Start- and End-node Segmental-HMM Pruning

Y. Shiga and P. J. B. Jackson *

Abstract

An efficient decoding algorithm for segmental HMMs (SHMMs) is proposed with multi-stage pruning. The generation by SHMMs of a feature trajectory for each state expands the search space and the computational cost of decoding. We reduce it in three ways: pre-cost partitioning, start-node (SN) beam pruning, and conventional end-node (EN) beam pruning. Experiments show that partitioning cuts computation by 20–25% for supervised training, and 40–50% for phone classification, without degradation in recognition accuracy; SN and EN beam pruning together give 80% reduction for embedded recognition on triphone SHMMs, with less than 0.1% degradation.

*Centre for Vision, Speech and Signal Processing, School of Electronics and Physical Sciences, University of Surrey, Guildford, GU2 7XH, United Kingdom, email: Y.Shiga@surrey.ac.uk

Introduction. In generating a feature-vector trajectory for each state, segmental HMMs (SHMMs) can represent speech dynamics and so outperform conventional HMMs, whose independence assumption limits their treatment of acoustic transitions, both for automatic speech recognition (ASR) and parametric speech synthesis [1, 2]. The HMM's success stems from its efficient recursive decoding algorithms, though embedded training and recognition (without phone boundary constraints) require far more computation than supervised training or phone classification. The search space expands further with combinations of context-sensitive models (e.g., triphones). Typical solutions are depth-first search [3, 4] and beam search for breadth-first decoding [5].

In Viterbi-style SHMM decoding [6], output probability b_i for state i is computed over observation sequences (segments) of various duration. So, the search problem re-emerges for SHMMs because multiple segment hypotheses must be evaluated at each time t; whereas, for HMMs, the output probability is evaluated *only once* per state for each observation y_t. It is essential to develop efficient decoders to save time for experimenting, as well as to develop real-time applications. Russell [6] proposes beam pruning and duration pruning for SHMMs, showing their effectiveness experimentally. Our aim is to study various pruning strategies to improve the computational efficiency of SHMM decoding.

We propose an efficient Viterbi-style SHMM algorithm that decomposes decoding into two operations: find best state transition, and select best segment

duration. Beam pruning is applied separately at each stage to cut the exploding search space, which offers a synergistic approach to computational reduction. We focus on minimizing the number of output probability calculations, which consume over 97% of computation in recognition, even with single multivariate Gaussian probability density functions (pdfs) per state. Other HMM pruning methods, e.g., on partial output probabilities, could be used with SHMMs, yet we concentrate here on more general strategies for segmental models. The next section outlines the decoding algorithm that underlies our pruning strategies. Then we describe the strategies, report their effect in experiments and conclude.

Two-step decoding algorithm. We introduce two probabilities for each node in the trellis, at time t in state i: *start-node probability* (SNP) and *end-node probability* (ENP). Let the ENP, $\delta_t(i)$, denote the probability of the best path to this node, given the observations up to t, $\boldsymbol{y}_1^t = y_1..y_t$, which is expressed recursively for maximum segment duration D [1], then factored:

$$\begin{aligned} \delta_t(i) &= \max_j \max_{d=1..D} \delta_{t-d}(j)\, a_{ji}\, b_i(\boldsymbol{y}_{t-d+1}^t) \\ &= \max_{d=1..D} \nu_{t-d+1}(i)\, b_i(\boldsymbol{y}_{t-d+1}^t), \end{aligned} \tag{1}$$

where a_{ji} denotes the transition probability from states j to i; $b_i(\boldsymbol{y}_{t-d+1}^t)$ is output probability of the segment (of duration d) for state i. The SNP, $\nu_t(i) = \max_j \delta_{t-1}(j)\, a_{ji}$, is the maximum product of the ENP and state-

transition probability from all predecessors to that node, depicted in Fig. 1 (left). The ENP in eq. (1) is the maximum product of SNP and output probability over all segment durations, as in Fig. 1 (right). Our implementation passes tokens, storing SNP for D frames, current ENP, best predecessor and duration for each node. The decomposition in eq. (1) cuts the number of multiplications by $1/D$. However, the algorithm alone cannot reduce the search space when embedded training/recognition is performed, or context-sensitive models decoded; pruning is needed.

Pruning. Our recognizer uses four types of pruning: (1) pre-cost partitioning, (2) SN beam pruning, (3) EN beam pruning, and (4) duration pruning [6]. Here, we examine (1), (2) and (3).

Pre-cost partitioning: First, permitted paths are propagated backward through the trellis, separately activating start and end nodes. Later in the forward decoding pass, segment and transition probabilities are accumulated to compute SNPs and ENPs only at valid SNs and ENs. Thus, likelihoods for nodes that cannot be occupied are not computed, exploiting the topology of permissible state transitions and maximum segment duration D, without any loss of recognition accuracy. It is effective in supervised (e.g., phone-level) training and classification. Fig. 2 illustrates a 3-state model aligned to a 7-frame utterance

starting at t_S and ending at t_E, with $D = 3$: (a) the backward pass starts from the final null node and activates two ENs at t_E, and hence 6 SNs; (b) those SNs activate new ENs; (c) activation propagates back to the initial null node; (d) the decoder's forward pass removes inaccessible SNs. Partitioning eliminated 14 out of 21 nodes, leaving 7 SNs and 7 ENs.

Beam pruning: After partitioning, the decoder independently prunes SNs and ENs. It prunes nodes for paths that could later attain a higher likelihood, but is particularly effective at reducing the computational load for embedded training and recognition. In end-node pruning, ENs are pruned at each time t using the EN beam threshold θ^{E} and the highest ENP for all states i, $\widehat{\delta}_t$:

$$\text{if } \ln \delta_t(i) < (\ln \widehat{\delta}_t - \theta^{\mathrm{E}}), \text{ then kill EN of state } i \text{ at } t.$$

EN pruning of $\delta_t(i)$ corresponds to standard beam pruning [6]. Many candidate segments evaluated in ENP calculation are pruned *after* output-probability computation. In start-node pruning, the SN beam threshold θ^{S} is applied wrt. $\widehat{\nu}_t$, the best SNP at t $\forall i$:

$$\text{if } \ln \nu_t(i) < (\ln \widehat{\nu}_t - \theta^{\mathrm{S}}), \text{ then kill SN of state } i \text{ at } t.$$

SN pruning applies immediately *before* output-probability computation, and directly reduces computation using θ^{S}.

Experiments. To assess the reduction in computational load of our SHMM decoder for each pruning strategy, we conducted experiments on the male-speaker training set of the TIMIT database: 3180 sentences (318 spkr.) for training; 80 sentences (8 spkr.) for evaluation. Acoustic features were computed with HTK: 13 MFCCs including C_0, 25-ms window, 10-ms step. Each SHMM state had a linear segment trajectory and gamma duration model. We used 49 monophone SHMMs and 1400 triphone SHMMs, a phone-level bigram language model, and simple back-off for training and recognition with triphones.

Effect of partitioning: Reductions in output-probability calculation from pre-cost partitioning were: 21.8 % and 24.4 % for supervised training with monophones and triphones respectively; 6.3 % and 6.0 % for embedded training; 42.2 % and 47.6 % for supervised recognition (i.e., classification); 0.4 % and 0.1 % for embedded recognition. The 6 % reduction for embedded training is notable, and partitioning was especially effective in *supervised* training and recognition with both model types. These gains come without any loss of recognition accuracy.

Effects of beam pruning: Fig. 3 compares effectiveness of SN and EN beam pruning with monophone and triphone models, in terms of accuracy and load with respect to the beam threshold. As pruning increases (threshold drops), the accuracy shows a critical point at which it fell away for both SN and EN prun-

ing, although the threshold is higher for SN pruning here. So, SNP should not be pruned as severely as ENP, despite the greater computational benefits it can bring. The choice of θ^{S} and θ^{E} is a trade off between lower computational complexity and higher recognition performance, whose optimal adjustment further investigation. Table 1 shows the result of combining SN and EN pruning with selected values of θ^{S} and θ^{E}, which yields extra computational reduction: 9.7% and 79.8% cuts in output-probability calculation for monophones and triphones, respectively.

Conclusions. To reduce computation in SHMM decoding, we introduced an efficient algorithm with pre-cost partitioning and two types of beam pruning. The experimental results using these pruning strategies showed that partitioning can reduce the computation by 20–25% in supervised training, by 40–50% in phone classification, and by 6% in embedded training, for both monophone and triphone models without degrading recognition accuracy. Combination of SN and EN beam pruning reduced computation by 80% overall (by 3–4% compared with only EN beam pruning, as in conventional beam pruning [6]), at a cost of $\leq 0.1\%$ decrease in accuracy. We plan to use this segmental recognizer for future studies on ASR and synthesis.

Acknowledgements. Thanks to EPSRC for funding the Dansa project (GR/S85511/01), and to Martin Russell for sharing an implementation of the segmental decoder (v.3.4, Balthasar project).

References

[1] V. V. Digalakis, "Segment-based stochastic models of spectral dynamics for continuous speech recognition," Ph.D. dissertation, Boston Univ., MA, 1992.

[2] M. J. F. Gales and S. J. Young, "Segmental hidden Markov models," in *Proc. Eurospeech '93*, Berlin, Sep. 1993, pp. 1579–1582.

[3] F. Jelinek, "A fast sequential decoding algorithm using a stack," IBM J. Res. and Dev., Nov. 1969.

[4] D. Paul, "Algorithms for an optimal A* search and linearizing the search in the stack decoder," in *Proc. ICASSP*, Toronto, 1991, pp. 693–996.

[5] R. Haeb-Umbach and H. Ney, "Improvements in time-synchronous beam search for 10000-word continuous speech recognition," *IEEE Trans. on Speech and Audio Processing*, vol. 2, pp. 353–356, 1994.

[6] M. J. Russell, "Reducing computational load in segmental hidden Markov model decoding for speech recognition," *IEE Electronics Letters*, vol. 41, no. 25, pp. 1408–1409, Dec. 2005.

Authors' affiliations: Y. Shiga and P. J. B. Jackson (Centre for Vision, Speech and Signal Processing, School of Electronics and Physical Sciences, University of Surrey, Guildford, GU2 7XH, United Kingdom, email: Y.Shiga@surrey.ac.uk)

Figure captions:

Fig. 1 Calculation of SNP (left) by finding the best predecessor, and of ENP (right) by selecting the best segment duration.

Fig. 2 Identification of valid start and end nodes by partitioning.

Fig. 3 Phone recognition accuracy (left) and computational load (right) versus beam threshold for SN and EN pruning.

Table captions:

Table 1 Recognition accuracy (%) and computational load (output probability calculations) with SN and EN pruning.

Figure 1:

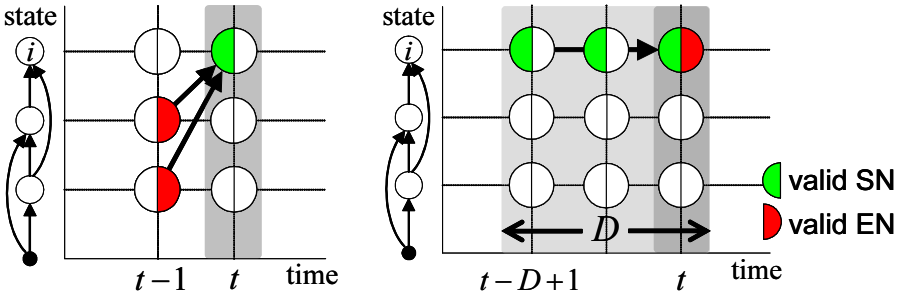

Figure 2:

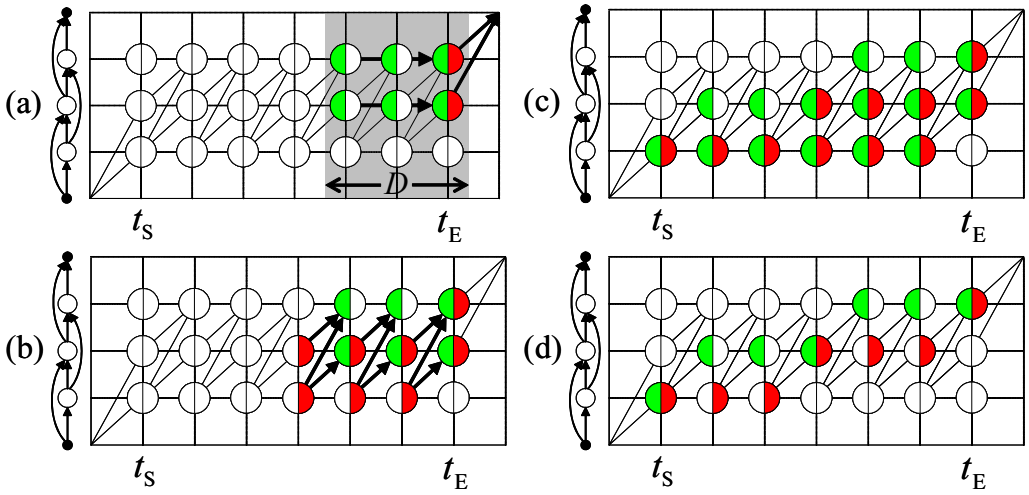

Figure 3:

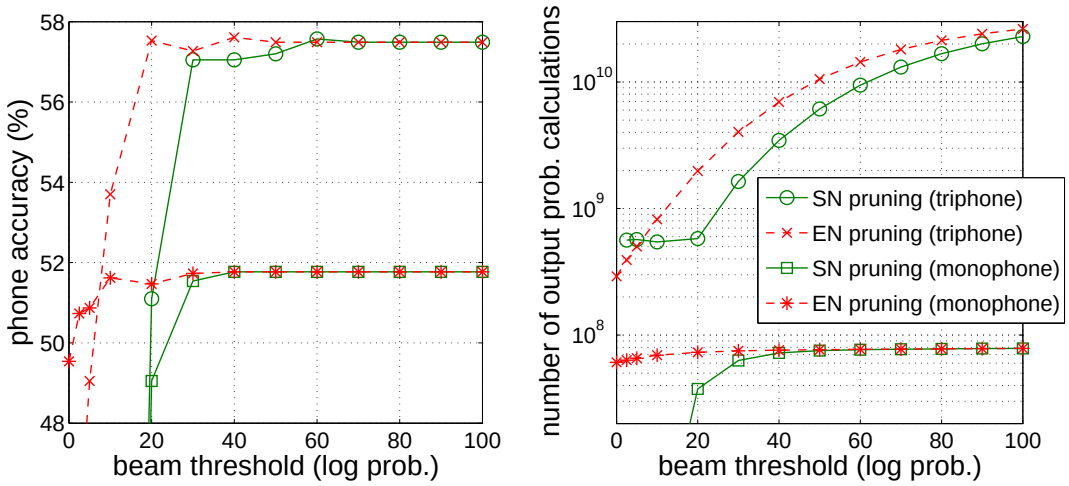
phone accuracy (%)
beam threshold (log prob.)
number of output prob. calculations
beam threshold (log prob.)
SN pruning (triphone)
EN pruning (triphone)
SN pruning (monophone)
EN pruning (monophone)

Table 1:

	monophone		triphone	
strategy	accuracy	load	accuracy	load
no pruning	51.8	7.96 $\times 10^7$	57.5	33.4 $\times 10^9$
SN (θ^{S}=40)	51.8	7.24 $\times 10^7$	57.6	9.45 $\times 10^9$
EN (θ^{E}=30)	51.7	7.51 $\times 10^7$	57.6	6.96 $\times 10^9$
SN + EN	51.7	7.19 $\times 10^7$	57.6	6.75 $\times 10^9$