ABSTRACT
Several types of Semi-Continuous HMM (SC-HMM) have been compared with the Continuous Density HMM (CD-HMM) in the context of Speaker Independent Isolated Words Recognition (SI-IWR). It is demonstrated that for the ten-digit vocabulary (TIDIGITS), the SC-HMM outperforms the CD-HMM when memory constraints are imposed on the system. SC-HMMs demonstrate recognition rate of about 95% with a total of 59 Gaussians, while under similar conditions the CD-HMM yield recognition rates of well below 80%. An algorithm for optimal selection of Gaussian functions for SC-HMM is presented.

INTRODUCTION
Most (speaker independent) speech recognition systems nowadays, use Continuous Density HMMs [1,2] to model the speech units (words, phonemes, Phonemes-Like Units). The observation probability densities in each state, are approximated by a finite linear combination of Gaussian probability distributions.

In this work two types of HMMs are compared: the Continuous Density HMM (CD-HMM) and the Semi-Continuous HMM (SC-HMM) [3]. The CD-HMM estimates the probability density of a given state by choosing the best Gaussians. The approximation of the probability density function is performed by optimizing the expectations and covariance matrices of the Gaussian functions, as well the linear combination coefficients. The SC-HMM uses a a linear combination of Gaussians chosen from a given bank of common Gaussians that serve all states. Here the probability density function of a state is estimated by optimizing only the linear combination coefficients of the available Gaussians. The bank of Gaussians is globally optimized. Ideally, the CD-HMM is superior to the SC-HMM, since its probability density functions are unconditionally optimized while those of the SC-HMM are constraint to the given bank. In real practice, however, this is not necessarily so.

In real life cases, the amount of data available for training is limited. The estimation of many parameters, as required in the CD-HMM may become a problem. Inaccurate estimation may cause the CD-HMM to be less accurate than the SC-HMM. In many applications, memory size may be a major factor in addition to recognition accuracy. The recognition accuracy and memory requirements of CD-HMM and SC-HMM Speaker Independent Isolated Words Recognition (SI-IWR) systems were evaluated and compared. A performance function involving the ratio of recognition accuracy and required memory size was defined and employed in the evaluation. The SC-HMM was found to be superior to the CD-HMM. For example the SC-HMM required a memory of 6.8KB with accuracy of 98.1%, while the CD-HMM required 10.8KB yielding only 97.7% correct recognition. The algorithms for optimal Gaussian selection and parameters coding as well as the results are discussed in the paper.

HMM CONFIGURATIONS
In this work two types of HMM configurations are compared: the Continuous Density HMM (CD-HMM) and the Semi-Continuous HMM (SC-HMM). In both types of HMM, the pdf of the observation of the jth state, \( p_j(O) \), is described as a linear combination of N Gaussians
where $c_{i,j}$ are the linear coefficients, $N$ are the Gaussians with expectations $\mu_{i,j}$ and covariance matrices $U_{i,j}$. In all experiments, diagonal covariance matrices were used.

Both the CD-HMM and SC-HMM had left-to-right and no skips (LTR-NS) architecture. The CD-HMM was trained by the Baum-Welch algorithm with the following initial conditions procedure: the data was assumed to be uniformly distributed among the states, the K-means algorithm was applied to the data of each state to calculate the optimal Gaussians. Each state was assigned $N_i; \ i=1,2,...,n_s$, Gaussians with $\sum_{i=1}^{n_s} N_i = N$. In the work reported here $N_i = N_j \ \forall i,j$. The initial conditions on the transition matrix were 0.9 on the main diagonal and 0.1 on the upper diagonal. Two types of SC-HMM were tested, they differ in their training initial conditions. Unlike the CD-HMM, here all the data, of each word, was grouped together to estimate (by means of the k-means algorithm) the best N Gaussians of the word. The pdf of each state is described by a linear combination of all N common Gaussians. In the first SC-HMM, the linear combination coefficients for each state are determined as the relative participation of each Gaussian used in the pdf estimation. In the second SC-HMM, denoted here as the Modified SC-HMM (M-SC-HMM), a lower level clipping value was used – a Gaussian whose participation in a certain state, was lower than the clipping level, was eliminated from the description of that state. In both SC-HMM, the transition matrix was initiated as in the CD-HMM case.

**GAUSSIANS MINIMIZATION ALGORITHM**

The number of Gaussians used in a HMM system determines, to a large extent, the accuracy and memory requirements. It is desired to optimize the number of Gaussians in such a way as to minimize memory for a given accuracy.

The basic idea in the minimization algorithm (patent pending) is to start with a large number of Gaussians and then in an iterative manner, choose several “similar” Gaussians, represent them by means of an “equivalent” Gaussian (by means of a merging operator) and check the accuracy of recognition of the reduced system. This process repeats itself until a stage where the merging of the most similar functions causes a significant degradation in accuracy.

The general description of the function minimization algorithm is described below.

1. **Initial estimation.**

   Train all word models by a conventional HMM training algorithm. In the case of CD-HMM, determine the number of Gaussians per state. In the case of SC-HMM determine the total number of Gaussians to be used. All Gaussians used to describe the pdfs of all states constitute the initial Gaussian bank.

2. **Clustering**

   2.1 Calculate the symmetric, weighted Euclidean distortion between any two Gaussians in the bank:

   \[ d_{ij} = d_{ji} = \sum_{n=1}^{p} (\mu_{in} - \mu_{jn})^2 \frac{\sigma^2_{jn}}{\sigma^2_{jn} + \sigma^2_{in}} \]

   where $\mu_{kn}$ is the expectation of the $n$th element of the $k$th Gaussian, $\sigma_{kn}$ is the standard deviation of the $n$th feature and $p$ is the dimensionality of the features space.

   2.2 Set the clustering threshold ($T_{h_C}$) to be:

   \[ T_{h_C} = \left( \text{Min}_{N_i \neq N_j} \{ d_{ij} \} \right) (1 + V_{th}) \]

   Where $V_{th}$ is the threshold variable.

   2.3 Group the Gaussians into clusters $\Omega_j; \ j=1,2,...$. Each cluster contains all the Gaussians, the distance to at least another member is less than $T_{h_C}$.

3. **Merging**

   Merge all Gaussians in each cluster to a single representative Gaussian. The expectation vector and covariance matrix of the Gaussian are the mean expectation and covariance of the Gaussians in the cluster.

4. **Evaluation**

   4.1 Save all words models.

   4.2 Using the training set of the database, re-train all models with the new Gaussian bank.

   4.3 Using the train set of the database, perform a recognition test. Denote the recognition rate (in
percent) of the current test: $R_i$ and that of the last iteration $R_{i-1}$.

4.4 If the degradation in recognition, due to the last merger, is more than 0.5%, (that is to say if $R_{i-1} - R_i > 0.5\%$) then
   a. Set a new value to the threshold variable ($V_{th}$)
      \[ V_{th} \leftarrow V_{th} \times 0.5 \]  
   b. Load the previous models and go to step 2.2.

5. Termination

The algorithm may perform with one of several termination conditions:

Condition 1:
   $R_i$ is lower the a given desired recognition threshold.

Condition 2:
The number of Gaussian functions is below $N_{\text{min}}$ (determined in advance as a limit on memory).

Condition 3:
   Either the recognition rate $R_i$, is below a given threshold, or the number of Gaussian functions is below a given $N_{\text{min}}$.

5.1 If the termination condition is met then terminate the algorithm. Else return to step 2.1.

EXPERIMENTAL RESULTS AND CONCLUSIONS

The TIDIGITS database was used for the evaluation. The database contains 111 male speakers and 114 female speakers. Unified (male and female) speaker-independent models were trained using the training part of the database. The test database included about 2500 utterances of the 11 (including “oh”) digits. Left-to-Right with no skip models (LTR-NS), with 6 to 12 states per word and up to the total (for all words) of about 400 Gaussians have been used. The feature vector consisted of a 20th order mel-scale filter bank cepstrum and del-cepstrum. The database was resampled in 8kHz, in an office environment by playing the original wave files using high-quality speakers, and recorded using a directional microphone (C-400).

Table 1 shows the recognition results with the CD-HMM and the two SC-HMM. In the CD-HMM each state was allocated the same number of Gaussians (“Gaussians per Word” divided by the number of states). In the SC-HMMs all states shared a bank of Gaussians with “Gaussians per Word” Gaussians. Note that here each word was individually optimized, no attempt was made to optimize the complete Gaussian bank.

The first row of table 1 shows the recognition results for models with 6 states and with very low number of Gaussians (one Gaussian per state in CD-HMM). The CD-HMM completely collapses yielding recognition rate of only 9.5%, while the SC-HMM yield above 96% recognition rate. In general, the SC-HMM outperforms the CD-HMM.

The results of Gaussian bank minimization experiment are shown in Figure 1. An 8 states LTR-NS CD-HMM was trained with 24 Gaussians per word. The 264 Gaussian functions were used as initial conditions for the minimization algorithm (denoted “Regular” –triangle points in fig.1). The experiment included also the SC-HMM and the M-SC-HMM described above. Note that in this experiment the number of Gaussians is the total bank for all words: optimization is performed here on the complete vocabulary. The recognition results are the results shown during minimization namely the results of the training set. Figure 1 shows that the regular initial conditions perform well up to about 150 Gaussians, below that the recognition rate drops sharply. The other two initial conditions perform well even with only 50 Gaussians. In order to better compare the results a figure of merit (FoM) was defined:

\[
\text{FoM} = \frac{\text{memory for pdf parameters [KBytes]}}{\% \text{ recognition}}
\]

Table 2 shows the details of the minimization results in four regions: around 260 Gaussians, 150,
77 and 59. The SC-HMM perform well even with 59 Gaussians, with $FoM = 0.032$.

REFERENCES


Fig. 1: Gaussian Minimization (training set) with three types of Initial Conditions

Table 2: Gaussian Minimization Results (test set) for regions: 260, 150, 77 and 59 Gaussians

<table>
<thead>
<tr>
<th>Initial Conditions</th>
<th>Number of Gaussians</th>
<th>Number of C_i Coefficients</th>
<th>% Recognition</th>
<th>Memory [Kbytes]</th>
<th>Figure of Merit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular</td>
<td>264</td>
<td>264</td>
<td>96.81</td>
<td>10.82</td>
<td>0.111</td>
</tr>
<tr>
<td>Semi</td>
<td>264</td>
<td>2112</td>
<td>97.62</td>
<td>12.67</td>
<td>0.130</td>
</tr>
<tr>
<td>M-Semi</td>
<td>235</td>
<td>1538</td>
<td>98.02</td>
<td>10.94</td>
<td>0.112</td>
</tr>
<tr>
<td>Regular</td>
<td>154</td>
<td>240</td>
<td>97.98</td>
<td>6.40</td>
<td>0.065</td>
</tr>
<tr>
<td>Semi</td>
<td>151</td>
<td>1624</td>
<td>97.41</td>
<td>7.66</td>
<td>0.079</td>
</tr>
<tr>
<td>M-Semi</td>
<td>150</td>
<td>1231</td>
<td>97.86</td>
<td>7.23</td>
<td>0.074</td>
</tr>
<tr>
<td>Regular</td>
<td>&lt;80.00</td>
<td></td>
<td>&lt;80.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semi</td>
<td>77</td>
<td>928</td>
<td>96.85</td>
<td>4.00</td>
<td>0.041</td>
</tr>
<tr>
<td>M-Semi</td>
<td>77</td>
<td>901</td>
<td>97.10</td>
<td>3.98</td>
<td>0.041</td>
</tr>
<tr>
<td>Regular</td>
<td>&lt;80.00</td>
<td></td>
<td>&lt;80.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semi</td>
<td>59</td>
<td>688</td>
<td>94.30</td>
<td>3.05</td>
<td>0.032</td>
</tr>
<tr>
<td>M-Semi</td>
<td>59</td>
<td>768</td>
<td>94.96</td>
<td>3.13</td>
<td>0.032</td>
</tr>
</tbody>
</table>