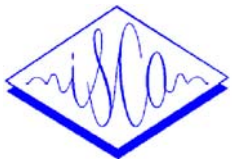

EUROSPEECH '97
5th European Conference on Speech Communication and Technology
Rhodes, Greece, September 22-25, 1997

MAXIMUM LIKELIHOOD SUCCESSIVE STATE SPLITTING ALGORITHM FOR TIED-MIXTURE HMNET

Alexandre Girardi, Harald Singer, Kiyohiro Shikano, Satoshi Nakamura
Nara Institute of Science and Technology
Takayama-cho 8916-5, Ikoma-shi, Nara-ken 630-01 Japan
E-mail: alex-g@is.aist-nara.ac.jp

ABSTRACT

This paper describes a new approach to ML-SSS (Maximum Likelihood Successive State Splitting) algorithm that uses tied-mixture representation of the output probability density function instead of a single Gaussian during the splitting phase of the ML-SSS algorithm. The tied-mixture representation results in a better state split gain, because it is able to measure differences in the phoneme environment space that ML-SSS can not. With this more informative gain the new algorithm can choose a better split state and corresponding data. Phoneme clustering experiments were conducted which lead up to 38% of error reduction if compared to the ML-SSS algorithm.

1. INTRODUCTION

ML-SSS algorithm has been proven to outperform other HMM design algorithms[1]. 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 state pdf's may be efficiently represented by 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.

Afterwards the HMnet is filled in appropriately with the unseen triphones and the single Gaussian representation of a state is replaced by a n-Gaussian representation.

However due to the single Gaussian representation of the pdf of a state during the state split the algorithm does not consider differences in state split gain that may arise for certain pdf's. To see this difference, consider a state s which is considered to be split in states s_1 and s_2 and whose pdf's we can see in Figure 1.

The split gain in both algorithms is a function of the pdf representation of the candidate split states. In this extreme case the resulting pdf's are identical for ML-SSS resulting in a split gain of zero. In the case of a tied-mixture representation we may obtain different pdf's for s_1 and s_2. Consequently resulting in a non zero split gain. Note that a non zero gain is desired once we have in this case information (phoneme context) that have some meaning when split.

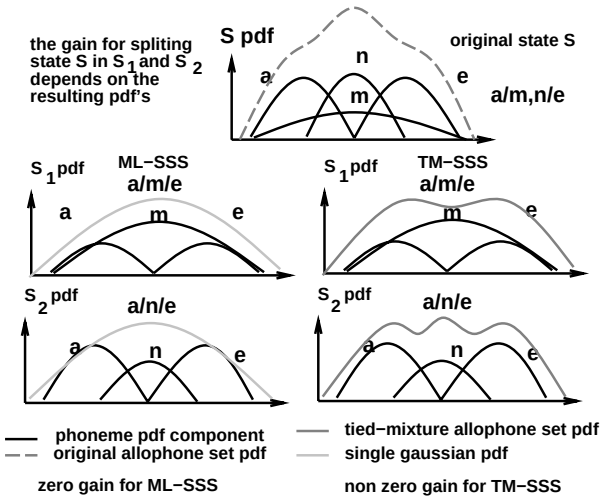

Figure 1. TM-SSS vs ML-SSS split gain

A continuous density representation of the pdf of a state would be probably even better, but its computational cost is prohibitive for this kind of algorithm. In the tied-mixture approach the computational cost is reasonable, provided that we accept the constraint that the codebook does not change during a split.

The usefulness of the tied-mixture pdf representation during the state split is the subject of this paper.

2. TM-SSS ALGORITHM

In order to represent the pdf of a state during the split phase by a tied-mixture representation, the proposed algorithm, from now on called Tied-Mixture Successive State Splitting algorithm (TM-SSS), introduces several changes to the basic algorithm (ML-SSS).

The new algorithm is as follows:

1. create a codebook
   (a) flat start(estimate means and variances)
   (b) vector quantization to get the codebook
   (c) Baum-Welch (BW) of initial HMM
2. iterate over the candidate states:
   (a) find best split for each domain and factor
   (b) split the state with the highest expected likelihood gain
   (c) **if** update codebook
      i. run BW, train means, variances, weights and transitions over all states
      ii. use only the most significant mixtures
      else
      i. run BW over affected states, only train weights and transitions
3. fill the HMnet with the unseen triphones

The BW algorithm follows the classical tied-mixture approach [2].

In order to derive the gain function we define the observed data y_1^T as $y_1^T = \{y_1, y_2, \ldots, y_f, \ldots, y_T\}$ and the related hidden components s_1^T as $s_1^T = \{s_1, s_2, \ldots, s_f, \ldots, s_T\}$.

Requiring that means and variances of the codebook are constant during the split, as stated in the previous algorithm, the gain function is expressed as a difference of the expected log likelihood before and after the split. For each iteration k:

$$
\begin{aligned}
&Q(\theta^{(k+1)}|\theta^{(k)}) = E[\log \mathrm{p}(y_1^T, s_1^T|y_1^T, \theta^{(k+1)})|\theta^{(k)}] \\
&= \sum_{s_1^T} \mathrm{p}(s_1^T|y_1^T, \theta^{(k)}) \log \mathrm{p}(y_1^T, s_1^T|\theta^{(k+1)}) \\
&= \sum_{s:s=s_t, s'=s_{t-1}}^{S} \sum_t \xi_t(s,s') \log a(s,s') \\
&+ \sum_{s:s=s_t}^{S} \sum_t \sum_l \gamma_t(s,l) \log b_l(s) \\
&+ \sum_{s:s=s_t}^{S} \sum_t \sum_l \gamma_t(s,l) \log N(y_t|\mu_l, \Sigma_l) \qquad (1)
\end{aligned}
$$

where

$$
\begin{aligned}
\gamma_t(s,l) &= \mathrm{p}(s_t = s, v_l|y_1^T, \theta^{(k)}) \\
\xi_t(s,s') &= \mathrm{p}(s_t = s, s_{t-1} = s'|y_1^T, \theta^{(k)}) \\
a(s,s') &= \mathrm{p}(s_t|s_{t-1}, \theta^{(k+1)}) \\
b_l(s) &= \mathrm{p}(v_l|s_t, \theta^{(k+1)}) \\
N(y_t|\mu_l, \Sigma_l) &= \mathrm{p}(y_t|v_l, \theta^{(k+1)})
\end{aligned}
$$

and $a(s,s')$ is the transition probability from the state $s = s_{t-1}$ to state $s' = s_t$, $b_l(s)$ is the weight of the codebook vector v_l and $N(y_t|\mu_l, \Sigma_l)$ is the Gaussian mixture of the observation y_t with respect to v_l. θ are the HMnet parameters.

Note that this equation allows us, assuming the same constraint over the counts as in ML-SSS[1], to consider only the contribution to $Q(\theta^{(k+1)}|\theta^{(k)})$ that comes from the split state s^*. Hence, our gain $G(s^*)$ resumes to:

$$
\begin{aligned}
G(s^*) = \;& Q(\theta^{(k+1)}|\theta^{(k)})(s_1, s_2) - Q(\theta^{(k+1)}|\theta^{(k)})(s^*) \\
=& \textstyle\sum_{s,s'} M_1(s,s') \log a(s,s') - M_1(s^*,s^*) \log a(s^*,s^*) \\
&+ \textstyle\sum_l [\sum_s M_2(s,c,l) \log b_l(s) - M_2(s^*,c_d,l) \log b_l(s^*)] \\
&+ \textstyle\sum_{j(c)} [M_3(s_0, j(c)) + M_3(s_1, j(c)) - M_3(s^*, j(c))]
\end{aligned} \qquad (2)
$$

where

$$
\begin{aligned}
M_1(s,s') &= \sum_{j(c)} \xi_{j(c)}(s,s') \\
M_2(s,c,l) &= \sum_{j(c)} \sum_{t:x_t = x_{j(c)}} \gamma_t(s,l) \\
M_3(s,j(c)) &= \sum_{t:x_t = x_{j(c)}} \sum_l \gamma_t(s,l) \log N(y_t, \mu_l, \Sigma_l)
\end{aligned}
$$

and $j(c)$ represents all the generalized allophones [3] in a certain context $c \in \{preceding, center, succeeding\}$ phonemes. If the split is in the temporal domain $(s,s') \in \{(s_1,s_1), (s_1,s_2), (s_2,s_2)\}$ (see Figure 2).

The first term $M_1()$ in equation 2 is independent of the c chosen and constant due to the constraint of fixed transition probabilities during the split. Additionally due to the constraint of constant codebook during split the last term $M_3()$ is constant, too. These constraints reduce our gain to the evaluation of the middle term $M_2()$.

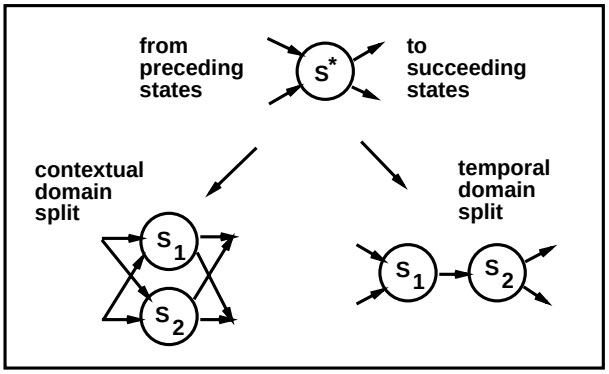

Figure 2. State split

The state split is performed, from the split state s^*, by creating two states s_1 and s_2, i.e. two new set of mixture weights. The split state is the one that gives the highest gain $G(s^*) = \max_s G(s)$. The split is accomplished by applying Chou's theorem[4] to the weights of the tied-mixtures, reestimating new centroids by iterating until the change in the gain becomes smaller than an arbitrarily chosen ϵ, typically $1.0e^{-5}$.

In the contextual domain the initial weight vector for state s_1 is obtained by copy (to assure that the likelihood will not decrease) and for state s_2 by disturbing the original weights $b_l(s^*)$ by $(1 + (-1)^l \epsilon')$ and then renormalizing. ϵ' is arbitrarily chosen, typically $1.0e^{-3}$.

The output of the TM-SSS training algorithm is a tree of non overlaping phoneme space environments [3]). The unseen triphones (gaps and holes of the phoneme space environment [3]) are placed in the leaves of this tree, based on the co-occurrence counts of phoneme labels collected from the split history.

3. EXPERIMENTS

Phoneme classification experiments using 25 phonemes for one male speaker (speaker MHT of ATR database [5]) were performed with the aim of comparing the performance of TM-SSS and ML-SSS.

The phoneme classification was performed using the preceding and succeeding contexts (see Figure 3), i.e., we assume that the correct left and right phoneme contexts are known.

Two codebook sizes for TM-SSS were considered and fixed at 256 and 512 Gaussians. The final 256 states ML-SSS HMnet was trained with 1 and 2 mixtures per state which give the same total number of 256 and 512 Gaussians.

Experiment description:

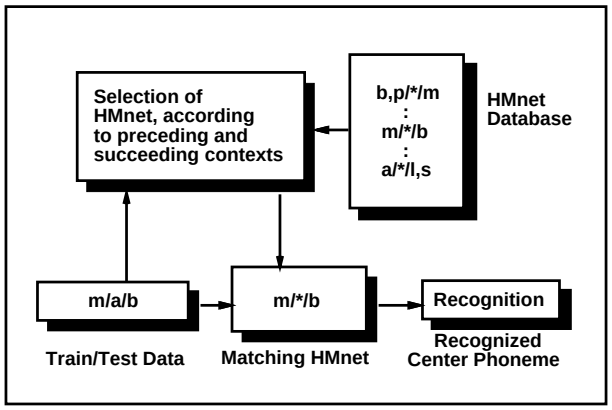

Figure 3. Phoneme classification

Table 1. Phoneme classification (%) for ML-SSS and TM-SSS with 256 states each

Gaussian	Algorithm	
Number	TM-SSS	ML-SSS
256	3.1	4.4
512	2.5	4.0

- The initial HMnet consists of 3 states aligned left to right, representing all the phonemes in all the contexts.
- Only contextual split was performed.
- The TM-SSS is performed with codebooks of 256 and 512 Gaussians, respectively
- The HMnets are split up to 256 states.
- The ML-SSS HMnet is retrained with 2 Gaussians per state.

Recognition results for both, TM-SSS and ML-SSS algorithms, are summarized in Table 1 for the speaker MHT. 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 5240 words. Only test data results are presented.

From Table 1 we see that even when ML-SSS has twice the number of Gaussians the recognition rate is smaller than TM-SSS.

Table 2 also shows how the HMnet generated with this new algorithm behaves when used with other speakers. In this experiment the HMnet obtained from the MHT speaker was retrained with the phonemes of the even-numbered words for the target speaker and tested against the phonemes of the odd-numbered words of the same target speaker. All the 5240 words uttered for each speaker had been used.

Another way to measure the goodness of the new approach is analyzing how the phonemes had been split.

Table 2. Cross recognition (%) for ML-SSS and TM-SSS with 256 states each

Spk.	256 Gaussians		512 Gaussians	
	TM-SSS	ML-SSS	TM-SSS	ML-SSS
MAU	7.4	7.8	4.9	5.5
MXM	8.3	9.7	6.3	6.9
FYM	10.9	12.5	8.9	7.2
FTK	10.6	12.1	8.6	7.4
FMS	10.4	11.3	8.2	7.9

In an HMnet every state path defines a model representing a set of triphones (or generalized allophones). In SSS this set of triphones are convex spaces which may be represented by sets of phonemes, each one representing one context (preceding, center and succeeding phoneme set).

For an HMnet with 256 states, almost all the center phonemes should have already been split. Table 3 shows the triphones whose center phonemes remain unsplit in the final HMnet, as well as their occurrence. *Orig* indicates HMnet before the estimation of the unseen triphones in the training data and *Fill* after filling the HMnet with the unseen triphones.

Table 3. Unsplit center phonemes for HMnet with 256 states (speaker MHT)

ML-SSS %		TM-SSS %	
Orig	Fill	Orig	Fill
(o, w) (q, t)	(o, w) (q, t) (k, q) (k, t)		(h, p)
3.9%	17.3%	0.0%	0.4%

We can see from Table 3 that using the TM-SSS algorithm almost all the center phonemes have been split. With ML-SSS an important amount of unsplit center phonemes, equivalent to 17.3% of all triphones contexts, remains. We think that the improvement with TM-SSS is due to the more accurate split gain calculation.

4. CONCLUSIONS

For the same number of Gaussians the results using TM-SSS are about 38% (relative decrease in error rate) better than using the ML-SSS algorithm for the speaker dependent case. For the cross recognition against 5 speakers the results are about 10% better for one Gaussian per state.

The superiority of the TM-SSS algorithm is shown not only by its recognition rate but also by the visible quality of the final phoneme split, as attested by its performance on splitting the center phonemes.

However, the present implementation of TM-SSS algorithm requires a larger amount of computation time during the training phase compared to ML-SSS.

REFERENCES

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

[2] J. R. Bellegarda and D. Nahamoo. Tied mixture continuous parameter modeling for speech recognition. *IEEE Trans. Signal Process.*, 38(12):2033–2045, 1990.

[3] S. Sagayama. Phoneme environment clustering for speech recognition. In *Proc. ICASSP*, volume 1, pages 397–400, Glasgow, May 1989.

[4] P. Chou. Optimal partitioning for classification and regression trees. *IEEE Trans. PAMI*, 13(4):340–354, April 1991.

[5] 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).