Internal Use Only (非公開)

002

TR-IT-0195

Tied-Mixture Based SSS HMnet Design

ジラルジ アレサンドレ
Alexandre Girardi

シンガー ハラルド
Harald Singer

1997.03

This report describes a new approach to ML-SSS algorithm that uses tied-mixture representation of the output probability density function instead of single Gaussian during the splitting phase of the SSS algorithm (Tied-Mixture SSS or TM-SSS algorithm). Due to this new representation we increase the recognition rate of the original ML-SSS algorithm by better choosing the split state and the split itself. Implementation and results will be shown.

©ＡＴＲ音声翻訳通信研究所

©ATR Interpreting Telecommunications Research Laboratories

目次

1 Introduction

In speech recognition, modeling of accoustic contexts is very important to achieve high accuracy recognition. Due to limited training data, high accuracy recognition may be accomplished by sharing states, tying mixtures and sharing other parameters, in order to model the phonemes robustly and precisely.

Parameter sharing may be implemented in two ways in the HMM training algorithm: using some useful rules automatically (as maximum likelihood state split) or using knowledge interactively. The automatic approach has the advantage that, as the level of detail in the system grows, it is still possible to manage the problem by questioning the rules or adding new rules to the existing ones. The knowledge approach leads to an excessively large number of sharing possibilities resulting in a system too complicated to be improved.

For these reasons, an approach that generates HMM context dependent models has been chosen. In this area ML-SSS algorithm has been proven to outperform other HMM design algorithms [23]. However it still has some weak points that should be explored.

ML-SSS is a divisive clustering algorithm. A network of HMM states is increased iteratively by splitting at each iteration a state either in the contextual or temporal domain. The split state is selected as the state that maximally increases the expected gain in likelihood, which is calculated based on the assumption that all pdf's are single Gaussians. The iteration is stopped when either there are no more states to split, the gain in likelihood is smaller than a preset threshold or the desired number of states has been reached.

The algorithm is very consistent except that the probability density function (pdf) of a state is represented by a single Gaussian (SG) during the split process. Figure 1 shows qualitatively how the pdf of a state would be better represented by either a tied-mixture (TM) or a continuous density (CD) approach. This may also lead to undesired split gain values in ML-SSS, see Figure 2.

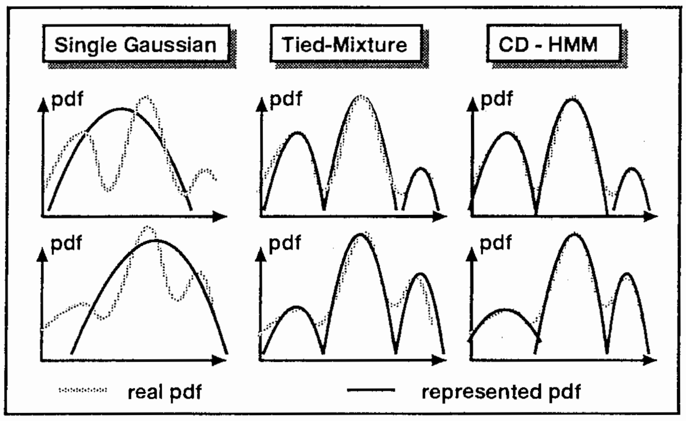

図 1: TM and CD vs SG pdf

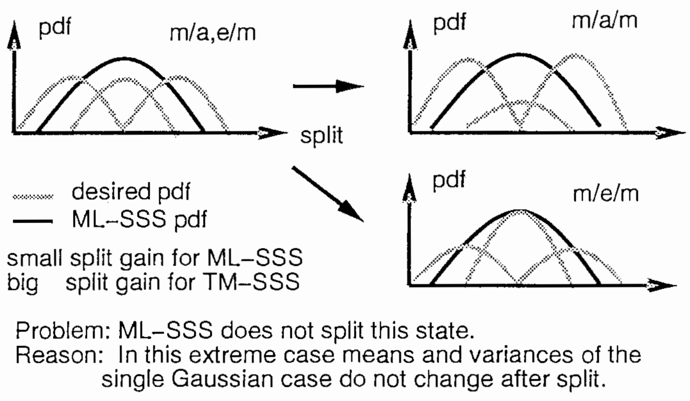
pdf
m/a,e/m
m/a/m
m/e/m
split
desired pdf
ML–SSS pdf
small split gain for ML–SSS
big split gain for TM–SSS
Problem: ML–SSS does not split this state.
Reason: In this extreme case means and variances of the single Gaussian case do not change after split.

図 2: Undesired gain values in ML-SSS

One solution is to represent the pdf by either CD or TM representations. Unfortunately in the CD case the evaluation of the gain in likelihood is computationally too expensive. On the other hand, the TM computational cost is reasonable, once we accept the constraint that the codebook does not change during a split.

This is not a strong constraint as the data affected by a split will contribute minimally to the codebook parameters for a large number of states. The usefulness of the TM pdf representation is the subject of this report.

2 TM-SSS Algorithm

TM-SSS algorithm is in summary a tied-mixture HMM where the topology is trained with a variant of the ML-SSS algorithm. TM-SSS has the benefit of both algorithms in the sense that it keeps a robust representation of the output density probabilities even for a large increase in the number of states in the SSS structure, also maximizing the expected likelihood of the split states as in ML-SSS [22]. This approach also carries the benefits of tied-mixture in the sense that we may avoid the introduction of singularities that may arise in the continuous single mixture HMM implementations [19]. Following is the core of the proposed algorithm.

```
TM-SSS Algorithm

• Preinitialization:
    1. Create a codebook of Gaussians
• Initialization:
    1. Run Baum-Welch ( tied-mixture )
    2. Get split information for all states
• Iterate:
    1. find best split for each domain and factor
    2. split the state with the highest expected likelihood gain
    3. if update codebook
        (a) run BW, train means, variances, weights and transitions over all states
        (b) use only the most significant mixtures
       else
        (a) run BW over affected states, only train weights and transitions
```

2.1 TM-SSS Preinitialization

In addition to the general algorithm described in Section 2 above, two new steps are performed at the first time one runs the algorithm.

1. Run Baum-Welch with N Gaussian mixtures for each center phoneme,

2. Using the above Gaussians to create a tied-mixture HMM

This initialization not only gives us a good initialization of the algorithm, but additionally gives us a good starting point for comparison with the ML-SSS algorithm.

2.2 TM-SSS Split Information

Initially define s^* as the state where the split test is applied and s_0 and s_1 as the two new hypothetic states generated from the split. In such a case the split test of a state s^* is carried out for both context and time domains.

In the context domain, a split test is performed individually for each context c with $c \in \{preceding, center, suceeding\} = \{p_d, c_d, s_d\}$ phoneme categories. The split in a specific context generates two new sets of phonemes $P(s_0, c)$ and $P(s_1, c)$ originated from the original set of phonemes $P(s^*, c)$, with the gain $G(s^*, c)$.

In the time domain t_d, the split test generate two new hypothetic states with the same sets of phonemes, i.e.,

$$\begin{aligned}
P(s_0, p_d) &= P(s^*, p_d) \\
P(s_0, c_d) &= P(s_0, s_d) = P(s_1, p_d) = P(s_1, c_d) = P(s^*, c_d) \\
P(s_1, s_d) &= P(s^*, s_d)
\end{aligned} \tag{1}$$

with the gain $G(s^*, t_d)$.

The split information is therefore composed by three informations:

- split factor s_f that gives the maximum gain $G(s^*, f)$, where s_f and $f \in \{t_d, p_d, c_d, s_d\}$,
- maximum gain $G(s^*, s_f) = max\{G(s^*, f)\}$ and
- split set of phonemes $P(s_0) = P(s_0, s_f)$ and $P(s_1) = P(s_1, s_f)$.

The next three sections describe in detail how to calculate the gain of a split and how to split in both the context and time domains.

(2.2.1) TM-SSS Split Gain

The split gain in TM-SSS is defined as the gain in likelihood of the observed data when a state is split. In order to calculate this gain we choose Maximum Likelihood (ML) as criterion.

ML tries to approximate the increase in recognition accuracy by maximizing the likelihood of the observation data.

ML-SSS is one approach similar to TM-SSS that is based on the ML criterion. In ML-SSS it is necessary to store state occupancy counts and other related values during the training phase [22]. Knowing that the number of these values grows with the square of the length of the observed data, we choose the following constraint:

Constraint: All likelihood computations assume fixed phone boundaries.

Alternatives to this constraint are embedded reestimation and Viterbi-style training algorithm. Embedded reestimation seems to be a good approach, but needs additional assumptions/constraints that are not the immediate purpose of this work. Viterbi-style training is an attractive alternative, but it is not used here, because it is known to be sub-optimal relative to Baum-Welch training.

The calculation of likelihood over the observation data directly implies many multiplications (CPU intensive), so the same approach as in the Expectation-Maximization (EM) algorithm [7] is used, i.e. instead of maximizing the likelihood of the observed data directly, the expected log likelihood of the observed data will be maximized.

To clarify let us define the observed data y_u^T as $y_u^T = \{y_1, y_2, \ldots, y_f, \ldots, y_T\}$ and the related hidden or unobserved components s_u^T as $s_u^T = \{s_1, s_2, \ldots, s_f, \ldots, s_T\}$, where

- y_t is generated by s_t,
- f is a frame index,
- u is an utterance index,
- T the total number of frames in an utterance and
- $s_t \in S = \{s(1), \ldots, s(N)\}$ is one of the states in the HMM where
- N represent the actual total number of states in the HMM.

Finally let us define $Q(\theta^{(p+1)}|\theta^{(p)})$ as the expected log likelihood of the observed data and the hidden states as follows

$$Q(\theta^{(p+1)}|\theta^{(p)}) = E_{\theta^{(p)}}[\log \mathrm{P}(y_1^T, s_1^T | y_1^T, \theta^{(p+1)})] \tag{2}$$

So maximizing $Q(\theta^{(p+1)}|\theta^{(p)})$ at worst gives no change to the likelihood of the observed data $L(\theta) = \log \mathrm{P}(y_1^T|\theta)$, mathematically

$$Q(\theta^{(p+1)}|\theta^{(p)}) \geq Q(\theta^{(p)}|\theta^{(p)}) \quad \Longrightarrow \quad L(\theta^{(p+1)}) \geq L(\theta^{(p)}). \tag{3}$$

Equation 3 allows to change the objective function from expected log likelihood to expected likelihood, which is computationally tractable. The problem now is to express $Q(\theta^{(p+1)}|\theta^{(p)})$ in terms of the sufficient statistics obtained in the split phase.

Let us then expand the expected log likelihood $Q(\theta^{(p+1)}|\theta^{(p)})$ using the conditional independence assumptions in the HMM

$$\begin{aligned}
Q(\theta^{(p+1)}|\theta^{(p)}) &= E[\log \mathrm{P}(y_1^T, s_1^T|y_1^T, \theta^{(p+1)})|\theta^{(p)}] = \sum_{s_1^T} \mathrm{P}(s_1^T|y_1^T, \theta^{(p)}) \log \mathrm{P}(y_1^T, s_1^T|\theta^{(p+1)}) \\
&= \sum_{s_1^T} \mathrm{P}(s_1^T|y_1^T, \theta^{(p)}) \left[\log \mathrm{P}(s_1^T|s_1^T, \theta^{(p+1)}) + \log \mathrm{P}(y_1^T|s_1^T, \theta^{(p+1)})\right] \\
&= \sum_{s_1^T} \mathrm{P}(s_1^T|y_1^T, \theta^{(p)}) \left[\log \mathrm{P}(s_1^T|s_1^T, {\theta^{(p+1)}}_A) + \log \mathrm{P}(y_1^T|s_1^T, {\theta^{(p+1)}}_B)\right] \\
&= \sum_{s:s=s_t, s'=s_{t-1}}^{S} \sum_t \xi_t(s,s') \log \mathrm{P}(s_t|s_{t-1}, {\theta^{(p+1)}}_A(s')) \\
&\quad + \sum_{s:s=s_t}^{S} \sum_t \sum_l \gamma_t(s,l) \log \mathrm{P}(y_t \sim v_l|s_t, {\theta^{(p+1)}}_{B_l}(s)) \\
&\quad + \sum_{s:s=s_t}^{S} \sum_t \sum_l \gamma_t(s,l) \log \mathrm{P}(y_t|y_t \sim v_l, {\theta^{(p+1)}}_{v_l}(s))
\end{aligned} \tag{4}$$

where

$$\begin{aligned}
\gamma_t(s,l) &= \mathrm{P}(s_t = s, x_t \sim v_l | y_1^T, \theta^{(p)}) && (5) \\
\xi_t(s,s') &= \mathrm{P}(s_t = s, s_{t-1} = s' | y_1^T, \theta^{(p)}) && (6) \\
v_l &= \{\mu_l, \Sigma_l\}. && (7)
\end{aligned}$$

Now the transition parameters $\theta_{A(s)}$, the distribution probabilities $\theta_{B(s)}$ and the mean μ_l and variance Σ_l for state s are separated.

Equation 4 allows separate estimation of the likelihood for both temporal or contextual split of a state, as well as for separate estimation of mean μ_l and variance Σ_l. Rearranging the terms and using a shorter notation, it follows that

$$\begin{aligned}
= & \sum_{s:s=s_t, s'=s_{t-1}}^{S} \sum_t \xi_t(s,s') \log \mathrm{P}(s_t|s_{t-1}, {\theta^{(p+1)}}_A(s')) + \sum_{s:s=s_t}^{S} \sum_t \sum_l \gamma_t(s,l) \log \mathrm{P}(y_t \sim v_l|s_t, {\theta^{(p+1)}}_{B_l}(s)) \\
& + \sum_{s:s=s_t}^{S} \sum_t \sum_l \gamma_t(s,l) \log \mathrm{P}(y_t|y_t \sim v_l, {\theta^{(p)}}_{v_l}(s))
\end{aligned}$$

$$= \sum_{s:s=s_t,s'=s_{t-1}}^{S} \sum_t \xi_t(s,s') \log a(s,s') + \sum_{s:s=s_t}^{S} \sum_t \sum_l \gamma_t(s,l) \log b_l(s)$$
$$+ \sum_{s:s=s_t}^{S} \sum_t \sum_l \gamma_t(s,l) \log N(y_t|\mu_l,\Sigma_l) \tag{8}$$

where $s \in S = s(1),\ldots,s(N)$ is one of the states in the HMM, N represent the actual total number of states in the HMM and

$$a(s,s') = \mathrm{P}(s_t = s|s_{t-1} = s', \theta_{A(s)}) = a_{ij} \tag{9}$$
$$b_l(s) = \mathrm{P}(s_t = s|\theta_{B_l(s)}) = b_i(l) \tag{10}$$
$$N(y_t|\mu_l,\Sigma_l) = \frac{1}{(2\pi)^{M/2}|\Sigma_l|^{1/2}} \exp\left[-\frac{1}{2}(y_t-\mu_l)\Sigma_l^{-1}(y_t-\mu_l)\right] \tag{11}$$

Equation 8 is considering a specific context c, i.e., one of the values on $c \in \{preceeding, center, suceeding\}$ $= \{p_d, c_d, s_d\}$. This grouping has also the benefit to reduce the memory storage for the γ and ξ values that appear in Equation 4, since they are summed up for each phone $j(s,c)$ in a specific context c for state s, where $j(c)$ takes values on $P(s,c)$. Note that for the special case of temporal domain split or the split state (not the hypothetical new states) we may assign $j^* = j(c_d)$. So, γ is then represented by three vectors on c as follows

$$\gamma_{j(c)}(s,l) = \sum_{t:x_t=x_{j(c)}} \gamma_t(s,l) \tag{12}$$
$$\xi_{j(c)}(s,s') = \sum_{t:x_t=x_{j(c)}} \xi_t(s,s') \tag{13}$$

So, using Equations 13 on Equation 4, gives

$$Q(\theta^{(p+1)}|\theta^{(p)}) = \sum_{s:s=s_t,s'=s_{t-1}}^{S} \sum_{j(c)\in P(s,c)} \xi_{j(c)}(s,s') \log a(s,s') + \sum_{s:s=s_t}^{S} \sum_{j(c)\in P(s,c)} \sum_l \gamma_{j(c)}(s,l) \log b_l(s)$$
$$+ \sum_{s:s=s_t}^{S} \sum_{t:j(c)\in P(s,c)} \sum_l \gamma_t(s,l) \log N(y_t|\mu_l,\Sigma_l)$$
$$= \sum_{s:s=s_t,s'=s_{t-1}}^{S} \sum_{j(c)\in P(s,c)} \xi_{j(c)}(s,s') \log a(s,s',c) + \sum_{s:s=s_t}^{S} \sum_{j(c)\in P(s,c)} \sum_l \gamma_{j(c)}(s,l) \log b_l(s)$$
$$+ \sum_{s:s=s_t}^{S} \sum_{j(c)\in P(s,c)} \sum_{t:x_t=x_{j(c)}} \sum_l \gamma_t(s,l) \log N(y_t|\mu_l,\Sigma_l) \tag{14}$$

where $a(s,s',c)$, $b_l(s,c)$, μ_l and Σ_l are estimated as

$$a(s,s',c) = \frac{\sum_{j(c)\in P(s,c)} \xi_{j(c)}(s,s')}{\sum_{j(c)\in P(s,c)} \gamma_{j(c)}(s)} \tag{15}$$
$$b_l(s,c) = \frac{\sum_{j(c)\in P(s,c)} \gamma_{j(c)}(s,l)}{\sum_{j(c)\in P(s,c)} \gamma_{j(c)}(s)} \tag{16}$$
$$\mu_l = \frac{1}{\vartheta(l)} \sum_{s\in S} \sum_{j(c)\in P(s,c)} [\vartheta_t(l) y_t] \tag{17}$$

$$\Sigma_l = \frac{1}{\vartheta(l)} \sum_{s \in S} \sum_{j(c) \in P(s,c)} [\vartheta_t(l)(y_t - \mu_l)(y_t - \mu_l)^t] \tag{18}$$

$$\tag{19}$$

where

$$\vartheta(l) = \sum_{s \in S} \sum_{j(c) \in P(s,c)} \vartheta_t(l) \tag{20}$$

$$\vartheta_t(l) = \mathrm{P}(y_t \sim v_l | y_1^T, \theta^{(p)}) = \sum_{s \in S} \gamma_t(s,l) \tag{21}$$

$$\gamma_{j(c)}(s) = \sum_l \gamma_{j(c)}(s,l) \tag{22}$$

As $a(s,s',c)$ do not depend on $j(c)$, Equation 14 simplifies to

$$\begin{aligned} Q(\theta^{(p+1)}|\theta^{(p)}) &= \sum_{s:s=s_t, s'=s_{t-1}}^{S} \sum_{j(c)} \xi_{j(c)}(s,s') \log a(s,s',c) + \sum_{s:s=s_t}^{S} \sum_{j(c)} \sum_l \gamma_{j(c)}(s,l) \log b_l(s) \\ &+ \sum_{s:s=s_t}^{S} \sum_{j(c)} \sum_{t:x_t=x_{j(c)}} \sum_l \gamma_t(s,l) \log N(y_t|\mu_l,\Sigma_l) \\ &= \sum_{s,s'} N_1(s,s',c) \log a(s,s',c) + \sum_s \sum_l N_2(s,c,l) log b_l(s) + \sum_s \sum_{j(c)} N_3(s,j(c)) \end{aligned} \tag{23}$$

where

$$N_1(s,s',c) = \sum_{j(c)} \xi_{j(c)}(s,s') \tag{24}$$

$$N_2(s,c,l) = \sum_{j(c)} \sum_{t:x_t=x_{j(c)}} \gamma_t(s,l) \tag{25}$$

$$N_3(s,j(c)) = \sum_{t:x_t=x_{j(c)}} \sum_l \gamma_t(s,l) \log N(y_t|\mu_l,\Sigma_l) \tag{26}$$

where

$$\xi_{j(c)}(s,s',l) = \sum_{t:x_t=x_{j(c)}} \xi_t(s,s',l) \tag{27}$$

$$\xi_t(s,s',l) = p(s_t = s, s_{t-1} = s', y_t \sim v_l | y_1^T, \theta^{(p)}) \tag{28}$$

$$\gamma_{j(c)}(s,l) = \sum_{t:x_t=x_{j(c)}} \gamma_t(s,l) \tag{29}$$

$$\gamma_t(s,l) = p(s_t = s, y_t \sim v_l | y_1^T, \theta^{(p)}) \tag{30}$$

$$\tag{31}$$

and $y_t \sim v_l$ means that y_t is quantized to v_l.

Computationally the stored values are $\gamma_{j(c)}(s,l)$, where l represent the quantization index of the VQ codebook.

Note that $N_2(s,c,l)$ in practice are computed once for each particular split in a particular fixed context c. So far, as a first result we got a way to evaluate the likelihood of the observed data using Equation 23 as follows

$$\begin{aligned} Q(\theta|\theta^{(p)}) &= E[\log \mathrm{P}(y_1^T, s_1^T|\theta)|y_1^T, \theta^{(p)}] \\ &= \sum_{s,s'} N_1(s,s',c)\log a(s,s',c) + \sum_{s}\sum_{l} N_2(s,c,l) log b_l(s) + \sum_{s}\sum_{j(c)} N_3(s,j(c)) \end{aligned} \quad (32)$$

Once we are splitting one state at a time and $\gamma_t(s,l)$ and $\xi_t(s,s',l)$ are fixed for all $s \neq s^*$, then $Q(\theta|\theta^{(p)})$ is constant for all states less that contain the split state s^*. So the gain in splitting a state $G(s^*)$ may be calculated as a difference in $Q(\theta|\theta^{(p)})$ for just the s that is being split less the gain of the corresponding resulting split states, namely s_0 and s_1, that is

$$\begin{aligned} G(s^*) &= \sum_{s,s'} N_1(s,s')\log a(s,s') - N_1(s^*,s^*)\log a(s^*,s^*) \\ &+ \sum_{l}\left[\sum_{s} N_2(s,c,l)\log b_l(s) - N_2(s^*,c_d,l)\log b_l(s^*)\right] \\ &+ \sum_{j(c)} [N_3(s_0,j(c)) + N_3(s_1,j(c)) - N_3(s^*,j(c))] \end{aligned} \quad (33)$$

Note that the gain $G(s^*)$ does not change due to $N_3(s,c)$, once v_l is supposed to be constant over the contextual and temporal domain splits.

So for the contextual split in the context c the gain is then expressed in terms of the new states s_0 and s_1 as

$$\begin{aligned} G(s^*,c) &= \sum_{k=0,1}\sum_{s_k,s_k} N_1(s_k,s_k)\log a(s_k,s_k) - N_1(s^*,s^*)\log a(s^*,s^*) \\ &+ \sum_{k=0,1}\sum_{s_k,s'} N_1(s_k,s')\log a(s_k,s') - N_1(s^*,s')\log a(s^*,s') \\ &+ \sum_{l}[N_2(s_0,c,l)\log b_l(s_0) + N_2(s_1,c,l)\log b_l(s_1) - N_2(s^*,c_d,l)\log b_l(s^*)] \end{aligned} \quad (34)$$

For the temporal split the gain is expressed as

$$\begin{aligned} G(s^*,td) &= N_1(s_0,s_0)\log a(s_0,s_0) + N_1(s_0,s_1)\log a(s_0,s_1) \\ &+ N_1(s_1,s_1)\log a(s_1,s_1) - N_1(s^*,s^*)\log a(s^*,s^*) \\ &+ \sum_{l}[N_2(s_0,c_d,l)\log b_l(s_0) + N_2(s_1,c_d,l)\log b_l(s_1) - N_2(s^*,c_d,l)\log b_l(s^*)] \end{aligned} \quad (35)$$

Note that $N_2(s_0,c_d,l)$, $N_2(s_1,c_d,l)$ and $N_2(s^*,c_d,l)$ don't have the same value, once $\gamma_{j^*}(s)$ will now be up-dated for each s. In other words γ is a function of $j(c)$ in the contextual domain (fixed for s^*, s_0 and s_1 on each element of $j(c)$), but in the temporal domain it depends on s^*, s_0 and s_1 and not on $j(c) = j(c_d) = j^*$ which is constant.

The gain in Equation 35 guarantees that the likelihood of the observed data is increased or at least kept at the same level if the split state chosen is the one with the greatest gain $G(s)$ in a specific domain (context or temporal splitting). This does not assure an increase in likelihood, as the gain $G(s)$ refers to expected likelihood instead of likelihood as pointed out in Equation 3. The constraint of fixed boundaries may be relaxed if an embedded training was performed, i.e. to align the training data to states in some pre-specified topology (driven by the labels) and then cluster the resulting state distributions to maximize the joint likelihood of the data and the given state sequence.

3 Contextual Split

In the contextual split as used in TM-SSS the aim is to split a state s^* in two new states s_0 and s_1 based on the observation data y associated to the split state s^*.

Note that an observation y may pass many states. In fact we assume we can cluster only the data y_t that passes through a specific state. The differences that may arise are minimized using a Baum-Welch-style clustering and a Baum-Welch reestimation of affected states after the split.

Each observation is described by a set of factors, i.e., its preceding, center and succeeding phonemes. So, we must group the observations according to its factors which correspond to a context domain c with $c \in \{preceeding,\ center,\ suceeding\} = \{p_d, c_d, s_d\}$. In such a case a context split is performed on a context domain at a time, in order to keep the convexity of the state split space A_{s^*}. In the same way we define the space of observations to the new states s_k as A_k, where $k = 0, 1$ and $A_{s^*} = A_0 \cup A_1$.

Additionally let us group the observation data y_t by phonemes $j(c)$ in a certain context c, namely $y_{j(c)}$. By definition $j(c)$ is generated according to the split state s^*, regardless if it will be used by any of the hypothetical new states s_0, s_1. To take care of the different phonemes that will pertain to a certain state s we define $P(s, c)$ as the set of phonemes $j(c)$ that pass in a certain state s.

As in the TM-SSS the purpose is to represent the HMnet using tied-mixtures, the split state s will be represented by its output distribution vector $b(s) = (b_1(s), \ldots, b_l(s), \ldots, b_L(s))$, where l represents one of the codebook index of the VQ.

The goal is then to estimate two new vectors of output distribution coefficients $b(s_0), b(s_1)$, so that the maximum likelihood criteria is observed. This is basically a divisive clustering problem. To solve this problem we use Chou's partitioning algorithm [6] as in ML-SSS [22]. That is, we can design a function $\hat{y} = f(x)$ that predicts the observation data y from the input data x. So the function f may be used to estimate the probability distribution $\hat{p}(y|x) = p(y|f(x))$, as in the tree language model [3]. The tied-distribution estimate interpretation corresponds to the use of divisive distribution clustering in speech recognition, e.g. [15, 27], and so decision tree design methodology applies here.

Chou's partitioning algorithm is applied here, since it is linear on the number of clustered phonemes $j(c)$ and on the number of dimensions of the observed data M and therefore faster compared to similar algorithms (CART [5]).

According to Chou's theorem [6] in the maximum likelihood sense, the loss function $\mathcal{L}(y, b)$ is $-\log \mathrm{P}(y|b)$. Under this objective, the "centroid" function becomes

$$\begin{aligned} b(s) &= \operatorname*{argmin}_{b(s)} E[\mathcal{L}(y, b(s))|s] = \operatorname*{argmax}_{b(s)} E[\log \mathrm{P}(y|b(s))|s] \\ &= \operatorname*{argmax}_{b(s)} \sum_{j \in P(s,c)} \log \mathrm{P}(y_{j(c)}|b(s)) \end{aligned} \tag{36}$$

which is the output weight estimate vector of $b_l(s)$ given by the reestimation formulas, since it must be estimated from data and the true $\mathrm{P}(y|s)$ is unknown. The divergence equation becomes

$$\begin{aligned} d(s,b) &= E[\mathcal{L}(y, b(s))|s] - i(s) \\ &= -\left[\sum_{j(c) \in P(s_0,c)} \sum_{t: x_t \in x_{j(c)}} \log \mathrm{P}(y_t|b(s_0)) + \sum_{j(c) \in P(s_1,c)} \sum_{t: x_t \in x_{j(c)}} \log \mathrm{P}(y_t|b(s_1)) \right] \\ &+ \sum_{j(c) \in P(s^*,cd)} \sum_{t: x_t \in x_{j(c)}} \log \mathrm{P}(y_t|b(s^*)) \end{aligned} \tag{37}$$

For the Baum-Welch style training the divergence Equation 37 changes to

$$d(s,b) = E[\mathcal{L}(y, b(s))|s] - i(s)$$

$$= -\left[\sum_{j(c)\in P(s_0,c)}\sum_{t:x_t\in x_{j(c)}}\sum_l \gamma_t(s^*,l)\log \mathrm{P}(y_t|b_l(s_0))\right.$$
$$\left.+\sum_{j(c)\in P(s_1,c)}\sum_{t:x_t\in x_{j(c)}}\sum_l \gamma_t(s^*,l)\log \mathrm{P}(y_t|b_l(s_1))\right]$$
$$+\sum_{j(c)\in P(s^*,cd)}\sum_{t:x_t\in x_{j(c)}}\sum_l \gamma_t(s^*,l)\log \mathrm{P}(y_t|b_l(s^*)) \qquad (38)$$

which corresponds to the sum of the distribution probabilities $\theta_{B_l(s)}$ terms of Equation 14 where the expected log likelihood of the observed data $Q(\theta^{(p)}|\theta^{(p)})$ for interaction (p) was used instead of $(p+1)$. That's why the estimated frequency occurrence of the l codebook $\gamma_t(s^*,l)$ at the split state s^* was used instead of the estimated frequency occurrence for the hypothetical new states $s_k, k = 0,1$ (i.e., $\gamma_t(s_k,l)$ in the first two terms of Equation 38.

Not considering the minor effects that mean μ_l and variance Σ_l have on the distance, once only part of the data necessary to update them will be used the divergence equation and using $b_l(s) = \mathrm{P}(y_t \sim v_l|s, b_l(s))$ becomes

$$d(s,b) = E[\mathcal{L}(y,b(s))|s] - i(s)$$
$$= -\left[\sum_{j(c)\in P(s_0,c)}\sum_{t:x_t\in x_{j(c)}}\sum_l \gamma_t(s^*,l)\log b_l(s_0) + \sum_{j(c)\in P(s_1,c)}\sum_{t:x_t\in x_{j(c)}}\sum_l \gamma_t(s^*,l)\log b_l(s_1)\right]$$
$$+\sum_{j(c)\in P(s^*,cd)}\sum_{t:x_t\in x_{j(c)}}\sum_l \gamma_t(s^*,l)\log b_l(s^*) \qquad (39)$$

Binary split design of a state s in the context $c = \{preceeding, center, suceeding\} = \{p_d, c_d, s_d\}$, proceeds as follows.

Maximum Likelihood Split Design Algorithm

- **Iterate** for a context $c = p_d, c_d, s_d$
  1. Split state s in the context c
  2. Calculate the gain $G(s)$
  3. If it is the greatest $G(s)$
     (a) Save the context
     (b) Save the splitting information (A_0, A_1)
- Get the gain for a temporal splitting

Note that the algorithm above is repeated for every affected state in step 3 of the TM-SSS algorithm. The information saved in this step is used to find the state to split and to split it in one specific domain and/or context, based on the greatest gain stored for each state in the HMnet. Following is the splitting algorithm for a state s^* in the specific context c, i.e. step 1 of the above algorithm.

Split Design Algorithm in a Context c

(All the terms below depend on c, but c is fixed here so it will be omitted)

- **Initialization** ($p = 0$) Initialize the distribution parameter centroids for the two new hypothetic states:

$$b^{(0)}(s_0) = [b_1(s_0), ...b_l(s_0), ...b_L(s_0)] = [b_1(s^*), ...b_l(s^*), ...b_L(s^*)] = b(s^*)$$

$$b^{(0)}(s_1) = [b_1(s_1), ...b_l(s_1), ...b_L(s_1)] \qquad b_l(s_1) = (1 + (-1)^l \epsilon) b_l(s^*) \qquad \forall l$$

- **Iterate** for $p = 1, 2, \ldots$

1. Find new binary partition $\{A_0^{(p)}, A_1^{(p)}\}$:
   For each $j : j \in P(s)$, assign all the $x_t : x_t \in x_j$ to $A_0^{(p)}$ if

$$\sum_l \gamma_j(s^*, l) \log b_l^{(p-1)}(s_0) \geq \sum_l \gamma_j(s^*, l) \log b_l^{(p-1)}(s_1) \qquad (40)$$

   otherwise, assign all the x_t to $A_1^{(p)}$.

2. Find centroids $\{b^{(p)}(s_k) : k = 0, 1\}$.

$$b_l^{(p)}(s_k) = \frac{\sum_{j \in P(s_k)^{(p)}} \gamma_j(s, l)}{\sum_{j \in P(s_k)^{(p)}} \gamma_j(s)} \qquad (41)$$

   where $\gamma_j(s, l)$ and $\gamma_j(s)$ comes from the reestimation formulas.

3. Test for convergence: stop if the partition does not change or if

$$\frac{G(s^*)^{(p)} - G(s^*)^{(p-1)}}{G(s^*)^{(p-1)}} < \eta$$

   where $G(s^*)$ is the contextual gain (Equation 34) and η is an heuristically chosen convergence threshold. Note that $G(s^*)^{(p)} \geq G(s^*)^{(p-1)}$.

The particular choice of the distribution parameter centroids in the initialization step of the above algorithm ensures that likelihood will not decrease since one of the states has the original state distribution parameters, analogous to the approach used in vector quantizer design.

The $\gamma_j(s, l)$ terms must be computed using both a forward and backward pass for each context c, in other words $\gamma_{j(c)}(s, l)$. This information is in principle available from the Baum-Welch iteration and must be stored in order to calculate the moments above in TM-SSS algorithm.

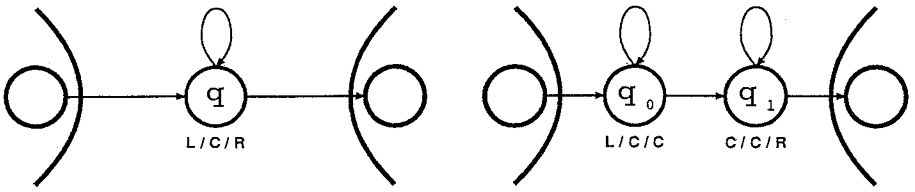

図 3: Temporal split of q into q_0 and q_1.

4 Temporal Split

In Section (2.2.1) a constrained EM approach to determine the split state was used, as it can be assumed that a state split does not change the counts for the two new states.

However, replacing one state for two states in sequence, the increase in expected likelihood is not a simple difference in expected likelihood, as their counts are affected by the split.

As in ML-SSS [22] we make use of a constrained EM criterion in the design of temporal splits, with the constraints being that the likelihoods of states other than the split state do not change in the parameter estimation stage of the split design. Here too, it is not a problem once later all affected states are updated in the Baum-Welch that follows the state split.

To be more explicit, let s^* be the split state and let q_0 and q_1 be the two states resulting from a temporal split, as illustrated in Figure 3. (We use the notation q for the hypothetical new states and s^* for the candidate state to be split.)

The parameters that must be estimated to describe the new state are $\theta = \{b_l(q_0), b_l(q_1)\}$, where $b_l(q)$ is the weight of the probability density functions of the tied-mixture and l represent the codebook index of the tied-mixture. In order to insure that only these parameters in the HMnet change and no others do, we require the following constraints:

$$\begin{aligned}
\gamma_t(s^*) &= \sum_l (\gamma_t(q_0, l) + \gamma_t(q_1, l)) \\
\xi_t(s^*, s^*) &= \xi_t(q_0, q_0) + \xi_t(q_1, q_0) + \xi_t(q_1, q_1);
\end{aligned}$$

where

$$\begin{aligned}
\gamma_t(i, l) &= p(s_t = i, y_t \sim v_l | \mathcal{Y}) \\
\xi_t(i, j) &= p(s_t = i, s_{t-1} = j | \mathcal{Y})
\end{aligned}$$

are the standard terms needed for HMM re-estimation, $y_t \sim v_l$ means that y_t was generated by v_l and $\mathcal{Y}$ represents the full training set.

These constraints can be easily satisfied by defining

$$\begin{aligned}
\tilde{\gamma}_t(q) &= p(q_t = q | s_t = s^*, \mathcal{Y}) \\
\tilde{\xi}_t(q, q') &= p(q_t = q, q_{t-1} = q' | s_t = s^*, s_{t-1} = s^*, \mathcal{Y})
\end{aligned}$$

and using the definition of conditional probability and the redundancy of $s_t = s^*$ to get

$$\begin{aligned}
\gamma_t(q, l) &= p(q_t = q, y_t \sim v_l | \mathcal{Y}) = p(q_t = q, s_t = s^*, y_t \sim v_l | \mathcal{Y}) = \tilde{\gamma}_t(q, l)\gamma_t(s^*) \\
\xi_t(q, q') &= p(q_t = q, q_{t-1} = q' | \mathcal{Y}) = p(q_t = q, q_{t-1} = q', s_t = s^*, s_{t-1} = s^* | \mathcal{Y}) \\
&= \tilde{\xi}_t(q, q')\xi_t(s^*, s^*).
\end{aligned}$$

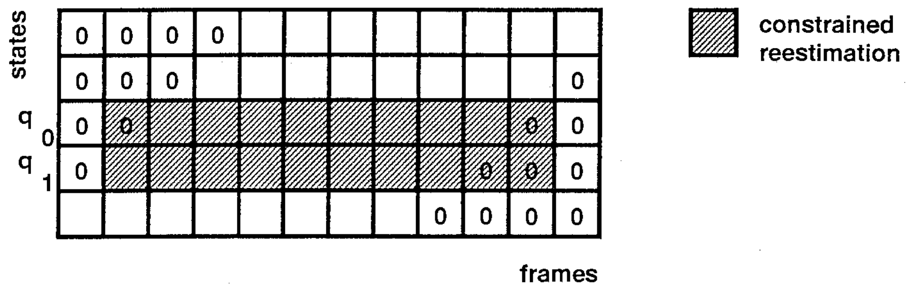

図 4: Illustration of data and states used in computing $\tilde{\gamma}_t(q)$ $\tilde{\xi}_t(q,q')$ for a temporal split. The zeroes indicate impossible state-observation pairs.

The terms $\tilde{\gamma}_t(q,l)$ and $\tilde{\xi}_t(q,q')$ can be computed using the standard forward-backward algorithm using only data where $\gamma_t(s^*) > 0$ and having non-zero state likelihood only for states q_0 and q_1 so that $\sum_l \tilde{\gamma}_t(q_0,l) + \tilde{\gamma}_t(q_1,l) = 1$. The constrained forward-backward is accomplished by passing a subset of the full data only over the two new hypothetical states as illustrated by the shaded region in Figure 4.

Once the terms $\tilde{\gamma}_t(q,l)$ are computed, the parameters θ are estimated according to

$$b_l(q) = \frac{\sum_t \tilde{\gamma}_t(q,l)\gamma_t(s^*)}{\sum_t \sum_l \tilde{\gamma}_t(q,l)\gamma_t(s^*)} \tag{42}$$

The other parameters are straightforward.

Four iterations of the Baum-Welch algorithm are used to compute $\tilde{\gamma}_t(q,l)$ and $\tilde{\xi}_t(q,q')$. The initial estimate for the transition probabilities use the observation distribution of the original state and choose the transition probabilities such that the expected duration of the two hypothesized states together is the same as the expected duration of the original state . This procedure is similar to the one in ML-SSS [22] and give us no garantee of decrease in likelihood.

5 TM-SSS Experiments

Phoneme classification experiments using 25 phonemes for one male speaker (speaker MHT of the Aset of ATR's speech database [28]) were performed with the aim of comparing the performance of TM-SSS and ML-SSS.

5.1 Train/Test Data and Preprocessing

The training data is the set of phonemes taken from the even-numbered words and the test data is the set of phonemes taken from the odd-numbered words of the 5240 words for speaker MHT. Table 1 describes the preprocessing conditions.

表 1: Preprocessing conditions

Parameter	Value
sample rate	12000 Hz
frame shift	5 ms
frame length	20 ms
pre-emphasis coef.	0.98
parameters	16 lpc cepstra, 16 Δ lpc cepstra, 1 log power, 1 Δ log power
frequency warping	none, i.e. linear
total dimension	34

5.2 Experimental Setup and Scoring

The training conditions for both ML-SSS and TM-SSS are as follows:

- HMnet topology training starts initially from 3 states that are aligned left to right, i.e. a 3 state model for all phonemes.
- Only contextual split is performed.
- VQ code size of TM-SSS is kept fixed at 256 Gaussians.
- Topology growing for HMnet is performed up to 400 states.

For scoring, phoneme classification was performed using the preceding and succeeding contexts (see Figure 5). Each phoneme test sample is scored against all those HMM allophone models in which the preceding and succeeding context match the preceding and succeeding context of the test sample. This works as if a perfect language model had chosen the model to test the phoneme sample.

5.3 Results and Discussion

Recognition results for both TM-SSS and ML-SSS algorithms, are summarized in Table 2 for closed (training data) and open (test data) conditions.

For the same number of Gaussians, the results using TM-SSS are about 10% (relative decrease in error rate) better than using the ML-SSS algorithm. An appropriate point for comparison is when the number of states is 256, because then the number of gaussians in both algorithms is identical.

However, the present implementation of TM-SSS algorithm requires a larger amount of computation time, i.e. about a factor of (codebook size $\times$ number of states) greater than ML-SSS during training. There is practically no difference in recognition time.

During the split, certain phoneme label contexts might be lost for allophones that are not observed in the training data. This is called the *unseen triphone problem.* For recognition, we have to extrapolate models

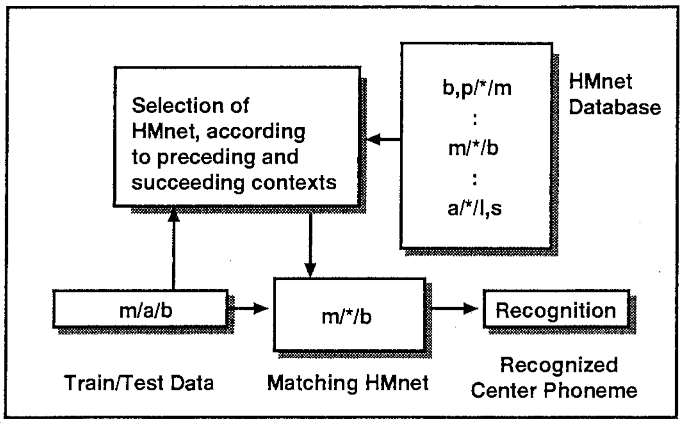
Selection of HMnet, according to preceding and succeeding contexts
b,p/*/m
:
m/*/b
:
a/*/l,s
HMnet Database
m/a/b
m/*/b
Recognition
Train/Test Data
Matching HMnet
Recognized Center Phoneme

図 5: Phoneme classification scheme

表 2: Phoneme misclassification rate (%) for speaker MHT

Number of	ML-SSS		TM-SSS	
states	closed	open	closed	open
100	12.4 %	12.7 %	5.2 %	5.4 %
200	4.2 %	4.6 %	3.3 %	3.7 %
256	3.4 %	**3.9%**	3.1 %	**3.6%**
300	2.7 %	3.1 %	2.7 %	3.3 %
400	2.2 %	2.7 %	2.2 %	3.0 %

for these unseen triphones. Currently, this is implemented by mapping states from the *raw* HMnet to a *filled* HMnet with a program called `Exe.fill_HMnet`.

In the trained HMnet there exist many state paths, each of them representing a separate allophone. Each allophone is represented by a set of preceding, center and succeeding phonemes (subsets of all the possible phonemes). To compare the quality of the HMnet's, we calculated the phoneme occurrence as an average, i.e. the total number of phonemes in a certain context summed up over all the HMnet allophones, normalized by the total number of allophones in the HMnet. Results are shown in Table 3

表 3: Phoneme/allophone occurrence comparison for 256 state TM-SSS and ML-SSS HMnet's

Phone	ML-SSS		TM-SSS	
Context	Raw	Filled	Raw	Filled
Preceding	1.61	2.73	1.92	2.95
Center	1.09	1.14	1.00	1.00
Succeeding	2.45	4.04	2.33	3.91
#Allophones	1020	1466	860	1343

As an example, suppose we have only the allophones m,n/a/m,n and b,d/e,a/s in an HMnet. In this case there are 2 allophones in the HMnet and there are 4 preceding phonemes m,n,b,d, so the preceding phoneme average is $4/2 = 2$. Similarly there are 3 center phonemes a,e,a, so the center phoneme average is $3/2 = 1.5$, and the succeeding phoneme average is $3/2 = 1.5$.

Despite the fact that all the center phonemes have been split in some context, there are still some unsplit center phonemes even for a 256 states ML-SSS. Table 4 summarizes them. For example, in the raw ML-SSS HMnet, there are 92 out of 1020 allophones, where the center contexts have not yet been split.

表 4: Non-split center contexts and corresponding allophone counts

ML-SSS		TM-SSS	
Raw	Filled	Raw	Filled
o, w q, t	o, w q, t k, q k, t		h, p
92/1020	205/1466	0/860	1/1343

These differences may be due to the ML-SSS deficiency in gain evaluation discussed in Section 1 and Figure 2.

6 Conclusions

TM-SSS is an alternative for automatic HMM state topology generation of acoustic models. It compares favorably with the state-of-the-art ML-SSS. More precisely, it is now possible to better evaluate the split gain and hence to get a more representative and robust HMnet from the training data. For a similar number of Gaussians, we observed a relative decrease in error rate of about 10% compared to ML-SSS.

The major drawback of TM-SSS is it computational cost that is theoretically (codebook size $\times$ number of states) greater than that of ML-SSS during training. With some tricks, we can reduce this training penalty to about a factor of 100, i.e. TM-SSS takes about 2 weeks vs 3 hours for ML-SSS. Nevertheless, there are no penalties during phoneme recognition and we can expect considerable time savings for word recognition using statistic language models.

We will continue with the development of an optimal HMnet generation algorithm. One possible approach is to split a state as a whole, i.e. not split a context at a time, but all the contexts at the same time. This seems possible for TM-SSS even without increasing substantially the computational requirements.

参考文献

[1] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis,* J. Wiley & Sons, New York, 1984.

[2] L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer, "A new algorithm for the estimation of hidden Markov Model parameters," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.,* New York, USA, 1988.

[3] L. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer, "A tree-based statistical language model for natural language speech recognition," *IEEE Trans. on Acoust., Speech, and Signal Proc.,* Vol. 37, No. 7, pp. 1001-1008, 1989.

[4] P.F. Brown, "Acoustic-phonetic modeling problem in automatic speech recognition," Ph.D. thesis, Department of Computer Science, Carnigie-Mellon University, 1987.

[5] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees,* Wadsworth International Group, 1984.

[6] P. A. Chou, "Optimal partitioning for classification and regression trees," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 13, No. 4, pp. 340-354, April 1991.

[7] A.P. Dempster, N. M. Laird and D. B.Rubin. "Maximum Likelihood from Incomplete Data via the EM algorithm," *Journal of the Royal Statistical Society,* vol.37, No. 1, 1977, pp. 1-38.

[8] G.R. Doddington, "Phonetically sensitive discriminants for improved speech recognition," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.,* pp. 556-559, Glasgow, Scotland, 1989.

[9] Y. Ephraim, A. Dembo, and L.R. Rabiner, "A minimum discrimination information approach for Hidden Markov Modeling," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.,* pp. 25-28, Dallas, USA, 1987.

[10] P. Gopalakrishnan, D. Kanevsky, A. Nadas, and D. Nahamoo, and M. Picheny, "Decoder selection based on cross-entropies," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.,* pp. 20-23, New York, USA, 1988.

[11] X.D. Huang, M.A. Jack, "Hidden Markov modelling of speech based on a semi-continuous model," *IEE Electronics Letters,* vol.24, no. 1, pp.6-7, 1988.

[12] X.D. Huang, "Semi-continuous Hidden Markov Models for speech recognition," Ph.D. thesis, Department of Electrical Engineering, University of Edinburg, 1989.

[13] X.D. Huang, Y. Ariki and M.A. Jack, "Hidden Markov Models for speech recognition" - Edinburg University Press, 1990.

[14] F. Jelinek, "Continuous speech recognition by statistical methods," *Proc. IEEE,* vol.64, pp. 532-556, 1976.

[15] A. Kannan, M. Ostendorf and J. R. Rohlicek, "Maximum Likelihood Clustering of Gaussians for Speech Recognition," *IEEE Trans. on Speech and Audio Proc.,* vol. 2, no. 3, 1994, pp. 453-455.

[16] K.F. Lee, "Hidden Markov models: past, present, and future," *Proc. European Conf. on Speech Commun. and Technology,* Paris, France, 1989.

[17] K.F. Lee, "Large-vocabulary speaker-independent continuous speech recognition: The SPHINIX system," Ph.D. thesis, Department of Computer Science, Carnigie-Mellon University, 1988; also Automatic Speech Recognition: The Development of the SPHINIX System, Kluwer Academic Publishers, 1989.

[18] Y. Linde, A. Buzo and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.,* vol. COM-28, pp. 84-95, Jan. 1980.

[19] A. Nádas, "A decision theoretic formulation of a training problem in speech recognition and a comparisón of training by unconditional versus conditional maximum likelihood," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. ASSP-31, pp. 814-817, 1983.

[20] A. Nádas, D. Nahamoo, M. Picheny and J. Powell, "An iterative "flip-flop" approximation of the most informative split in the construction of decision trees," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, pp. 565-568, 1991.

[21] M. Nishimura and K. Toshioka, "HMM-based speech recognition using multi-dimensional multi-labeling," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, pp. 1163-1166, Dallas, USA, 1987.

[22] M. Ostendorf, H. Singer, "Modification of SSS for Speaker-Independent HM-Net Design", ATR Technical Report TR-IT-0117, 1995.

[23] H. Singer and M. Ostendorf. Maximum likelihood successive state splitting. In *Proc. ICASSP*, pages 601–604, Atlanta, 1996.

[24] R. Schwartz, O. Kimball, F. Kubala, M. Feng, Y. Chow, C. Barry, and J. Makhoul, "Robust smoothing methods for discrete hidden Markov models," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, pp. 548-551, Glasgow, Scotland, 1989.

[25] J. Takami and S. Sagayama, "A sucessive state splitting algorithm for efficient allophone modeling," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, vol. I, 1992, pp. 573-576.

[26] H.P. Tseng, M. Sabin, and E. Lee, "Fuzzy vector quantization applied to hidden Markov modeling," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, Dallas, USA, pp. 641-644, 1987.

[27] S. J. Young, J. J. Odell and P. C. Woodland,"Tree-based state tying for high accuracy acoustic modeling," in *Proc. ARPA Workshop on Human Language Technology*, 1994, pp. 307-312.

[28] H. Kuwabara, Y. Sagisaka, K. Takeda, and M. Abe. Construction of ATR Japanese speech database as a research tool. Technical Report TR-I-0086, ATR, 1989. (in Japanese).