Improved HMM Models for High Performance Speech Recognition

Steve Austin, Chris Barry, Yen-Lu Chow
Alan Derr, Owen Kimball, Francis Kubala, John Makhoul
Paul Placeway, William Russell, Richard Schwartz, George Yu

BBN Systems and Technologies Corporation
Cambridge, MA 02138

ABSTRACT

In this paper we report on the various techniques that we implemented in order to improve the basic speech recognition performance of the BYBLOS system. Some of these methods are new, while others are not. We present methods that improved performance as well as those that did not. The methods include Linear Discriminant Analysis, Supervised Vector Quantization, Shared Mixture VQ, Deleted Estimation of Context Weights, MMI Estimation Using "N-Best" Alternatives, Cross-Word Triphone Models. While we have not yet combined all of the methods in one system, the overall word recognition error rate on the May 1988 test set using the Word-Pair grammar has decreased from 3.4% to 1.7%.

1 Introduction

We considered several directions for trying to improve the recognition accuracy within the basic framework of the BYBLOS system. The various techniques can be reasonably grouped into three general topics: changing the underlying distance metric in the spectral space, optimizing the few weights that are used with the system, and improving the phonetic coarticulation model by adding cross-word triphone context. We introduce each of these areas below and discuss them in more detail in the body of the paper. Finally, we will present recognition results for a combination of two of the methods.

Even in a discrete HMM system, there is an underlying distance metric that is used to divide the spectral space into distinct regions. It has been suggested that it is possible to improve recognition accuracy by performing a linear discriminant analysis. We have also considered several methods of nonlinearly warping the spectral space as part of the vector quantization process. We classify these methods as "supervised clustering" techniques. In addition, we implemented the technique that has been called "tied mixture vector quantization" (Bellagarda, 1989) or semi-continuous densities (Huang, 1989).

In the BYBLOS system there are a number of system parameters that are fixed for all speakers based on intuitions and as a result of running a small number of tuning experiments. Among these are the weights for the different context-dependent models of phonemes and the relative weights for different feature sets (codebooks). While the weights chosen are certainly reasonable, on the average, it would seem inconsistent to estimate millions of HMM probabilities automatically while having a handful of parameters set manually. Therefore, we implemented a deleted estimation algorithm to estimate the context model weights and developed a new MMI technique for estimating the feature set weights automatically.

One obvious extension to context-dependent modeling (which was introduced in BYBLOS in 1984) is to model context between phonemes that are not in the same word. In fact, three research sites (Paul, 1989; Lee, 1989; Murveit, 1989) reported modeling triphone context across word boundaries at the February 1989 meeting. We have now implemented a similar algorithm in the BYBLOS system. However, due to remarks from other researchers that the changes to the training and recognition programs were extensive and difficult to implement, we chose to implement the effect by precompiling all of the models in such a way that we did not need to change either the training or recognition programs.

In sections 2 to 4 we describe the algorithms implemented under each of these areas along with results. In section 5 we present recognition results under several different conditions, including the test results for the October '89 test set.

2 Distance Measures and Supervised VQ

This section deals with techniques for improving the distance measure used in VQ, in particular, using linear discriminant analysis, and nonlinear supervised clustering techniques. In addition, we present results when we replace the discrete densities with shared-mixture densities.

2.1 Linear Discriminant Analysis

In our baseline system we compute 14 mel-frequency warped cepstral coefficients (c1-c14) every 10 ms directly from the speech power spectrum. These parameters are grouped in one codebook. These 14 parameters are then used to compute "difference" parameters, by computing the slope of a least squares linear fit to a five-frame window centered around each frame. The 14 slopes of this fit for the coefficients then make up the second set (codebook) of features. Finally, we use a third codebook that has the log rms energy and the "difference" of this energy. The energy parameter is normalized relative to a decaying running maximum, so as to be insensitive to arbitrary changes in amplitude. We divide the 30 features among three codebooks to avoid the training problem associated with high dimensionality.

The recognition group at IBM (Brown, 1987) has proposed using several successive frames jointly in order to model the joint density more accurately together with linear discriminant analysis (LDA) to reduce the number of dimensions. We have attempted to use LDA to find a better set of features that could then be divided into sets that would, in fact be more independent. In addition, we might hope that we would automatically find a more beneficial weighting on the different features than simple Euclidean distance (which is what we use in the VQ).

First, we needed to define several classes that we wanted to discriminate. We chose the (50 or so) basic phonemes as that set, under the assumption that these modeled most of the distinctions that must be made in large vocabulary speech recognition. We segment all of the training data into phonemes automatically using the decoder constrained to find the correct answer. The recognized segment boundaries are then used to assign a phoneme label to each frame. Second, we compute the within (phoneme) class and between class means and covariances. We use the generalized eigenvector solution to find best set of linear discriminant features. Third, we simply cluster and quantize the new features as usual. Alternatively, we can divide the new features up into a small number of codebooks in order to reduce the quantization error.

We performed experiments with several variations in the number of codebooks and assignment of linear discriminants to codebooks. However, the results (averaged over several test speakers) did not improve over the baseline 3-codebook condition described at the beginning of this section. We can draw two possible conclusions from these results relative to previous successes with this technique. First, while it might be possible to find a small number of discriminant directions that are important for a small vocabulary task - especially one with minimal pair differences - it may not be as easy in a large vocabulary task, where the important distinctions are many and also very varied. That is, any choice of discriminants that is better for some distinctions may be worse for others. Second, it is not clear that optimizing phonetic distinctions on single frames will help a recognition system that uses models of triphones.

2.2 Supervised Vector Quantization

Since the simple linear discriminants did not improve results, we chose to consider a more complex warping of the feature space. We classify the general area as *supervised clustering* or *supervised VQ*. The basic idea is that instead of finding a codebook that minimizes mean square error, without regard to phonetic similarity, we should be able to use the training data to generate a codebook that tends to preserve differences that are phonetically important, and disregard feature differences (even if they are large) that are not phonetically important. Thus we attempt to maximize the mutual information between the VQ clusters and phonetic identity. We describe two techniques below that seemed like they should accomplish this goal. While both methods were able to derive a codebook that was more closely related to phonetic distance, neither resulted in an improvement in overall continuous speech recognition accuracy.

2.2.1 Binary Division of Space

The first algorithm is most closely related to the nonuniform binary clustering algorithm that we use to derive an initial estimate for k-means clustering (Roucos,Makhoul,Gish 85). We label all the speech frames phonetically as described in the previous section. All the labeled frames are initially in one cluster. Then, we iteratively divide the clusters until we have the

desired number. One of the many clustering algorithms we tried is given below.

First we have a procedure to measure the entropy reduction that would result from dividing a single cluster into two:

1. estimate a single Gaussian for the frames with each phoneme label in the cluster.

2. in general there will be several different phoneme labels in the cluster. Identify the two most "prominent" phonemes within the cluster. The most effective measure for this was simply the phoneme with the most frames.

3. divide all data into two new clusters using these two guassian distributions.

4. compute the difference between the entropy of the phoneme labels in the original cluster, and the average entropy of the two new clusters, weighted by the number samples in each subcluster.

The outer loop repeatedly divides the cluster that will result in the largest enropy reduction.

1. Place all the labeled frames initially in one cluster.

2. Using the above procedure compute the potential entropy reduction that would be obtained upon for dividing each of the clusters.

3. Adopt the division that resulted in the largest entropy reduction.

4. Create two new clusters and measure the potential entropy reduction for dividing each of the two resulting clusters as described above.

5. If we have fewer than 256 clusters, go to (3)

The resulting hierarchical codebook was then used to quantize all of the training and test data. When we applied the above algorithm to a single set of features (say 14), we found only a minor improvement in the mutual information above the case for unsupervised k-means. When we used all the features in one codebook, there was a larger gain. However, as with LDA, there was no gain in the overall recognition accuracy.

2.2.2 LVQ2: Kohonen's Learning Vector Quantizer

The LVQ2 algorithm (Kohonen, 1988) was used very effectively in a phoneme recognition system (McDermott, 1989). The algorithm amounts to a discriminative training of the codebook means to maximize recognition of frame the labels.

As before, we start with the set of phonemically labeled frames. Then we use the binary and k-means algorithm to divide the feature vectors from each phoneme into several clusters. We made the number of clusters for each phoneme proportional to the square root of the number of frames in that phoneme, such that the total number of clusters was 256. Each cluster has the name of the phoneme data in it. Then, we use LVQ2 to jiggle the means to optimize frame recognition.

For each feature vector:

1. find the nearest two clusters

2. if the nearest cluster is from the wrong phoneme and the second nearest is the correct phoneme, shift the mean of the correct cluster toward the feature vector in question and shift wrong cluster mean away.

The above algorithm is iterated until convergence (which requires some care). As suggested in the reference, we used several adjacent speech frames together as a longer feature vector. This resulted in significantly higher phoneme-frame recognition rates, both from the k-means initial estimate, and after improvement with LVQ2.

The LVQ2 algorithm was found to improve the frame recognition accuracy significantly (from 48% to 70%) on the training set, particularly for a large number of dimensions. However, the accuracy increased only to 57% on an independent test set. As before, there was no gain overall system recognition accuracy. This result is in contrast to the vast improvements seen in (McDermott, 1989). While one possible difference is that they used handmarked phoneme boundaries, in isolated word utterances for both training and test, we believe that the important difference was probably that the final recognition task in their case was simply to recognize the identity of the phoneme. This was quite similar to the optimization in the LVQ2.

The conclusion from these several efforts at improving the vector quantization or distance measure by looking at the phoneme labels of single frames (or even clusters of frames) was that any gains that were achieved were not relevant to the performance of the entire system.

Any method that would improve the vector quantization must be done within the context of the whole recognition system.

2.3 Shared Mixture VQ

One technique that partially avoids problems attributed to VQ is to use a fuzzy VQ technique (Tseng, ICASSP87) or a more rigorous shared mixture technique (Bellagarda, 1989). The basic notion is that each of the VQ regions is now treated as a guassian distribution that is shared by all of the probability densities in the entire HMM system. One of the effects of this is that an input feature vector is no longer "in" one cluster or another. Instead, there is a probability that it belongs to several clusters. The probability of an input feature vector for a state is now a weighted combination of the discrete probabilities of the nearby clusters. This might have some smoothing effect on the discrete probability densities. It also might avoid some of the quantization effects, since the probability for an input feature vector would vary continuously between two or more clusters.

We implemented a subset of the pieces of the shared mixture algorithms. In particular, we decided to avoid the computationally expensive reestimation of the mixture means and variances. Instead, we estimated a mean and full covariance matrix from the training data that fell within each of the original clusters. Then, we could compute for each training or test frame, the probability that it belonged to each of the 256 clusters. We found that the nearest five clusters accounted for 99% of the probability, and therefore discarded all but the nearest five. The five pairs of numbers (index and probability) then could replace the single VQ index in the probability lookup of either the training or recognition algorithms.

We performed experiments with the shared mixtures in the decoder alone, or in the training and decoder. We found a 10%-20% gain for just using it in the decoder. There was no gain for using it in the training. While the effect of shared mixtures might be similar to those of other density smoothing algorithms, we found an additional 5%-20% reduction in error rates for mixtures. This condition is included in the recognition results given at the end of this paper.

3 Optimizing System Parameters

Here we describe two techniques for estimating global system parameters in the BYBLOS system.

3.1 Deleted Estimation Of Context Weights

The BYBLOS system interpolates all the different probability densities of the context-dependent phonemes to obtain a robust estimate of the densities. Currently we use heuristic weights that are a function of:

- type of context (phone, left, right, triphone)
- number of occurrences in training (5 ranges)
- state in phone model (left, middle, right)

The values of these weights were set based on reasonable intuitions about the importance of phonetic contexts and amount of training on different parts of a phoneme. We ran a few tuning experiments (on an earlier database) to determine rough scaling factors on the initial weights. Therefore, it is likely that we would see no further improvement by estimating the weights automatically with deleted estimation. However, we might expect that if we estimated the weights automatically, we could use different weights for each speaker. We wanted to avoid any approximations if possible, due to assumptions about the alignments remaining fixed, and so we chose to iteratively estimate the weights and then reestimate the probability densities.

We were worried about the effectiveness of the jackknifing procedure that is normally used, since the weights for combining models are estimated for the case where only half of the data was used to estimate the models. Therefore, we developed a method for holding out only one utterance at a time, that was still very efficient:

Each normal pass of forward-backward is followed by a second pass that estimates the weights. At the end of the forward-backward pass, we retain the "counts". In the second pass we remove the "counts" from one sentence at a time and then estimate context weights using that deleted sentence.

1. Run usual forward-backward iteration on all sentences
2. For each sentence:
   (a) Run forward-backward on this sentence using "old" model to determine its contribution to the new model.
   (b) Subtract the contribution of this sentence from those models relevant to this sentence.

(c) Run forward-backward to compute weight counts from this sentence using the model with the contribution for this sentence removed.

3. Reestimate the context weights from the weight counts.

4. iterate

This algorithm requires only two times the computation of the normal forward-backward algorithm, and should result in a more accurate estimate of the weights than the usual procedure. Unfortunately, when we ran our initial experiments, we found no improvement, despite the fact that the likelihood of the training data had increased somewhat. It is possible that the initial heuristic weights are close enough, or that the "reasonable" continuity constraints existing in the initial weights were lost when each weight was estimated independently.

3.2 MMI Estimation Using "N-Best" Alternatives

We have found in the past that the recognition results can be improved by optimizing the weights for the different sets of features. We felt that it would make sense, therefore, to estimate these weights automatically. However, since these weights are actually exponents on the probability densities, it is not possible to estimate them using maximum likelihood (ML) techniques. Clearly, the largest likelihood would occur when all the weights were large. If we constrain the weights to sum to one, there is still a problem, since the ML solution would determine one weight that would be equal to one, and the others would be zero. This can be shown easily for the Viterbi case by realizing that the final likelihood is simply the product of the whole sentence likelihoods due to each codebook. Therefore, we needed to use a discriminative technique to estimate the feature weights.

We chose to use Maximum Mutual Information (MMI) Estimation to estimate these (and possibly other) parameters. In MMI, we want to maximize the likelihood of the correct answer (given the input) relative to the likelihood of all the possible answers. This typically is done by determining a set of alternative answers and performing a gradient descent to improve the mutual information. The problem of finding good alternatives to the correct answer is harder for continuous speech than for isolated words, where each alternative can be considered explicitly. However, the N-Best algorithm, (Chow, 1989) which is described elsewhere in these proceedings can be used to solve this problem.

The N-Best algorithm is a time-synchronous Viterbi-style beam search algorithm that can be made to find the most likely N whole sentence alternatives that are within a given a "beam" of the most likely utterance. The algorithm can be shown to be *exact* under some reasonable constraints. The computation is linear with the length of the utterance, and faster than linear in N.

We use the N-Best algorithm to generate a list of the most likely alternatives for each sentence in a held-out set. We then explicitly compute the likelihood of the correct sentence (if it is not already in the list). The mutual information for each sentence and its corresponding imposters is used to compute a set of weights for each sentence hypothesis. The weights for the correct sentences are positive, while the imposter sentences have negative weights. Then, we use all of the sentences (real and imposter) in the usual forward-backward algorithm with the counts multiplied by the weight for the sentence. States common to all sentence hypotheses for a sentence will get no counts. Then, we compute the gradient directly from the counts and adjust the parameters accordingly.

We used the above algorithm to estimate the (three) feature set weights for each speaker separately. We used the 600 training sentences to generate the models, which we assumed would not change. Then we generated 10 imposter sentences for each of the 100 development test sentences. We used five iterations to optimize the codebook weights. Then we evaluated the resulting models on the February 1989 test data. The result was a 10% reduction in error rate, relative to the initial weights, which were empirically optimized for all the speakers. The gain is somewhat small, but we are not sure how much gain to expect from optimizing only three parameters. Furthermore, we noticed that the gradient descent was dominated by a few bad sentences that it probably could not fix anyway. We believe that this area needs more work.

4 Cross-Word Triphone Models

A model of phonetic coarticulation between words has been proven to be effective by researchers at CMU, SRI, and Lincoln Labs (Paul, 1989; Lee, 1989; Murveit, 1989). However, we wanted to avoid changes to existing training and decoding programs. Therefore, we developed a compiler that reads a phonetic dictionary and a

word grammar and writes out a dictionary of triphones and a triphone grammar. That is, the new dictionary has one "word" for each triphone (about 7,000 in this case), and the new grammar specifies allowable sequences of these triphones. There are approximately 60,000 triphone arcs in the resulting grammar (for the word-pair grammar). Given this new dictionary and grammar, the training program did not need to change at all and the recognition program only needed to know how to write out the real words instead of the triphone names – a small change. As a result, we were able to implement the cross-word triphone effect in only 5 weeks.

We tested the new models on the May 1988 test data. The addition of cross-word triphone models reduced the word error rate by 30% as will be seen in the tables of results below.

5 System Recognition Results

The table below compares the word error rate with several combinations of smoothing, mixtures, and cross-word triphones. The "smoothing" algorithm is the Triphone Coocurrence Smoothing algorithm that was presented at the DARPA meeting in June '88. "Mixtures" means using the Shared Mixtures VQ as described above. "X-Word" means using Cross-Word Triphone models (without smoothing or mixtures). And, the last line includes all three algorithms.

Results are shown for the different speaker-dependent test sets, indicated by the dates of the test set. The results for the baseline system and for the system with smoothing have been reported previously for the May'88 and Feb'89 test sets, and are given for reference. We have been using the May'88 test set as our development test set. Therefore, each of the conditions is shown for this test set. As can be seen, the error rate with the Word-Pair grammar has been reduced from 3.4% to 1.7%. We never tested this configuration with no grammar until the Oct'89 test.

The results for the Oct'89 test set using all three algorithm extensions indicate that the word error rate with the Word-Pair grammar is 2.5%, and the error rate with no grammar is 10.6%. While these error rates represent the best performance reported so far on this database, we were surprised at the large increase in error rate from the May'88 test to the Oct'89 test. Therefore, we reran the system configuration used in February, 1989, which included only the smoothing algorithm. As can be seen, the word error rate was 3.8% on the Oct. '89 test, as compared with 2.7% on the May '88 test, which is consistent with the other results. It is clear that the October 1989 test is significantly harder (at least for our system) than the May 1988 test set, perhaps because it comes from a different recording session. However, the relative improvements in the algorithms were observed in the new test set as well as the old.

Percent word error using Word-pair grammar

	Test Set		
System	May '88	Feb. '89	Oct. '89
Baseline System	3.4	2.9	
Smooth	2.7	3.1	3.8
Smooth + Mix	2.5		
X-Word	2.3		
Smooth + Mix + X-Word	1.7		2.5

Percent word error using no grammar

	Test Set		
System	May '88	Feb. '89	Oct. '89
Baseline System	16.2	15.3	
Smooth	15.8	13.8	
Smooth + Mix	12.6		
X-Word			
Smooth + Mix + X-Word			10.6

6 Conclusions

We draw several conclusions from this work:

- Supervising the VQ with phoneme identity does not help overall recognition performance.
- Shared mixtures in the decoder reduces error rate by 10%-20% depending on the grammar, but after smoothing only by 5%-20%.
- We found no improvement for replacing the heuristically derived weights for the context-dependent models with weights determined by deleted estimation.
- We have implemented an algorithm for MMI training in continuous speech that uses alternatives generated by the N-Best algorithm. Initial experiments to optimize the three feature set weights using this procedure reduced word error rate by 10%.

- As expected, using cross-word triphone models reduced word error rate by 30%.

- The word error rate using the Word-Pair grammar is now close to 2%, depending on the test set. When no grammar is used the error rate was 10.6% on the Oct. '89 test set. Due to the very low error rate with the Word-Pair grammar, we will use the statistical class grammar (Derr, 1989) for most of our testing as it will be easier to measure improvements using this more difficult and more realistic grammar.

Acknowledgement

This work was supported by the Defense Advanced Research Projects Agency and monitored by the Office of Naval Research under Contract Nos. N00014-85-C-0279 and N00014-89-C-0008.

References

[1] Bellagard, J. and D. Nahamoo (1989) "Tied mixture continuous parameter models for large vocabulary isolated speech recognition" *IEEE ICASSP89*

[2] Brown, P. (1987) "The Acoustic-Modeling Problem in Automatic Speech Recognition" *PhD Thesis, CMU, 1987*

[3] Chow, Y.C. and R.M. Schwartz (1989) "The N-Best Algorithm: An Efficient Procedure for Finding Top N Sentence Hypotheses" *Elsewhere in these Proceedings of the Oct. 1989 DARPA Speech and Natural Language Workshop* Morgan Kaufmann Publishers, Inc., Oct. 1989.

[4] Derr, A. and R.M. Schwartz (1989) "A Statistical Class Grammar for Measuring Speech Recognition Performance" *Elsewhere in these Proceedings of the Oct. 1989 DARPA Speech and Natural Language Workshop* Morgan Kaufmann Publishers, Inc., Oct. 1989.

[5] Huang, X.D. and M.A. Jack (1989) "Semi-continuous hidden Markov models for speech recognition" *Computer Speech and Language.* Vol 3, 1989

[6] Kohonen, T., G. Barna, and R. Chrisley (1988) "Statistical Pattern Recognition with Neural Networks: Benchmarking Studies," *IEEE, Proc. of ICNN, Vol. I, pp. 61-68, July, 1988*

[7] Lee, K.F., H.W. Hon, , and M.Y. Hwang (1989) "Recent Progress in the Sphinx Speech Recognition System" *Proceedings of the Feb. 1989 DARPA Speech and Natural Language Workshop* Morgan Kaufmann Publishers, Inc., Feb. 1989.

[8] McDermott, E. and S. Katagiri (1989) "Shift-Invariant, Multi-Category Phoneme Recognition using Kohonen's LVQ2," *IEEE ICASSP-89, pp. 81-84*

[9] Paul, D. (1989) "The Lincoln continuous speech recognition system recent developments and results" *Proceedings of the Feb. 1989 DARPA Speech and Natural Language Workshop* Morgan Kaufmann Publishers, Inc., Feb. 1989.