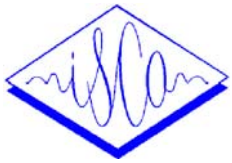

EUROSPEECH '97
5th European Conference on Speech Communication and Technology
Rhodes, Greece, September 22-25, 1997

# MATCHING TRAINING AND TESTING CRITERIA IN HYBRID SPEECH RECOGNITION SYSTEMS

*Xin Tu* *Yonghong Yan* *Ron Cole*

E-mail: (xintu/yan/cole)@cse.ogi.edu
Center for Spoken Language Understanding, Oregon Graduate Institute
P.O. Box 91000, Portland, OR 97291-1000, USA

## Abstract

Inconsistency between training and testing criteria is a drawback of the hybrid artificial neural network and hidden Markov model (ANN/HMM) approach to speech recognition. This paper presents an effective method to address this problem by modifying the feedforward neural network training paradigm. Word errors are explicitly incorporated in the training procedure to achieve improved word recognition accuracy. Experiments on a continuous digit database show a reduction in word error rate of more than 17% using the proposed method.

## 1. Introduction

Hybrid ANN/HMM speech recognition systems have been shown to be competitive with traditional HMM systems because they integrate the advantages of hidden Markov models and artificial neural networks. HMMs have been proven effective at modeling the temporal structure of the speech signal, while ANNs are suitable for static pattern classification due to their non-parametric nature and discriminative training scheme.

In our hybrid ANN/HMM speech recognition system, an ANN classifier is often used to estimate the posterior probabilities of phoneme classes given input patterns. A Markov chain is used to model the state transition probabilities. Phoneme classes correspond to states in the Markov chain, and the outputs of the classifier are used to approximate the state emission probabilities.

Different criteria are used during training and testing. When training a neural network classifier, traditionally, the optimization criterion for the classifier is the classification accuracy of the phoneme classes. However, the most often used criterion for testing a recognition system is the word or sentence recognition accuracy. The inconsistency between the two criteria raises the problem that optimizing phoneme classification during training does not necessarily lead to an optimal word recognition accuracy. The inconsistency between training and testing criteria often results in suboptimal solutions.

This problem has been studied recently. Bengio[1] proposed a global optimization method in an attempt to link the performance of frame level classification with the HMM phone models. Tebelskis[2] approached this problem by creating an extra word layer in a Multi-State Time Delay Neural Network classifier. In both cases, improved performance was obtained over the baseline systems.

This paper presents a new method to incorporate word errors into the training of a neural network classifier. The optimization criterion for the classifier is tuned toward minimizing the word recognition error rate, which leads to a consistent optimization criterion for training and testing. This method is used iteratively to construct a set of classifiers. A boosting technique is applied to combine these classifiers. Experiments on a continuous digit database show 17.8% word error reduction.

## 2. Resolving Inconsistency: Word Error Back propagation

### 2.1. System Description

The classifier in our hybrid ANN/HMM speech recognition system is a three-layer fully connected feedforward neural network (MLP). The output units are states which are subphonemes. Subphonemes are obtained by splitting a phoneme equally into a number of segments (the exact number depends on the phoneme) based on the phonetic transcriptions. Hence, every frame is associated with a "desired" output class which is a subphoneme. The MLP is used to estimate the posterior probabilities of the subphoneme units. During supervised training, for each input pattern, the target for the "desired" class is set to 1.0, and for the rest of the classes, the targets are all set to 0.0.

The *error signal* at output $j$ for input pattern $n$ is

defined by

$$e_{j(n)} = O_{j(n)} - T_{j(n)} \tag{1}$$

where $O_j$ is the output of node $j$, and $T_j$ is the target for node $j$. The *instantaneous sum of squared error* of the network for pattern $n$ is written as

$$\mathcal{E}_{(n)} = \frac{1}{2} \sum_{j \in C} e^2_{j(n)} \tag{2}$$

where the set $C$ includes all the output nodes of the network. Let $N$ denote the total number of input patterns in the training set, the *average squared error* is obtained by summing $\mathcal{E}_{(n)}$ over all $n$ and normalizing with respect to the size $N$, as shown by

$$\mathcal{E}_{av} = \frac{1}{N} \sum_{n=1}^{N} \mathcal{E}_{(n)} \tag{3}$$

The *average squared error* represents the *cost function* as the measure of the learning performance. The learning process adjusts the weights in the MLP so as to minimize the cost function.

In order to train the classifier to minimize the word error rate, word errors must be combined in the cost function during the optimization process. Bahl[3] and Lee[4] proposed a corrective training method to optimize the parameters of HMM models to improve the the recognition accuracy on the training data. The parameters were modified in such a way that the misrecognized word became less probable and the correct word became more probable. We extend this concept to optimize the neural network classifier in a hybrid speech system. Our goal is to improve the recognition accuracy on both the training data and testing data by standardizing the training and testing criteria.

There are two major difficulties in incorporating the word error into the cost function. One is how to calculate the word error. The other is how to map the word error to the frame level phonetic outputs in our neural network.

In an isolated word recognition task, word errors are all substitutions. If a word *w1* is misrecognized as *w2*, training can be tuned toward increasing the probability of *w1* and decreasing the probability of *w2*. In a continuous recognition task, the boundaries of the misrecognized word sequence will not always be aligned with those of the correct word sequence due to insertions and deletions. It is difficult to measure how "wrong" a word error is. However, we can use a dynamic programming algorithm to find the underlying state sequences. For a sentence which is misrecognized in the training data set, a Viterbi search can be used to find the misrecognized state sequence, and forced alignment can be used to generate the correct state sequence. By comparing the two state sequences, we can find out where word errors occur.

Figure 1 shows an example in which an incorrect insertion of "oU" takes place. For convenience, we plot the output classes as monophones in this figure. In the real application, all classes are context-dependent phonemes rather than monophones.

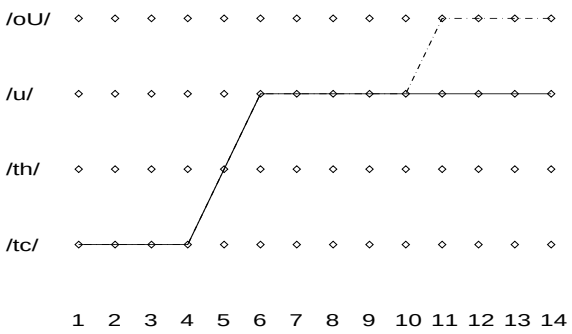


Figure 1: Forced Alignment. The correct utterance is "two"; the recognized utterance is "two oh".

In the example in Figure 1, the recognized state sequence (the dotted line) is different from the forced alignment (the solid line) for the last four frames where the insertion occurs. The "distance" between these two paths from frame 11 to 14 reflects the word error in this utterance. In order to minimize the "distance", the error signal needs to be adjusted such that the probability of the phoneme /oU/ will be reduced and the probability of the phoneme /u/ will be increased. We rewrite equation (1) as

$$e_j = \begin{cases} O_{j(n)} - (1.0 - \epsilon_{(n)}) & j = /oU/ \\ O_{j(n)} - (0.0 + \epsilon_{(n)}) & j = /u/ \\ O_{j(n)} - 0.0 & otherwise \end{cases} \tag{4}$$

Thus, by introducing a positive variable $\epsilon_{(n)}$, the target for /oU/ is decreased, and the target for /u/ is increased. The consequence of these changes is that the learning of /oU/ is discouraged, while the learning of /u/ is enhanced.

The variable $\epsilon_{(n)}$ is a cost related to the word error. We define $\epsilon_{(n)}$ as:

$$\epsilon_{(n)} = \max\{0, O_{/oU/(n)} - O_{/u/(n)}\} \tag{5}$$

which is the difference between the probability of the correct state and the incorrect state for a given input pattern.

After adding $\epsilon_{(n)}$ in the calculation of the error signal, the discrimination between the correct and misrecognized classes is enhanced whenever a word er-

ror occurs. For utterances which are correctly recognized, the calculation of the error signal remains the same. Thus, the training criterion is consistent with the testing criterion, since the classifier is optimized for the purpose of minimizing the word error rate.

However, training on errors is susceptible to overfitting the training data (poor generalization on testing data). This side effect suggests that a boosting[5] technique should be applied.

### 2.2. Boosting

Boosting is a general method to improve the performance of a learning algorithm. Freund and Schapire[6] proposed a boosting algorithm based on adaptively resampling and combining. The basic idea is to resample the training data based on the performance of previous constructed classifiers. The method assumes that all patterns in the training set have equal probabilities to be picked during the training of the first network, which is the case for most classification applications. Patterns in the training data which fail to be correctly recognized by the first classifier will have a higher probability of being sampled to train a new classifier than those that do not. The resampling process can be iteratively used, with the construction of the $(k+1)$st classifier depending on the performance of the $k$ previously constructed classifiers. Finally, the classifiers are combined as a single classifier by weighted voting. This method was claimed to be capable of improving the performance on the training set as well as the testing set.

Instead of adaptively resampling the training data, we used the same training patterns but adaptively adjusted the error signal during the training of a new classifier. The adjustment criterion was the correctness of the underlying words in an utterance.

Originally, boosting used weighted voting to combine multiple classifiers into a single classifier. This works for some static pattern classification problems. Since our classifier outputs state probabilities, averaging is more suitable to be used to combine the classifiers[7]. In order to weight each classifier equally, the outputs from each classifier are normalized before averaging their scores.

## 3. Experiment and Results

### 3.1. Baseline system

We set up our experiment using the OGI digit database, which is a subset of the OGI Numbers corpus[8] collected by the Center for Spoken Language Understanding at OGI. The OGI digit database is a speaker-independent, continuous digit database. The vocabulary contains 11 digits: "zero, one, two, three, four, five, six, seven, eight, nine, oh." Each utterance contains 1 to 10 continuously spoken digits from a telephone call. Calls are made from all over the United States. The database is randomly divided into three sets, with 2090 utterances for training, 512 for cross-validation, and 1899 for testing.

Due to the high computational complexity of neural network training, we did not use all available frames in the training set. For subphoneme classes with a relatively smaller number of training frames, all available frames might well be used, while for more common classes, only a subset of available frames was actually used for training. The frames were chosen in such a way as to spread them out uniformly over the whole training set.

Our neural network classifier is a three-layer fully connected feedforward MLP with 130 input nodes, 150 hidden nodes and 209 output nodes used for state probability estimation. The 130 input data consist of five contiguous frames of features. For each frame, the features are 12th order MFCC coefficients, their delta features, energy and delta energy. The analysis window is 25 ms long, updated every 10 ms. The MLP in our baseline system was trained with a stochastic back-propagation algorithm, using a mean-squared error (MSE) cost function.

### 3.2. Boosting two classifiers

The baseline system was our first classifier (classifier1). It was used to generate state probabilities for the training data. The probabilities were then used in a Viterbi search to decode the underlying words for each utterance. For an utterance which had word errors, we ran forced alignment to get the correct state sequence according to the phonetic pronunciations in our digit dictionary. By comparing the Viterbi search result and force aligned state sequence we located the frames which were misaligned. The error signal was then calculated according to equation (4) during the training of a second classifier (classifier2). By averaging the outputs of the two classifiers, a new classifier, called boost1, was constructed. Results are shown in Table 1.

### 3.3. Boosting three classifiers

Once we had boost1, we used it to construct a third classifier (classifier3) using the same method as described above. By averaging the outputs of the three classifiers, we got a new classifier which is called boost2. Results using boost2 are shown in Table 2.

Both boost1 and boost2 show improved performance

| Data | Unit | Baseline | boost1 |
|---|---|---|---|
| Training Set % | word | 2.15 | 1.32 |
| | sentence | 7.46 | 4.69 |
| Test Set % | word | 5.21 | 4.44 |
| | sentence | 18.01 | 16.38 |

Table 1: Word and sentence error rates on the OGI numbers database using boost1.

| Data | Unit | Baseline | boost2 |
|---|---|---|---|
| Training Set % | word | 2.15 | 1.28 |
| | sentence | 7.46 | 4.45 |
| Test Set % | word | 5.21 | 4.28 |
| | sentence | 18.01 | 15.96 |

Table 2: Word and sentence error rates on the OGI numbers database using boost2.

on training and test data sets. Boost2 achieved 17.8% word error reduction, and 11.4% sentence error reduction on the final test data set. However, the results from boost2 are marginally better than those from boost1 which suggests no more classifiers need to be constructed.

## 4. Conclusion

Hybrid speech recognition systems lack consistency between training and testing criteria. This often leads to a suboptimal solution. This paper has presented a method to standardize training and testing criteria for hybrid ANN/HMM speech recognition systems while preserving a feedforward neural network architecture. The best result using this method obtains a reduction of 17.8% in word error rate on the digit recognition task.

## 5. Acknowledgement

This work was supported by the National Science Foundation, the Office of Naval Research, DARPA, and the member companies of CSLU.

## References

[1] Y. Bengio, R. Mori, G. Flammia, and R. Kompe. Global optimization of a neural network hidden markov model hybrid. *IEEE Trans. on Neural Networks*, (3):252–259, 1992.

[2] J. Tebelskis. Performance through consistency: Connectionist large vocabulary continuous speech recognition. *Proc. ICASSP*, pages 3307–3310, 1995.

[3] L.R. Bahl, P.F. Brown, P.V.De Souza, and R.L. Mercer. A new algorithm for the estimation of hidden markov model parameters. *Proc. ICASSP*, pages 493–496, 1988.

[4] K.F. Lee and S. Mahajan. Corrective and reinforcement learning for speaker-independent continuous speech recognition. *Computer Speech and Language*, 4(3):231–245, 1990.

[5] L. Breiman. Bias, variance, and arcing classifiers. *Technical Report 460, Department of Statistics, University of California, Berkeley,California 94720*, April, 1996.

[6] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and application to boosting. *Proceeding of the second European Conference on Computational Learning*, 1995.

[7] G. Cook and T. Robinson. Boosting the performance of connectionist large vocabulary speech recognition. *Proceedings of the International Conference on Spoken Language Processing*, pages 1305–1308, 1996.

[8] R. Cole, M. Noel, T. Lander, and T. Durham. New telephone speech corpora at cslu. *Proceedings of the European Conference on Speech Technology*, pages 821–824, 1995.