LEARNING STATE-DEPENDENT STREAM WEIGHTS FOR MULTI-CODEBOOK HMM SPEECH RECOGNITION SYSTEMS

I. Rogina, A. Waibel
University of Karlsruhe, Postfach 6980, 76128 Karslruhe, Germany
Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA

ABSTRACT

Many speech recognition systems [Lee88], [Shi85], [Hua92], use multiple information streams to compute HMM output probabilities (e.g. systems based on semicontinuous or discrete HMM's use one codebook for cepstral coefficients, and another one for delta cepstral coefficients). The final score is a weighted sum of the contributions of every stream. These weights can be found empirically and usually the same set of weights is used for every acoustic model. There is reason to believe that there are features which are more important for some acoustic models than for others. Especially, one would expect the beginning and ending segment of a phoneme to be more context dependent than the middle part, so in that case the probability estimator of the speech recognizer should put more emphasis on the delta-spectrum than on the spectrum. Experiments [Shi85], [Boc93], have shown that spectral or cepstral coefficients are more important than their derivatives and more important than power or delta-power coefficients. In this paper we propose an algorithm for learning individual stream weights for every HMM state. Since these individual weights are a superset of the stream-only dependent weights, they can reproduce the results of the stream-only dependent weights and, additionally, discriminate between HMM states. Thus, the recognition performance must improve.

1. TRAINING

Consider a system which uses n information streams. At a given time t, and for a given HMM-state S, each information stream i will compute a class-dependent probability $P_i(x_t|S)$, where x_t is the speech signal at time t. The overall probability for state S at time t is then $P(x_t|S) = \prod_{i=1}^{n} P_i(x_t|S)^{\alpha_i}$, where α_i is a weighting factor for the i-th stream (sometimes called codebook exponent). We also demand $0 \leq \alpha_i \leq 1$ and $\sum \alpha_i = 1$, to make sure that $P(x_t|S)$ is a probability. We suggest to use a different α_i for each HMM state S, thus defining

$$P(x_t|S) = \prod_{i=1}^{n} P_i(x_t|S)^{\alpha_i(S)} \qquad (1)$$

Here $\alpha_i(S)$ reflects the importance of stream i for the HMM state S. A large $\alpha_i(S)$ will let stream i have a greater impact on the overall probability of S. Usually HMM-based speech recognition systems use $-\log P(x_t|S)$ instead of $P(x_t|S)$ for computing state sequence probabilities. This means that

$$-\log P(x_t|S) = \sum_{i=1}^{n} -\log P_i(x_t|S) \cdot \alpha_i(S) \qquad (2)$$

Now, we would like to train $\alpha_i(S)$ such that some optimization criterion is met. During training, the forced aligment procedure finds an optimal path. If at some time t the HMM-state on the optimal

0-7803-1775-0/94 $3.00

path, C, did not get the highest probability of all states, then there is some other state B that has the highest probability, and we will adjust our parameters to increase the probability for C and to decrease the probability for B. If the state on the optimal path already has the highest probability, no training will occur at all. Now, let $c_i(t)$ be the contribution of stream i to the score for the correct state C at time t, and let $LP_t(\alpha, C) := -\log P(x_t|C)$, so $c_i(t) := -\log P_i(x_t|C)$ and $LP_t(\alpha, C) = \sum_{i=1}^{n} c_i(t) \cdot \alpha_i(C)$. Let $b_i(t)$ be the contribution of the stream i to the score for the best state B at time t (i.e. the state with the highest probability). This means that $\sum_{i=1}^{n} c_i(t) \cdot \alpha_i(C) \leq \sum_{i=1}^{n} b_i(t) \cdot \alpha_i(B)$. The goal of the training procedure is to modify $\alpha_i(B)$ and $\alpha_i(C)$ such that $LP_t(\alpha, C)$ decreases and $LP_t(\alpha, B)$ increases. For that, we need to compute the derivative of $LP_t(\alpha, S)$ with respect to $\alpha_i(S)$. The update rule will then be:

$$\alpha_j(B)^{updated} = \alpha_j(B) + \lambda \cdot \frac{d\, LP_t(\alpha, B)}{d\, \alpha_j(B)} \quad (3)$$

$$\alpha_j(C)^{updated} = \alpha_j(C) - \lambda \cdot \frac{d\, LP_t(\alpha, C)}{d\, \alpha_j(C)} \quad (4)$$

We can easily see, that in the general case the updated system will produce a higher probability for the correct Viterbi-path (or for some given labels). Note that the partial derivative $(\frac{\partial\, LP_t(\alpha, S)}{\partial\, \alpha_i(S)})$ will not yield the correct result since the $\alpha_i(S)$ and $\alpha_j(S)$ are not independent from each other because of the above mentioned summation constraint. So any gradient descent step must result in a set of $\alpha_i(S)$'s which meet this constraint. Without this constraint the probability of a state S could be increased most, by increasing all $\alpha_i(S)$, which we don't want because we would like to discriminate between different features.

An infinitesimal step from $A(S) = \langle \alpha_1(S), \ldots \alpha_n(S) \rangle$ to $A'(S) = \langle \alpha'_1(S) = \alpha_1(S) + \delta_1, \ldots \alpha'_n(S) = \alpha_n(S) + \delta_n \rangle$ could thus be defined as $\delta_j := \epsilon$ and $\delta_i := \epsilon \cdot \frac{\alpha_i(S)}{1-\alpha_j(S)}$ for $i \neq j$. This step definition modifies $\alpha_j(S)$ by ϵ, while all the other $\alpha_i(S)$ are modified to meet the summation constraint and to keep their mutual ratios unchanged. Other step definitions are possible. Now we are numerically computing the derivative of a function, whose domain is an n-dimensional space, only for values on a hyperplane of that space. With this step definition we find: (the differential d^* will be used to denote that the derivative is restricted to the hyperplane of the feature space which meets the summation constraint):

$$\frac{d^* \sum_{i=1}^{n} b_i(t) \cdot \alpha_i(B)}{d^* \alpha_j(B)} = \lim_{\epsilon \to 0} \frac{LP_t(\alpha', B) - LP_t(\alpha, B)}{\epsilon}$$

$$= \lim_{\epsilon \to 0} \left(\frac{1}{\epsilon} \left(\sum_{i=1}^{n} (\alpha'_i(B) b_i) - \sum_{i=1}^{n} \alpha_i(B) b_i \right) \right)$$

$$= b_j + \frac{LP_t(\alpha, B) - b_j \alpha_j(B)}{\alpha_j(B) - 1} \quad (5)$$

Here, we have ignored that the actual size of the infinitesimal step is somewhat greater than ϵ, resulting in a somewhat greater denominator. But since we will use a stepsize λ for performing the gradient descent, λ will subsume this difference. For a simple two-feature system, eq. (5) results in:

$$\frac{d^* LP_t(\alpha, B)}{d^* \alpha_1(B)} = b_1 + \frac{\alpha_2(B) b_2}{\alpha_1(B) - 1} = b_1 - b_2 \quad (6)$$

2. IMPLEMENTATION ISSUES

We have to consider the possibility of one feature always being the best for some model. In this case its feature weight would rise with every update. Nothing would stop it from growing greater than 1.0 and thus pushing the other feature weights below 0.0. This, obviously, is an unwanted effect. One ad hoc solution to this problem could be to simply define some floor and ceiling value to keep the feature weights in their bounds. We used two other approaches. One is to apply a sigmoid to $\alpha_i(B)^{updated}$ and to update $\alpha_i(B)$ according to $\alpha_i^{updated} = \text{sigmoid}(\alpha_i(B) - \lambda \cdot \frac{d^* LP_t(\alpha, B)}{d^* \alpha_i(B)})$

Is there reason to believe that one feature F_a is more important than some other feature F_b in training iteration k and that F_b is more important in iteration $k+1$? In fact, one should expect little change from one iteration to the next, even if the segmentations of some sentences differ. So we can expect $\alpha_i(B)^{k+n}$, the value of

$\alpha_i(B)$ after iteration $k+n$ to be approximately $\alpha_i(B)^t - n \cdot \lambda(d^* LP_t(\alpha, B)/d^* \alpha_i(B))$ (if no sigmoid is applied). We have found that the differences from iteration to iteration are in fact so small that this approximation is valid, which suggested a second solution to the above mentioned problem, namely to run simply one or two iterations with a large stepsize, or alternatively to use a cross validation mechanism to decide what number of iterations (i.e. what stepsize λ) is best.

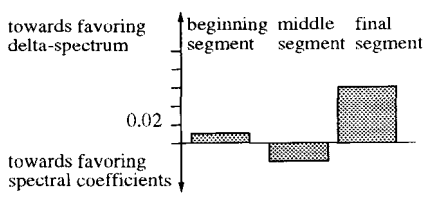
3. EXPERIMENTS

We have performed experiments on the English Conference Registration Task (CR) [Woo92] and the Resource Management Task (RM), using the speech recognizer [Sch92] of the JANUS Speech to Speech Translation System [Wai91]. The recognizer computes class conditional probabilities for a DTW search. It uses an LVQ-trained 50-cluster codebook for each of 48 phones and probability distributions over these clusters for 1300 context dependent phones. For CR we used about 1000 sentences from 60 speakers to train the system. The tests were performed on a test set with a perplexity of about 7 and a vocabulary size of about 400 words. The RM system was trained on about 3000 sentences from about 100 male speakers and tested on the male portion of the February 1989 official DARPA test set with a wordpair grammar of perplexity 60, and a vocabulary of 1000 words. (The tests on the female portion usually yield the same or slightly better results.) In the training phase, no other parameters besides the feature weights were modified. So the improvement in performance is due only to the training of the feature weights. The following table shows the results:

Error reduction by stream weights training

Task	error rate of baseline system	2 iterations of stream weight training	error reduction
CR	6.8%	3.6%	47%
RM	9.7%	7.8%	20%

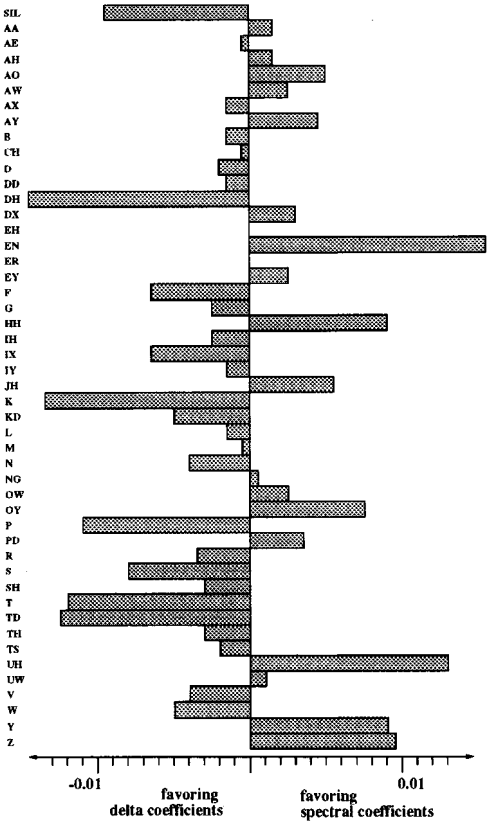
The feature weights used in the experiments from which we extracted the original error rate were fixed at 0.55 for feature F_a (16 mel scale spectral coefficients) and 0.45 for feature F_b (16 delta spectral coefficients). This is the same ratio as the one used in [Lee88], [Hua92]. We have verified that these values are in fact the best choice for fixed feature weights in our system. All experiments were performed without using a sigmoid, the stepsize λ was chosen to modify the feature weights by no more than $+/-0.1$ per iteration. In agreement with other results [Shi85], we have found that the optimal models of different phonemes depend on different features more heavily. Even different parts of the same phoneme rely on different features. The following figure shows the development of the feature weights for the beginning, middle, and final segments averaged over all phonemes:



We can see that the weights of the beginning and the final segments were trained to put more emphasis on the delta spectral features, while the weights for the middle segments favored the spectral features. This observation, however, is not enough convincing to say that generally all the phone boundary states are more dependent on delta coefficients. Except for some few phonemes we have not found the relative importance of the different streams to be significantly homogenous over some class of phonemes like vowels or consonants. Since we were using generalized triphones, there is also no reliable way to tell whether some class of contexts is tending more towards one feature or towards the other.

The following diagram shows the gradient for each of the 48 phonemes of our system, averaged over all contexts and HMM states. We can see that some phonemes like S and SH and even the silence

phoneme tend more to favoring delta-coefficients, although one might expect that these phonemes' acoustics are rather static and less context dependent. The explanation for this observation is the fact that delta coefficients do model the dynamics of a signal but not necessarily the context. A stable context-independent signal like silence also has very stable delta coefficients.



4. FUTURE WORK

So far we have only performed experiments with two streams. We believe that the proposed approach will be even more fruitful for systems with greater numbers of features, like e.g. delta-spectral-coefficients, delta-delta-spectral-coefficients, power, delta-power and delta-delta-power. Certainly, one may expect different results for other pairs of feature and delta-feature. Experiments with non-generalized triphones including cross-word triphones will give us more information about the dependence of the stream weights on the different types of contexts.

REFERENCES

[Boc93] E.L Bocchieri and J.G. Wilpon, "Discriminative feature selection for speech recognition", Computer Speech and Language, Vol. 7, pp. 229-246 (1993)

[Hua92] Huang, X., Alleva, F., Hayamizu, S., Hon H., Hwang, M., and Lee, K., "The SPHINX-II Speech Recognition System", Technical Report CS-92-112, Carnegie Mellon University, 1992

[Lee88] K.F. Lee, "Large Vocabulary Speaker-independent Continuous Speech Recognition: The Sphinx System", PhD Thesis, Carnegie Mellon University, April 18, 1988, Report CMU-CS-88-148

[Shi85] Shikano, K., "Evaluation of LPC Spectral Matching Measures for Phonetic Unit Recognition", Technical Report, Computer Science Department, Carnegie Mellon University, 1986

[Sch92] Otto Schmidbauer, Joe Tebelskis, "An LVQ based Reference Model for Speaker-Adaptive Speech Recognition", ICASSP 92, Vol. I, pp. 441-444

[Wai91] Alex Waibel et. al., "JANUS: A Speech to Speech Translation System Using Connectionist and Symbolic Processing Strategies", ICASSP 91, Vol. II, pp. 793-796

[Woo92] Cindy Wood, "The Conference Registration Task", Technical Report, Computer Science Department, Carnegie Mellon University, 1991