COMBINATION OF HIDDEN MARKOV MODELS WITH DYNAMIC TIME WARPING FOR SPEECH RECOGNITION

Scott Axelrod and Benoît Maison

IBM T.J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598, USA
{axelrod,bmaison}@us.ibm.com

ABSTRACT

We combine Hidden Markov Models of various topologies and Nearest Neighbor classification techniques in an exponential modeling framework with a model selection algorithm to obtain significant error rate reductions on an isolated word digit recognition task. This work is a preliminary investigation of large scale modeling techniques to be applied to large vocabulary continuous speech recognition.

1. INTRODUCTION

Increases in computational power, storage capacity, and training data available for use by automatic speech recognition (ASR) systems, combined with the perception that the performance of such systems has reached a plateau motivate us to consider modeling strategies for speech recognition that, while more resource intensive, have the potential to obtain significant reductions in error rate.

One old approach to acoustic modeling uses Dynamic Time Warping (DTW) techniques to match segments of test utterances to stored training data. DTW systems can capture long-range dependencies [1] in the acoustic data, and can potentially adapt to differences in gender, speaker, and accent by pinpointing, at decode time, similar data in the training set [2].

It is unrealistic, however, to expect to abandon the great efficiency and robustness offered by the smoothed out statistical HMM models. Besides, HMM models can also capture long distance dependences if they have an appropriate topology. Early work on fenonic baseforms [3] indicates such potential.

In this paper we take the modest steps of studying combinations of whole world HMM models with dynamic time warping procedures. One novelty of our approach is that we combine collections of such models using maximum entropy, a.k.a. exponential modeling.

2. EXPERIMENTAL SETUP

Although our interest lies in large vocabulary continuous speech recognition, all experiments described in this paper have been done on single digits. We extracted them from continuous speech data by doing forced alignments (using a state-of-the-art HMM) of the reference scripts. We collected the sequences of feature frames corresponding to the 11 digit words (from 'zero' to 'nine', with the addition of 'oh'). The features are 39-dim vectors obtained by splicing 9 consecutive 13-dim cepstra and applying LDA followed by MLLT [4]. A subset of IBM internal telephony-band training data yielded a total of 627806 sequences. In most experiments, we used the first 5K samples of each word as our training set, and another set of 1K samples as a held out set. We also extracted in the same manner a test set of approximately 500 samples of each word from an independent test set. The training data consists of large vocabulary utterances from a large variety of speakers and accents. The test data consists of connected digits strings. Both sets include landline and cellphone data.

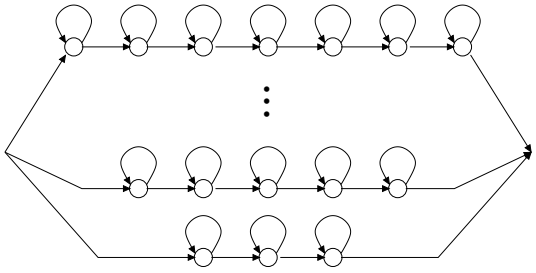

Fig. 1. Multi-branch HMM.

We are aware that recognizing digits when the word boundaries are known is a much easier task than continuous speech recognition (even though we give up useful phonetic context information). However, for the purpose of this paper, this setup gave us the greatest flexibility in trying various algorithms, models and model combinations.

3. HIDDEN MARKOV MODELS

We trained several HMM models for each word in the vocabulary both using the small training set "5K" (with $5K$ samples per word) and the full training set (with an average of $57K$ samples per word). All models have a topology similar to that depicted in Figure 1, namely a set of branches, each of which is a simple linear HMM with self loops. The observation probability distribution associated with each state is a full covariance Gaussian mixture model. We built models in which the number of states in each branch was fixed independently of the data as well as models where the number of states varied with the branch.

We will denote the models in the fixed length case as $F(B, N, S)$, where B is the number of branches, N is the number of Gaussian components for each of the state models, and S is the training set size. For those models, all branches of a graph for word w have fixed length $L(w)$ equal to the number of arcs in a standard linear HMM model. For the variable length model $V(B, N, S)$ with B branches, we took the number arcs for branch

b ($1 \leq b \leq B$) to be $b * L(w)$.

The observation distributions and the HMM transition rate were both trained to convergence by the Baum-Welch algorithm with no approximations. All the models were trained from a flat start without reference to any auxiliary data such as baseforms or a reference model. For the seed models we use uniform transition probabilities and a choice of initial hard state alignments. For efficiency, the seed Gaussian mixture models for each states was obtained by first applying a few rounds of k-means to the data aligned to the state and then training with the usual Expectation Maximization algorithm.

For a given model type we actually train Hidden Markov models on the training data for each of the 11 digit words separately. We assume a uniform probability distribution for each of the words so that the word hypothesized for a model type M ($F(B, N, S)$ or $V(B, N, S)$), given an acoustic data sequence X, is simply the one that maximizes the probability $p(X|w, M)$ of the HMM of type M trained on data for word w.

Word error rates on the heldout and test sets for a variety of models types built on the small training set are reported on in Table 1, while Table 2 gives results for models trained using all the training data. The reader will note that the simple linear (one-branch) HMM models of standard length obtain the best or near best performance, although he or she is cautioned not to draw too strong a conclusion from this fact since our purpose in obtaining these models was to obtain an eclectic, and hopefully complementary, variety of models to be combined later, rather than to search for the best performing individual models. (Although we do believe the best linear models in Tables 1 and 2 to be a relatively high quality baseline.)

model name	number branches	branch length	number gauss	heldout error	test error
F(5,1,5K)	5	fixed	1	2.43%	1.67%
F(10,1,5K)	10	fixed	1	2.48%	1.63%
F(1,5,5K)	1	fixed	5	2.79%	1.65%
V(3,2,5K)	3	vary	2	3.1%	1.9%
V(3,1,5k)	3	vary	1	3.4%	2.1%
V(3,3,5K)	3	vary	3	3.4%	2.3%
V(2,1,5K)	2	vary	1	3.5%	2.5%

Table 1. Word error rates for isolated digit recognition on heldout and test data for Hidden Markov Models with B branches, N Gaussians per state, and training set size S equal to $5K$ samples per word. The model $F(B, N, S)$ has a fixed number of states per branch whereas $V(B, N, S)$ has a varying number of states per branch. Models are sorted in order of increasing error on the heldout set.

4. NEAREST NEIGHBORS USING DTW

Dynamic Time Warping is a technique that allows the computation of distances between sequences of speech frames of different lengths. The name covers a broad range of variations. The nature of the dynamic programming (DP) recursion and the frame distance metric are two choices that we examine in this section. Another far-reaching decision is whether to keep all the training data, or only some representative samples. We follow the former approach, since it is most naturally suited to the use of

model name	# branches	branch length	# gauss	heldout error	test error
F(1,20,57K)	1	fixed	20	0.46%	0.48%
F(1,10,57K)	1	fixed	10	0.61%	0.59%
F(20,2,57K)	20	fixed	2	0.65%	0.72%
V(3,2, 57K)	3	vary	2	0.65%	0.91%
F(1,5, 57k)	1	fixed	5	0.72%	0.79%
F(5,2, 57k)	5	fixed	2	0.72%	0.97%
F(10,1,57k)	10	fixed	1	0.76%	0.99%
F(5,1, 57k)	5	fixed	1	0.82%	1.15%
V(3,1, 57k)	3	vary	1	0.89%	1.27%
V(2,1, 57k)	2	vary	1	0.97%	1.27%

Table 2. Word error rates for models trained on all training data ($57K$ samples/word on average).

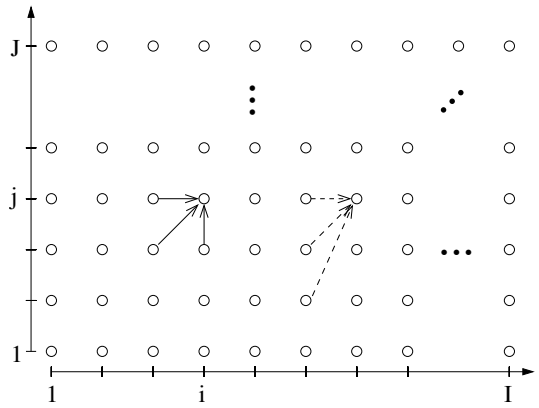

Fig. 2. Dynamic Time Warping Recursions: Symmetric (solid line) and Itakura constraint (dashed line).

"non-verbal" [2] information. We also investigate the use of multiple Nearest Neighbors (kNN) [5] to improve classification performance.

The dynamic programming lattice is illustrated in Figure 2. The first sequence (test) consists of I frames x_i, and the second (reference) sequence (template) has J frames y_j. We consider two recursions: the symmetric one and the Itakura constraint [6]. In the symmetric case, the score for matching the first i frames of sequence 1 to the first j frames of sequence 2, is recursively defined as

$$S_{i,j} = d_{i,j} + \min\{S_{i-1,j}, S_{i,j-1}, S_{i-1,j-1}\},$$

where $d_{i,j}$ is the distance from x_i to y_j, and

$$S_{1,1} \quad = \quad d_{1,1}.$$

$S_{I,J}$ is the total DTW cost. In the Itakura case, we have

$$S_{i,j} = d_{i,j} + \min\{S_{i-1,j}, S_{i-1,j-1}, S_{i-1,j-2}\}.$$

We consider several definitions of the frame distance $d_{i,j}$: $|x_i - y_j|$, $\|x_i - y_j\|^2$, $|C^{-\frac{1}{2}}(x_i - y_j)|$and $(x_i - y_j)'C^{-1}(x_i - y_j)$. The matrix C, is estimated as follows. For each pair of samples belonging to each word in the vocabulary, the best warping (the sequence of pairs (i, j)) is computed using the plain Euclidean distance. C

Metric	Non-normalized	Duration-normalized
Mean absolute diff.	5.00%	-
Mean abs. with C	3.98%	3.59%
Mean sqr. with C	3.59%	3.18%

Table 3. Comparison of frame distance metrics and effect duration normalizations with symmetric DP recursion (Word error rates on heldout set).

kNN technique	Symmetric with Norm.	Itakura constr.
1-best	3.18%	2.68%
voting	2.25%	1.92%
soft-voting	2.08%	1.86%
sum-of-exp.	2.02%	1.81%

Table 4. Effect of DP recursion and voting (Word error rates on heldout set). Mean squared distance and $C^{-\frac{1}{2}}$ transform are used.

is estimated as the average of $(x_i - y_j)(x_i - y_j)'$ over all pairs of samples of every word. Since C is like a covariance matrix, the latter two types of the frame distance can be efficiently computed by first applying the transform $C^{-\frac{1}{2}}$ (analogous to PCA with variance normalization) to all test and reference frames, then computing the mean absolute or mean squared distance. The first two rows of the first column of Table 3 show the effect of the transform $C^{-\frac{1}{2}}$ on the classification error when the mean absolute distance metric is used. We observed a similar gain for the mean squared distance.

The symmetric DP algorithm permits the matching of sequences of any length. However, since each frame of each sequence must be matched to some frame of the other sequence, the total DTW cost is dominated by the length of the longest of the two sequences. This effect makes it harder to compare the matching cost of a test sequence against templates of different durations. We apply the following correction: $d'_{I,J} = d_{I,J}(\frac{I}{\max(I,J)})^{\alpha}$ in an attempt to make the cost $d'_{I,J}$ depend on I. We use $\alpha = 0.7$ but a wide range of values give similar results. The dependence on I is irrelevant for isolated word classification, but is important in a lattice rescoring context, when paths with different numbers of words (but totaling the same number of frames) have to be compared. The last two rows of Table 3 show a significant improvement when duration normalization is used (second column). They also indicate that the mean squared distance metric (with PCA-like transform) outperforms the mean absolute distance.

The Itakura recursion, since it matches each test frame exactly once, does not require any duration normalization. It also gives the best performance, as shown by the columns of Table 4.

The rows of Table 4 report the classification error rates obtained with different kNN strategies. The first row corresponds to $k = 1$, i.e. the 1-best hypothesis considered above. The second row corresponds to voting with $k = 10$. Soft-voting is similar to voting, except that each entry in the top k carries the weight $\exp^{-\beta d}$ where d is its matching score. We used $\beta = 10^{-4}$ in these experiments. *Sum-of-exponentials* differs from soft-voting in that the k nearest templates of every candidate word are collected, instead of the global top k nearest neighbors. The final score of word w is $S_w = \sum_{n=1}^{k} \exp^{-\beta d_w^n}$ where d_w^n is the matching cost of the nth nearest template belonging to word w. Simple voting brings a large improvement, while soft-voting and *sum-of-exponentials* show modest additional gains.

Error rates on the test set will be reported in section 5.2 for the best DTW system on the heldout set (which uses Itakura constraints, mean square frame distances, and *sum-of-exponentials* scoring) taken alone and in combination with HMM systems.

5. MODEL COMBINATION

Our goal in this section is to combine subsets of word classifiers discussed above to see if they provide complementary information. Any word classifier C considered in the previous sections hypothesizes the word W for a given acoustic data sequence $X = (x_1, ..., x_I)$ to be the word that maximizes a score function $f_C(X, W)$. For an HMM model of type C, we take $f_C(X, W)$ to be the log probability $\log p_C(X|W)$. We take the feature $f_{DTW}(X, W)$ to be the log of the *sum-of-exponentials* score.

Since our goal in the future will be to combine the models in a way that can be easily incorporated in a continuous word recognition system, it seems like a good idea to try to combine the models in a probabilistic fashion. A standard way to do that is to form a posterior probability distribution which is a log linear combination of feature functions:

$$p_\lambda(W|X) = \frac{e^{<\lambda_W, f(X,W)>}}{Z_\lambda(X)}, \quad (1)$$

$$f(X, W) = (f_{C_1}(X, W), ..., f_{C_T}(X, W)), \text{ and} \quad (2)$$

$$Z_\lambda(X) = \sum_{W'} e^{<\lambda'_W, f(X,W')>} . \quad (3)$$

Here, $f(X, W)$ is a "feature vector" of dimension T, where T is the number of model types (DTW or HMM based classifiers) being combined. The denominator $Z_\lambda(X)$ is simply a factor to normalize p_λ as a posterior probability distribution for W.

Although other non-linear functions of the basic scores could be used instead of, or in addition to, the ones we have chosen here, the choice of log probabilities (and log *sum-of-exponential* scores) allows us to loosely interpret the probability (1) as a weighted geometric mean of separate distributions. Consistent with this point of view, we constrain all the weight to be positive, although the results do not vary too much if negative weights are allowed.

In section 5.2 we will report results of maximum likelihood classification according to distributions of the form (1), where the weights $\lambda = \{\lambda_W\}$ are chosen so as to maximize likelihood on the heldout data set described in section 2. Readers familiar with the language modeling literature may recall that exponential models trained to maximize posterior probability are also commonly referred to as maximum entropy models [7]. This is because the distribution (1) for the maximizing λ equals the distribution which has maximum (conditional) entropy subject to a certain constraint. In the language modeling case the features are typically all discrete and there are specialized algorithms for performing the optimization. In our case the features are continuous. We find that the optimization proceeds readily using a generic quasi-newton search algorithm with care taken to handle the numerics when summing exponential of large (and occasionally infinite) values.

5.1. Model Selection

How can we make a judicious choice of which models to combine that will reduce the total computational time required and also reduce the risk of overtraining? This problem is a special case of

	without DTW		with DTW	
combination	heldout	test	heldout	test
single	2.43%	1.67%	1.81%	1.45%
all	2.10%	1.40%	1.96%	1.42%
select	2.09%	1.42%	1.31%	1.09%

Table 5. Word error rates on training and heldout data for various maximum entropy combinations of models trained on the small training data set both including DTW (last two columns) and not including DTW (first two columns). Types of combinations are: "single" – best single HMM / DTW used alone; "all" – all HMM / all HMM+DTW; "select" – results of algorithm in Figure 3.

what is known as the feature selection problem [8, 9]. In this paper we propose a profoundly simple approach to feature selection which captures an idealized strategy for making progress in general. Code for the algorithm (in the matlab language) is presented in Figure 3. The algorithm simply adds features one at a time, at each step adding the greedy choice of feature whose addition give the greatest improvement in error rate on the heldout set. The algorithm terminates when no feature can be added which improves heldout error rate. The function `models_to_keep` in Figure 3 takes as input a list of `all_models` from which we want to select a `kept` subset. Models are represented as powers of 2 and model combinations are represented as integers equal to the sum of the binary powers for the models in the combination. The function `wer_train` takes in a list of model combinations (a.k.a. a vector of integers) and outputs the list of corresponding error rates. The function `delete` deletes an element from a list.

```
function kept = models_to_keep(all_models)
  candidates  = all_models;
  kept        = 0;
  while(length(candidates)>0)
    [wer_new newI] = ...
      min( wer_train(kept+candidates) );
    if ( wer_new >= wer_train(kept) )
      break;
    end
    kept        = kept + candidates(newI);
    candidates  = delete(candidates,newI);
  end
```

Fig. 3. Algorithm for selection of models to combine.

5.2. Results

Table 5 reports the result on the heldout data (used for training exponential model weights) and the test data set for maximum entropy combinations of various subsets of the DTW system and the seven HMM models of Table 1, all of which were trained on the small training data set. The exponential model features (3) were taken to be logs of HMM likelihoods and logs of DTW *sum-of-exponentials* scores. The first row reports error rates for individual models; the second row reports result obtained by combining all models (with and without DTW); and the last row reports results for the subset of models chosen by our model selection algorithm based on performance on the heldout set.

The following can be observed from the test results in Table 5: (1) it helps to combine HMM systems; (2) the DTW system in isolation performs better than any individual HMM system and in fact about as well as any combination of HMM systems; and (3) the combination of both DTW and HMM systems picked by the model selection algorithm was the overall best system and significantly outperforms the combination of all models.

Not shown in the table is the fact that the greedy feature selection algorithm found the model combination with absolute minimum heldout error rate (out of 128 possible combinations), both in the case when DTW was not included as well as when it was required to be included. For the sake of any curious readers, we note that, on the heldout set, the best combination of HMM models without the DTW system consisted of the first five models in Table 2, while the best model subset to combine with the DTW system consisted of the model $F(10, 1, 5K)$ by itself.

Finally we report that word error rate on the test data of the DTW system using all of the training data is 1.13%. This is significantly worse than the best HMM system. We did not find any combination of system that had better performance than the single best HMM system (the first line of Table 2).

6. CONCLUSIONS

We have shown that Nearest Neighbor techniques can be significantly enhanced by distance modeling, smoothed voting, and normalization. We have also seen significant error rate reductions by combining DTW and multi-branch HMM models using maximum entropy techniques and a greedy model selection algorithm. We view this work as a first step towards large scale models for large vocabulary speech recognition.

7. REFERENCES

[1] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[2] M. De Wachter, K. Demuynck, D. Van Compernolle, and P. Wambacq, "Data driven example based continuous speech recognition," in *Proc. Eurospeech*, Geneva, Switzerland, September 2003.

[3] F. Jelinek, *Statistical Methods for Speech Recognition*, The MIT Press, 1997.

[4] R. Gopinath, "Maximum likelihood modeling with gaussian distributions for classification," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Seattle, WA, 1998.

[5] K. Fukunaga, *Statistical Pattern Recognition*, Academic Press, 1990.

[6] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 67–72, Feb 1975.

[7] A. Berger, S. Della Pietra, and V. Della Pietra, "A maximum entropy approach to natural language processing," *Computational Linguistics*, , no. 22(1), 1996.

[8] D. D. Lewis, "Feature Selection and Feature Extraction for Text Categorization," in *Proceedings of Speech and Natural Language Workshop*, San Mateo, California, 1992, pp. 212–217, Morgan Kaufmann.

[9] G. J. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*, Wiley, 1992.