Investigations of Issues for Using Multiple Acoustic Models to Improve Continuous Speech Recognition

Rong Zhang and Alexander I. Rudnicky

Language Technologies Institute, School of Computer Science
Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213
{rongz, air}@cs.cmu.edu

10.21437/Interspeech.2006-201

Abstract

This paper investigates two important issues in constructing and combining ensembles of acoustic models for reducing recognition errors. First, we investigate the applicability of the AnyBoost algorithm for acoustic model training. AnyBoost is a generalized Boosting method that allows the use of an arbitrary loss function as the training criterion to construct ensemble of classifiers. We choose the MCE discriminative objective function for our experiments. Initial test results on a real-world meeting recognition corpus show that AnyBoost is a competitive alternate to the standard AdaBoost algorithm. Second, we investigate ROVER-based combination, focusing on the technique for selecting correct hypothesized words from aligned WTN. We propose a neural network based insertion detection and word scoring scheme for this. Our approach consistently outperforms the current voting technique used by ROVER in the experiments.

Index Terms: Boosting, ROVER.

1. Introduction

The past few years have witnessed the success of Boosting algorithm in many research fields including continuous speech recognition. In Boosting training, a set of recognition models are iteratively generated such that the examples misclassified by the current model will be given higher weights in the training of subsequent models. In a generalization stage, the hypotheses predicted by individual models are composed together to form the final hypothesis using combination techniques such as majority voting. Most Boosting approaches such as AdaBoost [1] and LogitBoost [2] can be viewed as special cases of AnyBoost, an abstract algorithm developed by [3] as an endeavor to unify Boosting training into a generalized framework via gradient descent in function space. The advantage of AnyBoost is that it provides a platform that enables us to investigate appropriate loss functions.

MCE (Minimum Classification Error) is a discriminative training method extensively used in continuous speech recognition [4]. The goal of MCE is to increase the separability between desired and competing classes. In MCE training, model parameters are optimized by minimizing the value of a sigmoid-based differentiable loss function that quantitatively measures the classification error on training set. The successes of both Boosting and MCE suggest that a combination of these two techniques may have the capability to improve the performance of acoustic modeling. This paper presents an AnyBoost-based training scheme that uses the MCE discriminative criterion for constructing ensembles. Our solution is different from the work of other researchers [5] in which MCE is performed as a separate post-processing module to update the classifiers obtained from Boosting training.

Hypothesis generation is another important issue for ensemble based continuous speech recognition. Standard Boosting uses sentence level majority voting to select the most likely hypothesis as the final output. This method ignores some important information associated with individual words in the hypothesis, such as confidence and segmentation. Research has shown that ROVER (Recognizer Output Voting Error Reduction) [6], a word level combination method that integrates word information, can significantly improve recognition accuracy.

However, the present version of ROVER has its own weakness. For example, the voting module adopted by ROVER to search the best word sequence from WTN (Word Transition Network) is essentially the linear combination of two types of information: frequency of occurrence and confidence score. In many cases, the correct hypothesized words cannot be found due to the simplicity of this strategy. This paper proposes a neural network based two-level scoring scheme to address this problem. Once the WTN is constructed, we first use a binary classifier to determine if the questioned WTN node is an insertion error. If not, each word in the node will be scored on the basis of a variety of features extracted from multiple information sources, and the one with the highest score is chosen as the decoding result.

2. AnyBoost with MCE criterion

In this section we investigate the AnyBoost algorithm embedded with a MCE loss function for acoustic model training.

2.1. Discriminative Loss Function of MCE

Suppose we have a training set of N labeled examples $\Psi = \{(\mathbf{x}_i, y_i) \mid 1 \le i \le N\}$, where feature vector $\mathbf{x}_i \in \mathbf{R}^D$ and class label $y_i \in Y = \{c_1, c_2, ..., c_M\}$. We wish to learn a classifier $f(\mathbf{x}, y)$, which defines a mapping $\mathbf{R}^D \times Y \to [0,1]$, giving each feature/class pair $(\mathbf{x}, y)$ a probabilistic or confidence measure. The loss function that MCE training aims to minimize is as follows [4] (other variants exist).

$$L_{MCE}(f) = \frac{1}{N}\sum_{i=1}^{N} L_i(f) \tag{1}$$

where $L_i(f)$ denotes the classification loss of $f(\mathbf{x}, y)$ made on instance $(\mathbf{x}_i, y_i)$ such that

$$L_i(f) = \frac{1}{1+\exp\{\rho * [f(\mathbf{x}_i, y_i) - \frac{1}{M-1}(\sum_{y \in Y \text{ and } y \ne y_i} f^{\eta}(\mathbf{x}_i, y))^{1/\eta}]\}} \tag{2}$$

In (2), parameter η controls how the competing class $y \ne y_i$ are weighted, and ρ controls how sharply the sigmoid function

changes at the transition point. If we create a pseudo class $\bar{y}_i$ to cover all the competing classes that $y \neq y_i$, and let

$$f(\mathbf{x}_i, \bar{y}_i) = \frac{1}{M-1}[\sum_{y \in Y \text{ and } y \neq y_i} f^{\eta}(\mathbf{x}_i, y)]^{1/\eta} \quad (3)$$

then (1) can be rewritten as

$$L_{MCE}(f) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{1+\exp\{\rho * [f(\mathbf{x}_i, y_i) - f(\mathbf{x}_i, \bar{y}_i)]\}} \quad (4)$$

Noting the fact that $\mathbf{x}_i$ is misclassified when $f(\mathbf{x}_i, y_i) < f(\mathbf{x}_i, \bar{y}_i)$, the loss function $L_{MCE}(f)$ is essentially a sigmoid-based differentiable 0-1 function that mimics the classification error rate on a training set. Minimization of the value of $L_{MCE}(f)$ via certain optimization methods such as gradient descent will increase the probability of desired class y_i being predicted, and meantime, decrease the probability of alternative classes $y \neq y_i$ being predicted.

2.2. AnyBoost

Traditionally, MCE is only used to train a single classifier. The AnyBoost algorithm provides a general framework for Boosting approaches and allows the use of the MCE criterion to generate ensembles.

Let $lin(\{f\})$ denote the set of all linear combinations of classifiers in $\{f\}$, $F_t \in lin(\{f\})$ denote the ensemble of classifiers after the t-th component f_t has been learned. The goal of AnyBoost is to iteratively find a new classifier f_{t+1} to add to F_t so that the training loss $L(F_t + \alpha_{t+1} f_{t+1})$ can decrease, where α_{t+1} denotes the weight of f_{t+1} in ensemble. Viewed in terms of parameter space, this is equal to seeking the "direction" f_{t+1} such that $L(F_t + \alpha_{t+1} f_{t+1})$ most rapidly decreases. The desired direction is simply $-\nabla L(F_t)$, the negative functional gradient of loss L at F_t. However, in some situations, we can not choose $f_{t+1} = -\nabla L(F_t)$ since $-\nabla L(F_t)$ may not exist in $lin(\{f\})$. Instead, AnyBoost searches for a f_{t+1} that maximizes $< -\nabla L(F_t), f_{t+1} >$, the inner product of the new classifier with the gradient of the loss function. Once f_{t+1} is learned, an appropriate line-search can be used to determine the value of α_{t+1} for hypothesis combination.

In the case of the MCE loss function in use, the inner product $< -\nabla L(F_t), f_{t+1} >$ can be approximated as follows in terms of desired class y_i and pseudo class $\bar{y}_i$.

$$\begin{aligned} &< -\nabla L(F_t), f_{t+1} > \\ &\approx -\sum_{i=1}^{N}[\frac{\partial L_i(F_t)}{\partial F_t(\mathbf{x}_i, y_i)} f_{t+1}(\mathbf{x}_i, y_i) + \frac{\partial L_i(F_t)}{\partial F_t(\mathbf{x}_i, \bar{y}_i)} f_{t+1}(\mathbf{x}_i, \bar{y}_i)] \\ &= -\sum_{i=1}^{N} w_t(i)[f_{t+1}(\mathbf{x}_i, \bar{y}_i) - f_{t+1}(\mathbf{x}_i, y_i)] \end{aligned} \quad (5)$$

where

$$w_i(t) = \frac{\exp(\rho * [F_t(\mathbf{x}_i, y_i) - F_t(\mathbf{x}_i, \bar{y}_i)])}{\{1+\exp(\rho * [F_t(\mathbf{x}_i, y_i) - F_t(\mathbf{x}_i, \bar{y}_i)])\}^2} \quad (6)$$

Please note that finding an f_{t+1} to maximize $< -\nabla L(F_t), f_{t+1} >$ is equivalent to finding an f_{t+1} to minimize the weighted error

$$\varepsilon = \sum_{i=1}^{N} D_t(i)[f_{t+1}(\mathbf{x}_i, \bar{y}_i) - f_{t+1}(\mathbf{x}_i, y_i)] \quad (7)$$

where $D_t(i)$ is normalized $w_t(i)$ such that

$$D_t(i) = w_i(t) \Big/ \sum_{i=1}^{N} w_i(t) \quad (8)$$

The preceding discussion results in the MCE based AnyBoost algorithm, illustrated in Table 1, in which new classifiers are iteratively generated to minimize the weighted error ε.

Initialization:

- Let $D_0(i) = \frac{1}{N}$ for $1 \le i \le N$.
- Let $F_0 = \phi$.

For t=1 to T:

- Learn f_t with respect to distribution $D_{t-1}(i)$.
- If $\langle -\nabla L(F_{t-1}), f_t \rangle \le 0$, then return F_{t-1}. Else,
- Choose weight α_t for f_t via line search.
- Let $F_t = F_{t-1} + \alpha_t f_t$.
- Update distribution $D_t(i)$ according to (6) and (8).

Table 1 AnyBoost with MCE criterion

In traditional AdaBoost training, the condition to increase the weight of an example is that the example is misclassified, namely $f_{t+1}(\mathbf{x}_i, y_i) < f_{t+1}(\mathbf{x}_i, \bar{y}_i)$. In contrast, (6) shows that in AnyBoost with MCE criterion, the weight of an example is maximized when $f_{t+1}(\mathbf{x}_i, y_i) = f_{t+1}(\mathbf{x}_i, \bar{y}_i)$. This indicates that AnyBoost is willing to sacrifice both positive examples that $f_{t+1}(\mathbf{x}_i, y_i) >> f_{t+1}(\mathbf{x}_i, \bar{y}_i)$ and negative examples that $f_{t+1}(\mathbf{x}_i, y_i) << f_{t+1}(\mathbf{x}_i, \bar{y}_i)$ in the training of subsequent classifiers. The former is easy to understand since that $f_{t+1}(\mathbf{x}_i, y_i) >> f_{t+1}(\mathbf{x}_i, \bar{y}_i)$ means the example $(\mathbf{x}_i, y_i)$ has been well encoded in the current model. The decrease of the weights of negative examples is also necessary in some situations. For example, in the case that the training set contains many transcribing errors, it is better to exclude these erroneous examples from model training rather than give them higher weights as AdaBoost does.

3. A new scoring scheme for ROVER

ROVER is a word-level combination approach developed at NIST that aims to yield reduced word error rate by exploiting differences in the nature of the errors made by multiple speech recognition system [6]. Rover proceeds in two stages. In the first stage, the best word hypotheses produced by different recognizers are progressively aligned together to build a single composite Word Transition Network (WTN) by using dynamic programming. Once the WTN is generated, each node in the network is then evaluated by a voting module to select the best word as the final recognition result.

The present version of ROVER uses a simple voting strategy that linearly combines two types of information in word selection: frequency of occurrence and confidence score. The general scoring formula is as follows.

$$Score(word) = \beta * N(word) + (1-\beta) * C(word) \quad (9)$$

where $N(word)$ denotes the normalized frequency of occurrence, $C(word)$ denotes the average or maximum confidence score, and β is a parameter trained to balance $N(.)$ and $C(.)$.

Our preliminary experiments showed that in many cases, the correct hypothesized words cannot be found due to the simplicity of this strategy. To address this problem, we propose a neural network based two-level insertion detection and word scoring scheme. Once the WTN is constructed, we first use a binary classifier to determine if the WTN node in question is an insertion error. If not, each word in the node will be scored on the basis of a variety of features extracted from multiple information sources (Table 2).

For each node in the WTN:
- Compute features for the node and each word in the node.
- Use neural network based binary classifier to determine: Is the node an insertion error?
- If yes, return *null* for this node; else,
- Use neural network based scorer, giving each real word a new confidence score.
- Return the one with highest score.

Table 2 Two-Level Scoring Scheme

The implementation of the two-level scheme involves two aspects: the identification of useful features and the training of neural networks. In our experience (and in that of others), good features usually play a critical role in creating a successful system. Our features are based on use in previous work.

The first task is to train a neural network based binary classifier to determine if a WTN node is an insertion error. There are five features adopted to fulfill this task.

- *Average frequency of occurrence for real words.*
- *Average frequency of occurrence for filler words and null arcs.*
- *Average word level posterior probability for real words.* Word level posterior probability is an extensively used feature in confidence annotation [7]. This feature measures how likely a particular hypothesized word is a correct recognition result. The value is computed from the word lattice or the N-best list by summing and normalizing the scores of paths passing through the word in question.
- *Average word level posterior probability for filler words and null arcs*. A default value is set to null transition arcs since they do not have word probabilities.
- *Entropy*. This feature is designed to measure the degree of confusion within a WTN node. The feature value is computed from the normalized frequency of occurrence.

To train the neural network, the class label of each WTN node is set to either 1 or 0, representing whether it is an insertion error or not. The value can be manually transcribed or obtained by performing an alignment with references. The standard Back-propagation algorithm is used in our experiments as the training method. The hidden layer of the neural network is set to contain 20 nodes.

The next task is to select the *best* real word from the WTN node not classified as insertion. This is realized by a neural network based *scorer*. For each word to be evaluated, its input to this neural network consists of seven features.

- *Frequency of occurrence.*
- *LM Back-off Mode*. This is a language model related feature. For each real word, the value of *LM back-off mode* is determined according to whether the 1, 2, or 3-gram is used to compute language model score.
- *Contextual LM Back-off Mode*. The average *LM Back-off Mode* over the left and right neighbors of the questioned word.
- *Utterance level posterior probability*. The posterior probability of the sentence hypothesis that the questioned word occurs in.
- *Word level posterior probability.*
- *Frame level posterior probability*. This feature originally measures the probability of a word occurring at a given frame [8]. For a word in the WTN node, the feature value is computed by averaging frame probability, across all the frames that the word spans.
- *Recognizer's word accuracy*. The word accuracy of the recognizer that generates the questioned word. The value is computed on the training set.

The neural network is trained in a discriminative way to minimize the following objective function.

$$L = \sum_{i=1}^{I} \exp(s(u_{i,c}) - s(u_{i,d})) \qquad (10)$$

where $s(.)$ denotes the output value of the neural network scorer, $u_{i,d}$ denotes the input feature vector of the desired word in WTN node i, $u_{i,c}$ denotes the feature vector of the second *best* word competing with $u_{i,d}$, and I is the number of WTN nodes participating the training. In our experiments, the neural network scorer has one hidden layer containing 30 nodes, and we use gradient descent as the learning approach to optimize its parameters.

4. Experiments

Our research was carried out in the context of a continuous speech recognition task in a meeting environment [9]. We selected meetings from the ICSI Bro-series, of these 22 meetings were used as the training set which has about 30K transcribed utterances, and the remaining 1 meeting as the test set which has about 1.2K utterances or 7.5K words. 3K utterances were further separated from the training set to form a hold-out set for neural network training. The sampling rate is 11025Hz, and the frames rate is 105 per second. A 13-dimension MFCC feature vector is computed for each frame and then converted to a 39-dimension acoustic feature vector by adding delta and delta-delta coefficients.

All of our experiments, both training and test, were performed using the Carnegie Mellon Sphinx III system, a fully-continuous HMM recognizer designed for LVCSR. The dictionary adopted in the experiments was drawn from *cmudict* (http://www.speech.cs.cmu.edu/cgi-bin/cmudict). The context independent phone set for acoustic model training contained 49 basic phonemes. In context dependent training stage, these phones were transformed to triphones and then tied together to make 2000 senones. A 3-state left-to-right architecture was adopted to model each speech unit. Each state was modeled with a mixture of 32 Gaussians. The language model was trained using the speech transcripts and text data from other available sources i.e. WSJ.

4.1. Experiment on AnyBoost

Our first experiment investigated the AnyBoost algorithm with MCE criterion in acoustic model training. The class space Y is

constrained to N-Best list, and $f(\mathbf{x}, y)$ is set to the posterior probability $P(y \mid \mathbf{x})$, where $\mathbf{x}$ denotes the sequence of feature vectors for an input utterance, and y denotes a sentence hypothesis. $P(y \mid \mathbf{x})$ can be derived from acoustic model scores and language model scores. The number of acoustic models in the ensemble is set to 7 due to computation concerns. The performance of AnyBoost is compared with standard AdaBoost. Table 3 shows the results, in which T=n is the word error rate obtained from hypotheses combination of n models. We use standard ROVER for the combination. The baseline is 31.42%, which is the word error rate achieved by using single acoustic model (T=1).

%	T=2	T=3	T=4	T=5	T=6	T=7
Ada.	30.48	30.02	29.67	29.49	29.56	29.66
Any.	30.48	30.14	29.71	29.74	29.34	29.46

Table 3 AnyBoost vs. AdaBoost

Table 3 shows that AnyBoost and AdaBoost are in general comparable to each other in the word error rate they achieved. It is worth noting that the performance of AdaBoost starts to degrade when the 6-th acoustic model are added for decoding. This suggests that increase of the weights of *hard-to-learn* examples to some extent may cause problems i.e. overfitting, especially in the case that these examples are corrupted with noise or mis-transcribed. In contrast, AnyBoost with MCE criterion demonstrates some advantages in this situation.

4.2. Experiment on ROVER

The second experiment investigates the new two-level scoring scheme for ROVER combination. Table 4 shows the recognition results of standard ROVER (old) and the new approach (new). The acoustic models are trained from the previous experiment using AnyBoost algorithm with MCE criterion. Experimental results demonstrate the effectiveness of the two-level scheme, which consistently outperforms traditional method in all six different size ensembles, and finally reduces word error rate from the baseline of 31.42% to 29.24%, representing a 6.9% relative reduction. The experimental results are also illustrated in Figure 1.

%	T=2	T=3	T=4	T=5	T=6	T=7
Old	30.48	30.14	29.71	29.74	29.34	29.46
New	29.95	29.80	29.52	29.23	29.20	29.24

Table 4 Standard ROVER (Old) vs. New Scoring Scheme

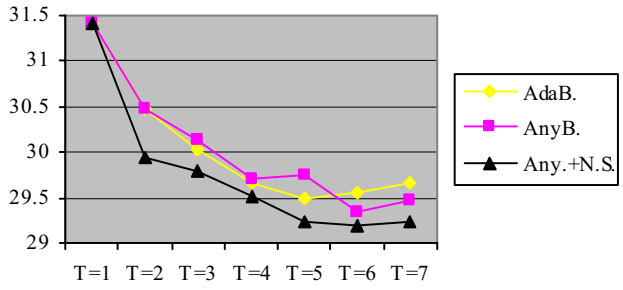

Figure 1 Comparison of Three Approaches

5. Conclusion

This paper investigated two issues in ensemble based continuous speech recognition: how to construct multiple acoustic models and how to combine the hypotheses. We applied the classic MCE discriminative criterion in Boosting training using the framework of AnyBoost algorithm. In addition, we developed a new two-level insertion detection and word scoring scheme for ROVER combination. The new scheme is conceptually simple and straightforward to implement. Encouraging results are observed in experiments with a real-word meeting recognition corpus.

6. Acknowledgement

This work was supported under DARPA grant NBCH-D-03-0010. The content of the information in this publication does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

7. References

[1] R. E. Schapire, "A brief Introduction to Boosting", Proc. of the 16th International Joint Conference on Artificial Intelligence, 1999.

[2] J. Friedman, T. Hastie and R. Tibshirani, "Additive Logistic Regression: A Statistical View of Boosting", Technical Report, Stanford University, 1998.

[3] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting Algorithms as Gradient Descent", In Advances in Neural Information Processing Systems 12, pp 512-518, MIT Press, 2000.

[4] R. Schluter, "Investigations on Discriminative Training Criteria", Ph.D. Thesis, Aachen, Germany, 2000.

[5] I. Zitouni, H. K. J. Kuo, and C. H. Lee, "Combination of Boosting and Discriminative Training Techniques for Natural Language Call Steering Systems", Proc. of ICASSP, 2002.

[6] J. G. Fiscus, "A Post-Processing System to Yield Reduced Word Error Rates: Recognizer Output Voting Error Reduction (ROVER)", Proc. of ASRU 1997.

[7] F. Wessel, K. Macherey and R. Schluter, "Using Word Probabilities as Confidence Measures", Proc. of ICASSP 1998.

[8] R. Zhang and A. I. Rudnicky, "A Frame Level Boosting Training Scheme for Acoustic Modeling", Proc. of ICSLP 2004.

[9] S. Banerjee, J. Cohen, T. Quisel, et al, "Creating Multi-Modal, User-Centric Records of Meetings with the Carnegie Mellon Meeting Recorder Architecture", Proc. of ICASSP 2004 Meeting Recognition Workshop.