4b	70kB	1kB	71kB
SS_16_6b	140kB	5kB	145kB

Table 2 Memory needed for the storage of densities and codebooks of qHMMs and SDCHMMs.

For the rest of the experiments, the recognition rates will be shown for model sets Q_8_5+3, Q_16_3+1, SS_8_6b and SS_16_4b. These were chosen as their memory footprint is about the same size (~70kB).

5.3. Performance in noise and with adaptation

Next, the performance of qHMMs and SDCHMMs were tested in noisy conditions and with adaptation. Table 3 shows the recognition rates for the baseline systems and for two qHMM model sets and two SDCHMM model sets. The adaptation method used here was supervised MAP, which was applied to the models after each recognized word [8]. Only the model means were adapted. Also, the actual codebooks were not adapted; only the density indices pointing to the codebooks were updated during adaptation [5]. The noisy data was created by randomly adding noise from different sources and at different levels. Car, café, concert noise were included. The SNR of the noisy data ranged from 5 to 20dB.

	Clean		Noise	
Model set		Adapt		Adapt
BL_8	96.51%	98.59%	87.47%	94.49%
BL_16	97.33%	98.93%	88.23%	95.04%
Q_8_5+3	96.53%	98.53%	87.53%	94.28%
Q_16_3+1	96.55%	98.32%	87.38%	92.49%
SS_8_6b	96.38%	98.41%	86.87%	93.25%
SS_16_4b	97.19%	98.62%	88.00%	92.77%

Table 3 Word accuracies in clean and noisy conditions (5 to 20dB SNR) with and without MAP adaptation.

As it can be seen, from Table 3, the recognition accuracies for clean speech with adaptation enabled are quite even for the qHMM and SDCHMM models. However, when noisy data was used, some differences in the scores can be observed. The SS_16_4b set gave the best performance and the SS_8_6b the worst. The accuracies for the two qHMM sets were between the accuracies of the two SDCHMM sets.
When adaptation was enabled and noisy data was used, significant differences in the recognition accuracies could be seen. While all cases benefited from the MAP adaptation, the recognition rates of the 8-mixture models were improved much more than the recognition rates of the 16-mixture models. The reason for this might be explained by the fact that both (qHMM and SDCHMM) 16-mixture models use 4-bit quantization which is coarser than the quantization used for the 8-mixture models (6-bit or 8-bit) and the fact that adaptation is done on the indices pointing to the density codebooks and not on the codebooks themselves. To elaborate, during adaptation a density index is changed only if the density after adaptation is closer to another codebook element than the one it was pointing to before. Thus, the coarser the quantization used, the less likely it is for a density index to change during adaptation. In addition, it can be seen from the results that qHMM models responded better to adaptation than SDCHMM models, especially in the presence of noise.

5.4. Feature quantization

As was mentioned in Section 4.2, the recognition speed can be further increased by the use of feature quantization. The recognition accuracies for a few qHMM and SDCHMM model sets and different feature quantizations are shown in Table 4. The feature quantizers used here were trained exactly like the mean and variance quantizers of the qHMMs and were trained on the mean values of the densities in the baseline model sets.

	Feature quantization			
Model set	**5 bits**	**4 bits**	**3 bits**	**2 bits**
Q_8_5+3	96.53%	96.56%	95.97%	91.54%
Q_16_3+1	96.49%	96.65%	96.43%	94.83%
SS_8_6b	96.45%	96.78%	95.72%	91.51%
SS_16_4b	97.19%	97.25%	96.78%	94.16%

Table 4 Word accuracies with feature quantization (clean speech, no adaptation).

From Table 4, above, it can be seen that, for SDCHMMs even as low as 3-bit quantization of the features does not degrade performance significantly. With 4-bit quantization, the performance is even better than without feature quantization. The same kind of performance can also be seen for qHMMs.

6. Conclusions

In this paper, we have compared the performance of two low footprint acoustic modeling techniques, qHMMs and SDCHMMs. Compared with continuous density HMMs, both techniques provided memory savings and probability calculation speed up without sacrificing recognition accuracy significantly. The memory footprint of qHMMs and SDCHMMs was found to be much smaller than for the continuous density HMMs. The densities of the qHMM and SDCHMM model sets could be represented in only 6% to 13% of the bytes needed for the continuous case. Moreover, the memory footprint of qHMMs was slightly smaller than the SDCHMM memory footprint because of the differences in the number of codebooks used.
The SDCHMMs outperformed the qHMMs in clean conditions (97.19% for the SS_16_4b set vs. 96.53% for the Q_8_5+3 set). This was also noticed for noisy test data (88.00% vs. 87.53%). When adaptation was enabled, however, the improvement in the results was greater for the qHMMs than for the SDCHMMs. This was most evident in the noisy case where the qHMMs clearly outperformed the SDCHMMs (92.77% vs. 94.28%).
In addition, the performance of qHMMs and SDCHMMs was tested with feature quantization. For both cases, even as low as 3-bit quantization was found to give acceptable recognition rates.

7. Acknowledgements

This work has partially been funded by the European Union under the integrated project TC-STAR - Technology and Corpora for Speech to Speech Translation -(IST-2002-FP6-506738, http://www.tc-star.org).

8. References

[1] Bocchieri E. and Mak B., "Subspace Distribution Clustering for Continuous Observation Density Hidden Marko Models", Proceedings of the 5th European Conference on Speech Communication Technology, Vol. 1, pp. 107-110, 1997.
[2] Bocchieri E. and Mak B., "Subspace Distribution Clustering Hidden Markov Model", *IEEE Transactions on Speech and Audio Processing*, 9(3) pp. 264-275, March 2001.
[3] Webb A., *Statistical Pattern Recognition*, Arnold, 1999.
[4] Vasilache M., "Speech Recognition Using HMMs with Quantized Parameters", Proceedings of the International Conference on Spoken Language Processing, Vol.1, pp. 441-443, Beijing, China, 2000.
[5] Vasilache M. and Viikki O. "Speaker Adaptation of Quantized Parameter HMMs", Proceedings of Eurospeech-Scandinavia, Vol. 2, pp. 1265-1268, Aalborg, Denmark, 2001.
[6] Vasilache M., Iso-Sipilä J. and Viikki O., "On a Practical Design of a Low Complexity Speech Recognition Engine", Proceedings of the International Conference on Acoustics, Speech and Signal Processing, Vol. 5, pp. 113-116, Montreal, Quebec, Canada, 2004.
[7] Viikki O., Bye D. and Laurila K., "A Recursive Feature Vector Normalization Approach for Robust Speech Recognition in Noise", Proceedings of the International Conference on Acoustic, Speech and Signal Processing, Seattle, WA, USA, 1998.
[8] Gauvain J.-L., Lee C.-H., "Maximum a Posteriori Estimation of Multivariate Gaussian Mixture Observations of Markov Chains", *IEEE Transaction on Speech and Audio Processing*, Vol. 2, No. 2, pp. 291-298, 1994.