MAXIMUM LIKELIHOOD SUCCESSIVE STATE SPLITTING

Harald Singer[1], *Mari Ostendorf*[2]

[1]ATR Interpreting Telecommunications Res. Labs.
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan
E-mail: singer@itl.atr.co.jp
[2]ECS Department, Boston University

ABSTRACT

Modeling contextual variations of phones is widely accepted as an important aspect of a continuous speech recognition system, and much research has been devoted to finding robust models of context for HMM systems. In particular, decision tree clustering has been used to tie output distributions across pre-defined states, and successive state splitting (SSS) has been used to define parsimonious HMM topologies. In this paper, we describe a new HMM design algorithm, called maximum likelihood successive state splitting (ML-SSS), that combines advantages of both these approaches. Specifically, an HMM topology is designed using a greedy search for the best temporal and contextual splits using a constrained EM algorithm. In Japanese phone recognition experiments, ML-SSS shows recognition performance gains and training cost reduction over SSS under several training conditions.

1. INTRODUCTION

Successive state splitting (SSS) is a powerful technique for HMM design that provides a mechanism for automatically learning the most appropriate HMM topology [1]. The basic idea behind SSS is that a network of HMM states (referred to as an HM-Net) can be increased in size by choosing to split the state with the most variability, and then picking the best splitting domain (either temporal or contextual) for that state. The iterative application of this splitting results in an HM-Net that efficiently represents contextual and temporal variability of specified subword units (e.g. phones or morae). SSS has been used successfully by ATR and shown to outperform other HMM design techniques in several studies [2].

A disadvantage of SSS, as it is currently implemented, is that it only works well for training topologies on speaker-dependent data. In speaker-independent training, the state with the most variability, which would be chosen by SSS, is likely to reflect speaker variability rather than coarticulation or temporal effects.

A technique similar to successive state splitting used in many HMM systems is divisive distribution clustering, sometimes referred to as decision tree context modeling, e.g. [3, 4] for Gaussian distributions. In distribution clustering, training observations are associated with states in some pre-specified HMM topology, sufficient statistics are collected for each state, and the distributions are clustered using decision tree growing of contextual splits to maximize the likelihood of the training data. Unlike SSS, decision tree context modeling has been successfully used in speaker-independent HMM training. An important difference between the decision tree and SSS approaches is that the choice of which distribution to split in decision tree modeling is based on a specific contextual split rather than on a generic measure of state distribution variance as used in SSS. This difference suggests a distribution variance as used in SSS. This difference suggests a solution to the problem of speaker-independent training in SSS: simply choose the state to split based on the most likely splitting domain for that state, rather than choosing the splitting domain after choosing the state.

2. ALGORITHM

The original version of SSS [1] is an iterative algorithm that progressively grows an HM-Net, where each state in the network is associated with a 2-component Gaussian mixture. First the state is selected to be split according to which has the largest divergence between its two mixtures, then splits in the contextual and temporal domains are tested and the best one is chosen, and the HM-Net is retrained using the Baum-Welch algorithm. In the new version of the algorithm, we jointly choose the best state to split and the best split for that state, which primarily involves a reordering of the main steps of the original SSS algorithm. The new algorithm, which we will refer to here as ML-SSS to distinguish it from SSS, proceeds as follows (see also Figure 1)

- **Initialization:** Run Baum-Welch
- **Iterate:**
  1. For each candidate state, find the best split domain and factor, and calculate the expected likelihood gain.
  2. Split the state with highest expected likelihood gain.
  3. Determine affected states and run Baum-Welch for all data that passes through affected states.

Similar to the approach in [5], we build the topology with single Gaussians and then retrain with mixtures. In order to design the HM-Net using a maximum likelihood objective in all steps of the algorithm, the tests for a split change form. In addition, since the tests are called frequently, the search for the best contextual split must be implemented efficiently, which is accomplished here by an extension of the Chou optimal partitioning algorithm [6]. In our reformulated version of SSS, HM-net design is similar to divisive distribution clustering, except that distribution sharing is allowed in both the contextual and temporal domains and that arbitrary topologies can be learned through combinations of temporal and contextual splits.

0-7803-3192-3/96 $5.00©1996 IEEE

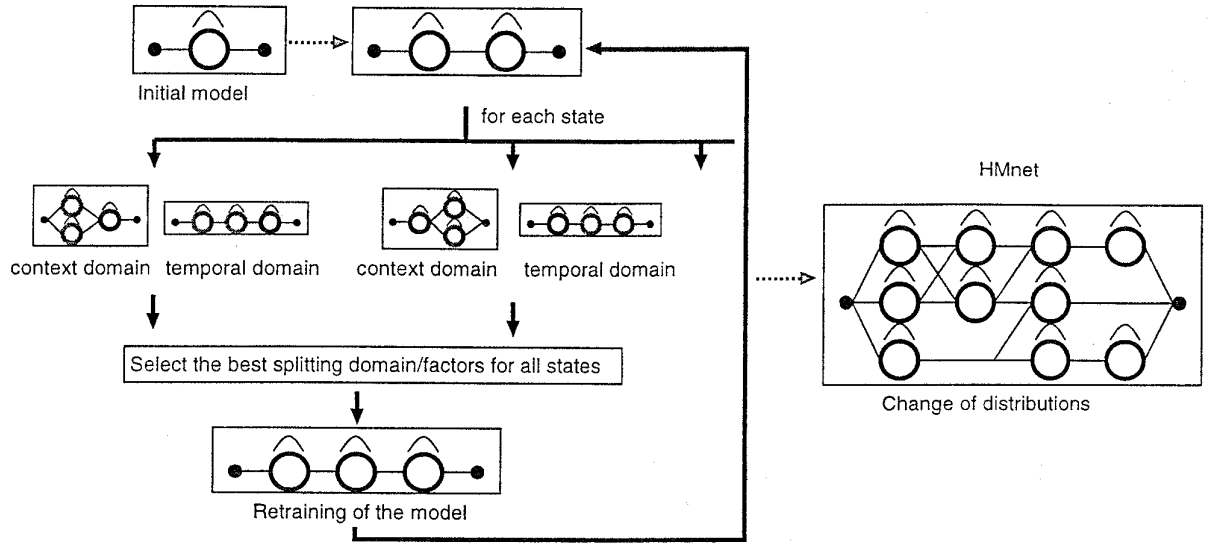

Figure 1. Restructured ML-SSS algorithm

The objective at each iteration of the algorithm is to choose and split states to maximize the likelihood of the training data. As in HMM parameter estimation for a fixed topology, it is not feasible to maximize the likelihood of the observations directly, so instead we use the expectation-maximization (EM) algorithm and maximize the expected log likelihood $Q(\theta|\theta^{(p)}) = E_{\theta^{(p)}}[\log \mathrm{P}(y_1^T, s_1^T|\theta)]$, where y_1^T is the observed data and s_1^T is the hidden HMM state sequence. Further, in estimating the possible gain in expected likelihood by splitting a particular state s^*, the contribution to $Q(\theta|\theta^{(p)})$ from all other states is held fixed by constraining the counts

$$\gamma_t(s) = \mathrm{P}(s_t = s|y_1^T, \theta^{(p)}) \quad (1)$$

$$\xi_t(s,s') = \mathrm{P}(s_t = s, s_{t-1} = s'|y_1^T, \theta^{(p)}) \quad (2)$$

for $s, s' \neq s^*$ in a "constrained" EM algorithm. Thus, for the case where s^* is split into s_0 and s_1, the associated gain in likelihood is

$$G(s^*) = G_A(s^*) + G_B(s^*) \quad (3)$$

$$G_A(s^*) = \sum_{s=s_0,s_1} \left[\sum_{s'=s_0,s_1} N(s,s') \log a_{ss'} \right] - N(s^*,s^*) \log a_{s^*s^*} \quad (4)$$

$$G_B(s^*) = \sum_{s_0,s_1} \sum_t \gamma_t(s) \log \mathrm{P}(y_t|s, \theta_{B(s)}) - \sum_t \gamma_t(s^*) \log \mathrm{P}(y_t|s^*, \theta_{B(s^*)}) \quad (5)$$

where $a_{ss'} = \mathrm{P}(s_t = s|s_{t-1} = s', \theta_{A(s)})$ and

$$N(s,s') = \sum_t \xi_t(s,s').$$

Equation 3 gives a criterion by which we can compare different candidate splits within and across domains and across states, and choose the split which most increases the expected likelihood of the entire training set. The parameters $\{\theta_{A(s)}, \theta_{B(s)}\}$ are chosen using the constrained EM algorithm testing different possible splits, and then the state with the largest gain is split accordingly. If the initial split is chosen appropriately, the constrained function $Q(\theta|\theta^{(p)})$ is guaranteed to be non-decreasing since the terms depending on $s \neq s^*$ do not change and the likelihood due to other terms cannot decrease. Note that Equation 3 does not give the increase in likelihood *per se*, but rather the increase in expected likelihood, and so maximizing $G(S)$ over S only guarantees that likelihood is non-decreasing, not necessarily that we have chosen the split that maximally increases likelihood.

Since $G(s^*)$ measures an increase in the expected joint likelihood of the observations and states, it takes a different form than the test used in SSS for choosing a splitting domain, which is based directly on observation likelihood. In addition, Equation 3 takes a different form from the criterion used in SSS for determining the best node to split (Equation 1 in [1]), but in this case the ML-SSS criterion is preferable. The SSS criterion is a measure of the distance between the two generic mixture components and not the gain in likelihood relative to having a single state, and it cannot be related to an increase in the likelihood of the training data in any way.

For the specific case of evaluating $G(s^*)$ for **contextual splits**, the data in s^* can be unambiguously assigned to either s_0 or s_1, so only the maximization step of the EM algorithm is required. In addition, the constraint effectively fixes the state transition probabilities out of the new state and so $G(s^*) = G_B(s^*)$. The problem of finding a binary split based on phonetic context to maximize $G_B(s^*)$ is the same as the problem of finding the best question in decision tree clustering. In ML-SSS, because splits are evaluated frequently, it is critical that split design be efficient. We use the Chou partitioning algorithm [6] with a maximum Gaussian log likelihood objection function and with expected state likelihoods in place of state occupancy counts.

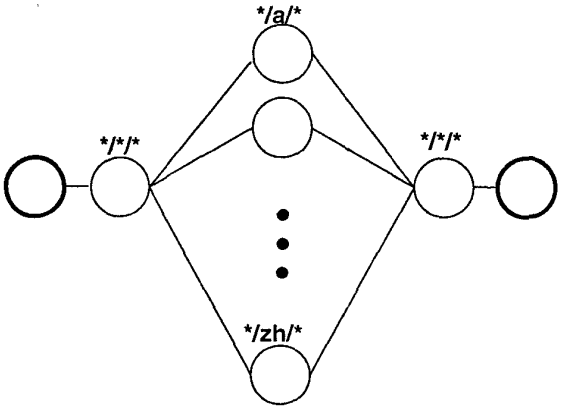

Figure 2. Initial HM-net topology for all experiments.

For the specific case of evaluating $G(s^*)$ for temporal splits, the association of data to states is hidden, so a local EM algorithm is needed to estimate the best temporal split parameters. In addition, both $G_A(s^*)$ and $G_B(s^*)$ contribute to the expected gain $G(s^*)$. In the local EM algorithm, a forward-backward algorithm is run over the observations associated with state s^* (i.e. $\{y_t : \gamma_t(s^*) > 0\}$) for the two new states s_0 and s_1 only, providing normalized counts $\tilde{\gamma}_t(s)_B$ an $\tilde{\xi}_t(s, s')$. The normalized counts are then scaled by $\gamma_t(s^*)$ an $\xi_t(s^*, s')$ for parameter re-estimation. Fixed points of known boundaries are used in order to reduce computational requirements of the temporal split, taking advantage of a feature of SSS design. In the experiments reported here, the fixed points are phone boundaries.

Compared to SSS, the contextual split design procedure is much faster in ML-SSS, but the temporal split design procedure is somewhat slower. On balance, ML-SSS is faster than SSS design, especially since temporal splits are disallowed after a certain maximum length is reached. However, ML-SSS requires storage of the forward-backward counts for split design, which increases the memory requirements of HMM topology design.

3. EXPERIMENTS AND RESULTS

We conducted a series of phoneme recognition experiments on subsets of several Japanese corpora, comparing SSS and ML-SSS in speaker-dependent, multi-speaker and speaker-independent tests. Subset 1, for training and testing, consisted of 3 male and 3 female professional speakers each uttering 5240 isolated words. Subset 2, for testing only, consisted of 10 male and 10 female professional speakers each uttering 279 phrases. Subset 3, for training only, consisted of 15 female speakers, each uttering 50 phoneme balanced sentences in continuous phrases. The overall duration of the subset 3 training data was 64 minutes (not counting silence intervals), and the number of phonemes was 41607.

The initial HM-net topology in Fig. 2 allows sharing even between different phones. In all experiments, the HM-net topology is designed assuming single-mode Gaussian distribution starting from this initial topology. The models are then retrained to have a specific mixture output distribution. Models were trained with 200 and/or 400 states, and 1 and/or 3 mixtures.

The most difficult test of ML-SSS is in speaker-dependent experiments, since the major source of variation is phonetic

Table 1. *Speaker-dependent phoneme recognition accuracy for SSS vs. ML-SSS topology design. N_s = number of states, N_m = number of mixtures.*

	% Accuracy					
	N_s=200, N_m=1		N_s=400, N_m=1		N_s=400, N_m=3	
spkr	SSS	ML-SSS	SSS	ML-SSS	SSS	ML-SSS
MHT	93.9	92.8	95.4	94.3	96.1	96.0
MAU	93.6	93.2	95.2	95.2	96.4	96.7
MXM	91.7	91.9	93.8	93.9	95.3	95.1
FTK	91.5	91.1	92.9	94.0	94.7	95.0
FMS	89.7	91.3	91.9	93.2	94.2	94.6
FYM	90.7	92.4	92.9	93.6	95.1	95.5
avg	91.9	92.1	93.7	94.1	95.3	95.5

context and SSS works well on this task. Training was performed on the even numbered words in subset 1, testing was performed on the odd numbered words in subset 1, separately for each speaker. As shown in Table 1, ML-SSS gave slightly better performance than SSS for almost all conditions and speakers, except for the one speaker that SSS had been tuned on (2-4% error reduction on average). In addition, ML-SSS had lower computational costs, though higher memory requirements.

In multi-speaker experiments, we pooled the even-numbered words in subset 1 for training and tested on the odd-numbered words in subset 1. Results are shown in Fig. 3. ML-SSS consistently performs better than SSS, the difference being greatest for the higher context resolution model (400 states) using only 1 mixture. Using 3 mixtures the difference is smaller, which is not surprising since the lack of allophones can be compensated for by the introduction of mixtures.

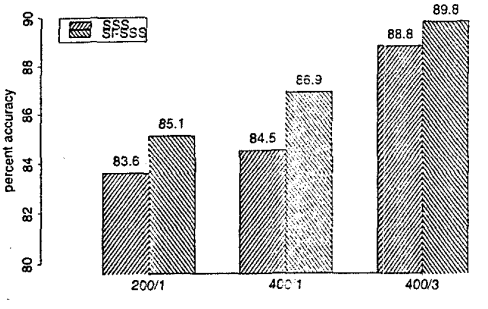

Figure 3. Phoneme recognition accuracy for SSS vs. ML-SSS for a multi-speaker recognition task.

In the first speaker independent test, we used the previously trained models for the multi-speaker experiments, but performed recognition tests on subset 2, i.e. 279 phrase utterances of 20 speakers. Results for 400 states and 3 mixtures are displayed in Fig. 4, where the 6 training speakers have been marked with an asterisk. We can see that ML-SSS is performing up to 4% better than SSS and only doing slightly worse, i.e. less than 1%, for some of the training speakers which had previously been used for the development of the original SSS algorithm.

In a second experiment, we trained on continuous speech material, under the hypothesis that the resulting topology would be more suitable for recognizing continuous speech. However, both SSS and ML-SSS require phoneme bound-

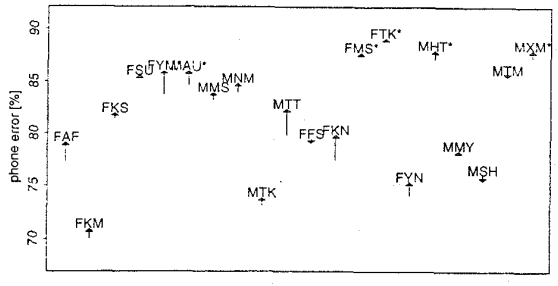

Figure 4. Phoneme recognition accuracy for SSS vs. ML-SSS topology training for a speaker-independent recognition task. Arrows show change from SSS to ML-SSS performance. Training speakers are marked with an asterisk.

ary information for practical implementation, which was only given for the isolated words. To get this phoneme alignment, we first performed Viterbi alignment using VFS speaker adaptation [7] of previously trained models.

The automatically phoneme-aligned data from subset 3 (15 female speakers selected by a speaker tree clustering of 286 speakers [8]) were then used for ML-SSS topology training. The resulting HM-net was then retrained with 3 mixture Gaussians per state using standard Baum-Welch retraining. For comparison, we also used the subset 3 data to retrain an HM-net that had a topology designed using the standard SSS paradigm, which is to used the hand-labeled speaker-dependent data from speaker MHT. Thus the difference in the two models is only in the topology design: SSS based speaker-dependent hand-labeled data vs. ML-SSS based on speaker-independent Viterbi-aligned data. As shown in Figure 5, there is a consistent gain in performance using a speaker-independent ML-SSS topology rather than the speaker-dependent SSS topology, despite the presumed small loss in accuracy due to using Viterbi rather than hand alignments.

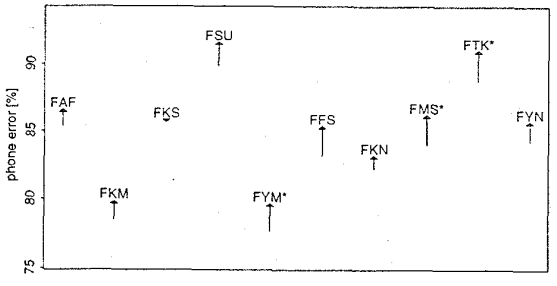

Figure 5. Phoneme recognition accuracy for single speaker SSS vs. multi-speaker ML-SSS topology training, when both HM-nets are retrained on speaker-independent continuous speech. Arrows show change from SSS to ML-SSS performance. Training speakers are marked with an asterisk.

Finally, to assess the benefits of speaker-independent topology training on continuous data vs. isolated words, we also retrained an HM-net, that had been built from the isolated word utterances in subset 1, with the continuous speech data in subset 3. Results in Table 2 show that any training on continuous speech improves performance if testing is on continuous speech, but further gains are observed when the topology is initially trained on continuous speech.

Table 2. Comparison between different topology and retraining strategies for recognition on continuous speech.

topology/retraining	average accuracy
words/words	79.7
words/phrases	82.5
phrases/phrases	85.5

4. SUMMARY

In this paper we have proposed a new algorithm for HMM topology design, that is both an extension of successive state clustering and a generalization of decision tree distribution clustering. SSS is limited in that it cannot handle speaker-independent training data, primarily because states are chosen to be split based on variance and not based on the gain that would be achieved by a specific split. By choosing the node and the candidate split at the same time, we can avoid splitting states that simply reflect speaker variability and therefore achieve better performance in a speaker-independent task. By allowing both contextual and temporal splits, ML-SSS also represents a generalization of decision tree distribution clustering techniques.

Experimental results demonstrate that ML-SSS outperforms SSS, with the added benefit of lower computational costs, and that better results can be obtained by matching the style of the data used for topology training to the test style. We anticipate that in training on spontaneous speech data, the advantages of ML-SSS will play a significant role and enable the design of robust HMM topologies.

REFERENCES

[1] J. Takami and S. Sagayama, "A successive state splitting algorithm for efficient allophone modeling," In *Proc. ICASSP*, Vol. 1, pages 573–576, San Francisco, March 1992.

[2] A. Nagai, K. Yamaguchi, and A. Kurematsu, "ATREUS: A comparative study of continuous speech recognition systems at ATR," In *Proc. ICASSP*, Vol. II, pages 139–142, Minneapolis, 1993.

[3] L. Bahl, P deSouza, P. Gopalakrishnan, D. Nahamoo, and M. Picheny, "Decision trees for phonological rules in continuous speech," In *Proc. ICASSP*, pages 185–188, 1991.

[4] K. Lee, S. Hayamizu, H. Hon, C. Huang, J. Swartz, and R. Weide, "Allophone clustering for continuous speech recognition," In *Proc. ICASSP*, pages 749–752, 1990.

[5] S. Young, J. Odell, and P. Woodland, "Tree-based state tying for high accuracy acoustic modelling," In *Proc. ARPA HLT Workshop*, pages 286–291, 1994.

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

[7] K. Ohkura, M. Sugiyama, and S. Sagayama, "Speaker adaptation based on transfer vector field smoothing with continuous mixture density HMMs," In *Proc. ICSLP*, pages 369–372, Banff, 1992.

[8] T. Kosaka and S. Sagayama, "Tree-structured speaker clustering for fast speaker adaptation," In *Proc. ICASSP*, pages 570–573, Adelaide, 1994.