MEMORY SPACE REDUCTION FOR HIDDEN MARKOV MODELS IN LOW-RESOURCE SPEECH RECOGNITION SYSTEMS

Sergey Astrov
Siemens AG, Corporate Technology, D-81730 Munich, Germany
sergey.astrov.external@mchp.siemens.de

ABSTRACT

Low-cost recognition systems based on hidden Markov models (HMM) for mobile speech recognizers (mobile phones, PDAs) have a limited quantity of memory and processing power. Furthermore, the resources have to be shared between several applications. In this paper memory efficient HMMs were investigated for low-cost recognition platforms. The feature parameter tying HMM and subspace distribution clustering HMM (SDCHMM) were explored. In order to achieve less memory requirements, a shared codebook approach for feature parameter tying HMM and SDCHMM was developed and its effectiveness was experimentally proved. It was shown that this approach leads to a relative increase of word error rate of less than 10% for 50% of memory reduction.

1. INTRODUCTION

Highly accurate HMM-based speech recognizers need much high speed memory (in the order of 100-1000 Kbytes), which is nowadays not widely available on mobile systems like mobile phones or PDAs, due to constraints of cost and energy consumption (battery). Currently the ROM of about 128-256 Kbytes is available for speech recognition. Taking into account the memory and processor computing power available within the next 1-2 years, the best coding schemes for reducing the memory requirements of HMMs on mobile systems have to be obtained. Most coding schemes lead to an increase of the recognition error rate. This increase should be as low as possible. Given all these constraints the following problem has to be solved: given the constraints of maximal computing power and minimal increase of word error rate (WER) which coding-decoding schemes of the HMM-parameters lead to minimal memory requirements?

In the last years the tying technique was considered as the universal approach that reduces memory consumption, reduces required processing power and increases the robustness of HMMs. Most of the research efforts were concerned with large- and medium-vocabulary systems. A recognizer with a vocabulary of 500 distinct words and system requirements of 50 MIPS processing power and 1 Mbytes DRAM was reported in [4]. The SDCHMM approach described in [1, 2] allows 7- to 18-fold reduction in memory requirements and a decrease of recognition time up to 30-60% without any loss of recognition accuracy. Nevertheless the average memory consumption in these approaches is still several hundreds of kilobytes for large-vocabulary recognizers.

In this paper SDCHMMs and their modifications were investigated for small-vocabulary tasks (the vocabulary is limited to 100 words).

The paper is organized as follows. In section 2 an overview of memory reduction techniques for HMMs is given. The feature parameter tying HMM and SDCHMM approaches are described and the modifications of stream organization and codebook sizes of SDCHMM are investigated. The estimation of memory sizes is given. The experiments and their results are described in section 3. Then follow the conclusions.

2. MEMORY SPACE REDUCTION

In low-cost recognition systems Gaussian mixtures occupy the most part of HMM ROM. The Siemens ‘ear’ recognizer which was the basis of the work of this paper, works with 24-dimensional feature vectors (1). Each component of the feature vector is represented by a 1-byte integer value.

\[
X = (x_1, x_2, ..., x_{24}).
\]  

(1)

The observation probability function \(b_s(x)\) of state \(s\) is a sum of \(M\) Gaussian mixtures \(\mu\) with a diagonal covariance matrix with equal variances \(\sigma\):

\[
b_s(x) = \sum_{m=1}^{M} c_{s,m} \mathcal{N}(x, \mu_{s,m}, \sigma),
\]

\[
\mathcal{N}(x, \mu_{s,m}, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left\{-\frac{1}{2} \sum_{n=1}^{N} \frac{(x_n - \mu_{s,m,n})^2}{\sigma^2}\right\}
\]

(2)

where \(N=24\) is the feature vector dimension.

In the experiments phoneme based HMMs as well as whole word HMMs were investigated. A phoneme-based context independent HMM for small-vocabulary recognizer typically consists of 4000 mixtures, a whole word HMM typically consists of 1200 mixtures. The memory requirements to store mixtures of these HMMs are 96000 bytes (96000=4000-24) and 28800 bytes respectively. In the following the memory reduction of mixture components is considered.

The memory reduction procedure consists of two stages. In the first, the coding stage the parameters of already trained HMMs are coded in an off-line mode. The coded HMM parameters occupy less memory.

In the second stage, the decoding is performed “on the fly” during the recognition process. Only the parameters of the HMM which are instantly needed are decoded, so that no extra memory for decompression of the whole HMM is required. The
coding-decoding procedure should be simple in order to keep
the required processing power as low as possible, the
cognition accuracy should not degrade dramatically.

In the following the memory reduction based on vector
quantization and tying techniques is considered.

A good overview of tying approaches is given in [3, 5].
Tying of HMM parameters is applied because of three major
reasons. Tied models have less parameters, thus less memory is
required. Furthermore, the reduction of computational load
during the recognition is possible. Tied HMMs are more robust
in the case of small amounts of training data available, as less
parameters have to be trained. This last reason is especially
important for large-vocabulary speech recognizers.

Two tied HMM modifications are tested and compared:
feature parameter tying HMM and SDCHMM.

2.1. Feature parameter tying HMM

The idea of feature parameter tying is shown in Fig. 1. The two
2-dimensional mixtures shown there can not be tied as the
Euclidean distance is large. Yet the mean values in the second
dimension are very close, and they can be tied in the feature
parameter level. The feature parameter tying is described in [3]
as the fourth level of parameter tying. The mean values of
Gaussian mixtures are merged into some representative mean
values in each dimension by using the clustering technique. In
[3] it was shown that the recognition performance did not
degrade with 16 representative mean values for every
dimension for 26 Japanese phoneme recognition tasks.

![Figure 1: Feature parameter tying](image)

Applying such a technique to HMMs used by the ‘ear’
recognizer with 16 representative mean values for every
dimension leads to 50% memory reduction: before tying each
component of mixture occupied 8 bits for 256 possible values
and after tying they occupy only 4 bits for 16 values.

The memory reduction is not exactly 50% as extra memory is
required to store the codebook of 16 representative mean
values for every dimension. In the ‘ear’ recognizer a codebook
needs an extra 384 bytes. To decrease the memory requirements
for the codebook, the shared codebook approach is proposed. For all dimensions together only one shared
codebook is generated.

2.2. Subspace distribution clustering HMM

SDCHMM was proposed in [1] where the conversion procedure for continuous density HMM was described. In [2]
the direct training of SDCHMM was presented. The theory of
SDCHMM is shortly shown in the following.

The set of $P$ mixtures is represented as a set of vectors (3)

\[
\begin{align*}
\{\mu_{1,1}, \mu_{1,2}, \mu_{1,3}, \mu_{1,4}, \ldots, \mu_{1,N-1}, \mu_{1,N}\} \\
\{\mu_{2,1}, \mu_{2,2}, \mu_{2,3}, \mu_{2,4}, \ldots, \mu_{2,N-1}, \mu_{2,N}\} \\
\ldots \\
\{\mu_{P,1}, \mu_{P,2}, \mu_{P,3}, \mu_{P,4}, \ldots, \mu_{P,N-1}, \mu_{P,N}\}
\end{align*}
\]

(3)

This set can be broken into subspaces (streams):

\[
\begin{align*}
\{\mu_{1,1}, \mu_{1,2}\} & \quad \{\mu_{1,3}, \mu_{1,4}\} & \quad \ldots & \quad \{\mu_{1,N-1}, \mu_{1,N}\} \\
\{\mu_{2,1}, \mu_{2,2}\} & \quad \{\mu_{2,3}, \mu_{2,4}\} & \quad \ldots & \quad \{\mu_{2,N-1}, \mu_{2,N}\} \\
\ldots & \quad \ldots & \quad \ldots & \quad \ldots \\
\{\mu_{P,1}, \mu_{P,2}\} & \quad \{\mu_{P,3}, \mu_{P,4}\} & \quad \ldots & \quad \{\mu_{P,N-1}, \mu_{P,N}\}
\end{align*}
\]

(4)

In (4) 2-dimensional streams are shown. One dimension represents one coefficient of the mixture. The first stream is the set of $P$ vectors $\{\mu_{p,1}, \mu_{p,2}\}$, the second stream is the set $\{\mu_{p,3}, \mu_{p,4}\}$ and the last stream is the set $\{\mu_{p,N-1}, \mu_{p,N}\}$, $p=1,...,P$.

In [1] several stream structures were designed. In the case of ‘ear’ recognizer the dimensions of stream vectors are almost
uncorrelated because of a linear discriminative analysis in the
feature extraction module, that is why the simple structure as
shown in (4) is used. $K$-dimension streams definition can thus
be described as follows: the first $K$ dimensions of mixtures are
placed in the first stream, the second $K$ dimensions form the
second stream, etc.

The continuous density HMM was converted to the
SDCHMM as it is described in [1]. The codebook was
generated using binary split vector quantization algorithm [6].

The memory consumption of stream indexes is estimated as
shown in (5).

\[
\text{mem}_{\text{indexes}} = PKQ,
\]

(5)

where $P$ denotes the number of mixtures,
$K$ denotes the number of streams,
$Q$ denotes the size of stream indexes in bytes.
The memory consumption of independent codebooks (in bytes) is estimated as shown below:

$$\text{mem}_{\text{codebooks}} = KLDM \ . \tag{6}$$

where $K$ denotes the number of streams, $L$ denotes the dimension of streams, $M$ denotes the number of vectors in the codebook, and $D$ denotes the size of mixture components in bytes.

The memory consumption of the shared codebook is

$$\text{mem}_{\text{shared \_codebook}} = LDM \ . \tag{7}$$

The memory for SDCHMM consists of the memory for the mixtures and the memory for the codebooks.

Apart from the memory efficiency SDCHMMs are also computationally efficient as a small number of subspace Gaussians log likelihoods can be precomputed only once for each feature vector and stored in RAM. During Viterbi decoding of SDCHMM, the log likelihood can be calculated as the sum of precalculated stream log likelihoods and the log mixture weight.

The required memory for precalculated stream log likelihoods in bytes is defined as

$$\text{mem}_{\text{distances}} = KGM \ . \tag{8}$$

where $K$ denotes the number of streams, $M$ denotes the number of vectors in the codebook, and $G$ denotes the size of the log likelihoods in bytes.

### 3. RESULTS AND DISCUSSION

The memory reduction for speech recognition of German and Spanish languages was experimentally investigated. Three HMMs were tested on different recognition tasks.

The results are shown in Tables 1-3. In the first column there are the dimensions of the streams. Preliminary tests have shown that streams of dimension more than 4 with 256 prototypes in a codebook lead to a big degradation of recognition rate.

In the second column the type of codebook (shared or independent for each stream) is shown. The codebook sizes are 256 vectors for 2-, 3- and 4-D streams, and 16 vectors for 1-D streams.

In the third column the required memory to store the mixtures with codebooks is shown. The last columns show the WER for different recognition tasks. For the baseline (continuous density HMM) the memory size is shown in bytes. For the SDCHMMs the memory size and WER is shown as the ratio compared to the baseline results. For example, 0.261 in the memory size column means that the current SDCHMM requires only 26.1% of baseline memory. 1.301 in the WER column means that WER was increased up to 30.1% relative to the baseline.

**German Generalist** HMM is a context-independent phoneme-based HMM which consists of 85 segments and has 4000 mixtures. The results are shown in Table 2. The following tasks were tested:

- **MoTiV** is a recognition task with a vocabulary of 26 isolated command words, the utterances were recorded in cars, the WER is high because of the noise, the test sequence consists of 2600 utterances.
- **MoTiV NR** is the same task as MoTiV, but with a reduction of noise by means of a spectral subtraction, that is why the recognition accuracy is better.

<table>
<thead>
<tr>
<th>stream dim</th>
<th>codebook type</th>
<th>memory</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline 96000 b</td>
<td>4 shared 0.261</td>
<td>1.301 0.935 0.996</td>
<td></td>
</tr>
<tr>
<td>baseline 96000 b</td>
<td>4 indep. 0.314</td>
<td>1.186 0.962 1.002</td>
<td></td>
</tr>
<tr>
<td>baseline 96000 b</td>
<td>3 shared 0.341</td>
<td>1.019 1.004 1.015</td>
<td></td>
</tr>
<tr>
<td>baseline 96000 b</td>
<td>3 indep. 0.397</td>
<td>1.138 1.012 1.018</td>
<td></td>
</tr>
<tr>
<td>baseline 96000 b</td>
<td>2 shared 0.505</td>
<td>1.029 0.974 0.996</td>
<td></td>
</tr>
<tr>
<td>baseline 96000 b</td>
<td>2 indep. 0.564</td>
<td>0.986 1.005 1.000</td>
<td></td>
</tr>
<tr>
<td>baseline 96000 b</td>
<td>1 shared 0.500</td>
<td>1.033 0.962 0.969</td>
<td></td>
</tr>
</tbody>
</table>

*Table 1: Test results: German Generalist*

**Spanish Generalist** HMM is a context-independent phoneme-based HMM which consists of 85 segments and has 4000 mixtures. The results are shown in Table 2. The following tasks were tested:

- **AppW** is a recognition task with a vocabulary of 32 different isolated application words, the test set consists of 1414 utterances.
- **Spell** is a spelling recognition task, the test sequence consists of 10073 Spanish words spelled letter by letter, the vocabulary is 30 Spanish spelling words.
- **Digits** task has the vocabulary of 10 Spanish digits, the test sequence consists of 483 isolated digits.

<table>
<thead>
<tr>
<th>stream dim</th>
<th>codebook type</th>
<th>memory</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline 96000 b</td>
<td>4 shared 0.261</td>
<td>1.182 1.080 1.250</td>
<td></td>
</tr>
<tr>
<td>baseline 96000 b</td>
<td>4 indep. 0.314</td>
<td>2.000 1.107 0.875</td>
<td></td>
</tr>
<tr>
<td>baseline 96000 b</td>
<td>3 shared 0.341</td>
<td>1.182 1.056 1.000</td>
<td></td>
</tr>
<tr>
<td>baseline 96000 b</td>
<td>3 indep. 0.397</td>
<td>1.273 1.031 1.000</td>
<td></td>
</tr>
<tr>
<td>baseline 96000 b</td>
<td>2 shared 0.505</td>
<td>0.909 1.011 1.125</td>
<td></td>
</tr>
<tr>
<td>baseline 96000 b</td>
<td>2 indep. 0.564</td>
<td>0.909 1.000 1.125</td>
<td></td>
</tr>
<tr>
<td>baseline 96000 b</td>
<td>1 shared 0.500</td>
<td>1.364 1.009 0.875</td>
<td></td>
</tr>
</tbody>
</table>

*Table 2: Test results: Spanish Generalist*

**German Specialist** HMM is a whole-word HMM which consists of 238 segments and has 1199 mixtures. The results are shown in Table 3. The following tasks were tested:

- **SDI-1** and **SDI-2** are connected digits recognition tasks in low-noise conditions, the test set consists of 1433 and 1402 utterances respectively.
- **SDIIm** is a connected digits recognition task, the test set consists of 5329 digits recorded on the GSM channel.
- **SieTill** is an isolated digits recognition task, the test sequence consists of 43204 digits.
The processing power in MIPS are estimated for “worst case” log likelihoods calculation without any optimization where all of the mixture log likelihoods have to be calculated. As can be seen the described approaches lead to a reduction of MIPS from 67 to 19.

Table 4: System requirements

<table>
<thead>
<tr>
<th>parameter</th>
<th>baseline</th>
<th>2-D streams</th>
<th>3-D streams</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM, bytes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mixtures</td>
<td>96000</td>
<td>48000</td>
<td>32000</td>
</tr>
<tr>
<td>mixture log weights</td>
<td>8000</td>
<td>8000</td>
<td>8000</td>
</tr>
<tr>
<td>shared codebook</td>
<td>-</td>
<td>512</td>
<td>768</td>
</tr>
<tr>
<td>total</td>
<td>104000</td>
<td>56512</td>
<td>40768</td>
</tr>
<tr>
<td>RAM, bytes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>log likelihoods</td>
<td>6144</td>
<td>4096</td>
<td></td>
</tr>
<tr>
<td>Processing power, MIPS</td>
<td>67</td>
<td>26</td>
<td>19</td>
</tr>
</tbody>
</table>

Table 3: Test results: German Specialist

As can be observed in Tables 1-3, the recognition performance of SDCHMMs with independent codebooks is almost the same as of SDCHMMs with shared codebooks, but independent codebooks require more memory as shared codebooks. The smaller the HMM the more memory (relatively) is required for separate codebooks. As shown in Table 3 the memory consumption of SDCHMMs with 2-D streams and independent codebooks is 41% higher than that of SDCHMM with 2-D streams and a shared codebook. Also the SDCHMM with 2-D streams and shared codebook occupies less memory (and has better recognition performance) as the SDCHMM with 3-D streams and independent codebooks.

In all SDCHMMs in the tests the codebook size is 256 prototypes. The codebook sizes of 256 vectors were always used, as this leads to an index size of 1 byte for the codebook vectors. The indexes are thus easy to handle. In the case of constant codebook size (256 vectors) the higher the dimension, the higher a quantization error, and thus the higher an increase of the WER can be expected. In Tables 1-3 the relative increase of WERs in 2-D streams SDCHMMs are less than 5%. 3-D streams approach leads to less than 14% increase of WER. The increase of WER of feature parameter tying HMMs (1-D stream SDCHMMs) with 16 codebook vectors is higher than the WER increase of 2-D streams SDCHMMs, as the precision of scalar quantization is less than that of vector quantization.

In Table 2 AppW and Digits tasks have high relative increases of WER, but these results are not reliant as in AppW task only 11 of 1414 words were recognized wrongly, in Digits task only 8 of 483 words were recognized wrongly. The wrong recognition of only one extra word in Digits task leads to increase of WER up to 12.5% relatively.

The speech modeling by SDCHMM can be interpreted as a speech modeling by baseline HMM with some distortion. This distortion can lead to a such SDCHMM which recognition accuracy for a certain tasks is better than of the baseline HMM. The improvement of the recognition accuracy occurs mostly on tasks with high WERs.

Estimation of system resources

The required resources of the described low-cost recognition systems are shown in Table 4. The results are shown for the shared codebook approach. In the case of 2-D streams about 57 Kbytes of fast-acting ROM is required. In the case of 3-D streams about 41 Kbytes are required.

The system requirements for a small-vocabulary recognizer was estimated. They could be reduced up to 19 MIPS for emission calculation and 41 Kbytes of ROM for HMM. These resources are already available on contemporary mobile devices.

5. REFERENCES