IDIAP RESEARCH REPORT

Boosting HMMs with an Application to Speech Recognition

Christos Dimitrakakis [a] Samy Bengio [b]

IDIAP–RR 03-41

September 2003

Dalle Molle Institute for Perceptual Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 – 27 – 721 77 11
fax +41 – 27 – 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

[a] IDIAP, CP952, 1920 Martigny, Switzerland, dimitrak@idiap.ch
[b] IDIAP, CP952, 1920 Martigny, Switzerland, bengio@idiap.ch

IDIAP Research Report 03-41

Boosting HMMs with an Application to Speech Recognition

Christos Dimitrakakis Samy Bengio

September 2003

1 Introduction

Boosting and other ensemble learning methods attempt to combine multiple hypotheses from a number of *experts* into a single hypothesis. This is feasible for classification and regression problems, where the hypothesis is a fixed-length, real-valued vector. Other problems, such as sequence prediction and sequential decision making, can also be cast in the classification and regression framework, thus making the application of ensemble methods to these problems feasible. However in some cases the hypothesis is a sequence of symbols of unspecified length. One such application is speech recognition, where the hypothesis can be a sequence of words or phonemes. The most common machine learning algorithms for sequence processing employ Hidden Markov Models (HMMs) - however, little research has been done in designing new ensemble learning algorithms that are specific to HMMs, or to sequence processing in general.

This paper attempts to fill this gap by examining methods for boosting with HMMs with an application to speech recognition. Firstly, we investigate two distinct methods of applying boosting to HMM training: either perform boosting at phoneme classification level, or at the sentence recognition level. This results in an ensemble of models, where each model has been trained on a different part of the data. In the first case each expert is trained on a subset of frames, while in the second case each expert is trained on a subset of sentences. The ensemble performance is then examined at the sentence recognition level, and for the first case, also at the phoneme classification level. The paper is organised as follows: An introduction to boosting and Hidden Markov Models is given in sections 2 and 3. A review of related research and an outline of methods used in this paper is given in section 4. Experiments on two speech recognition tasks are described in section 5. We conclude with an outline of open research problems in this direction.

2 Boosting

Boosting algorithms [9, 12, 7] are a family of ensemble methods for improving the performance of classifiers by training and combining a number of *experts* through an iterative process that focuses the attention of each new expert to the training examples that were hardest to classify by previous ones. The most successful boosting algorithm for classification is AdaBoost [12], where an ensemble of experts is able to decrease the training error exponentially fast as long as each one has a classification error smaller than 50%.

More precisely, an AdaBoost ensemble is composed of a set of n_e experts, $\mathcal{E} = \{e_1, e_2, ..., e_{n_e}\}$. For each input $x \in X$, each expert e_i produces an output $y_i \in Y$. These outputs are combined according to the reliability $\beta_i \in [0, 1]$ of each expert:

$$y = \sum_{i=1}^{n_e} \beta_i y_i.$$

The expert training is an iterative process, which begins with training a single expert and subsequently trains each new expert in turn, until a termination condition is met. The experts are trained on bootstrap replicates of the training dataset $\mathcal{D} = \{d_i | i \in [1, N]\}$, with $d_i = (x_i, y_i)$. The probability of adding example d_i to the bootstrap replicate $\mathcal{D}_j$ is denoted as $p_j(d_i)$, with $\sum_i p_j(d_i) = 1$. At the end of each boosting iteration j, β_j is calculated according to $\beta_j = \frac{1}{2} \ln \frac{1+\epsilon_j}{1-\epsilon_j}$ where ϵ_h is the average loss of expert e_j, given by $\epsilon_j = \sum_i p_j(d_i) l(d_i)$, where $l(d_i)$ is the *sample loss* of example d_i. If, for any predicate π, we let $[\pi]$ be 1 if π holds and 0 otherwise, it can be defined as: $l(d_i) = [h_i \neq y_i]$. After training in the current iteration is complete, the sampling probabilities are updated so that $p_j(d_i)$ is increased for misclassified examples and decreased for correctly classified examples according to:

$$p_{j+1}(d_i) = \frac{p_j(i) e^{\beta l(d_i)}}{Z_j},$$

where Z_j is a normalisation factor to make D_{j+1} into a distribution. Thus, incorrectly classified examples are more likely to be included in the next bootstrap data set. Because of this, the expert created at each boosting iteration concentrates on harder parts of the input space.

3 Hidden Markov Models

A Hidden Markov Model is defined as a set of n_s states $\mathcal{Q} = \{q_1, q_2, ..., q_{n_s}\}$, transition probabilities from state q_k to state q_j and emission probabilities for each state q_k. The following assumes a basic knowledge of HMMs and introduces the basic notation and the definition of two tasks related to sequence recognition: the classification of a given sequence to one of an *a priori* known number of classes, which amounts to selecting the most likely class from the available selection; and the task of sequence decoding, which amounts to finding the most likely state sequence through a given model, given an observation sequence.

3.1 Sequence Classification

Let us define a sequence of n observations as $S = \{s_1, s_2, ..., s_n\}$. Given a set of n_c classes $\mathcal{C} = \{c_1, c_2, ..., c_{n_c}\}$ we wish to determine which class $c_i \in C$ the sequence belongs to.

We train one HMM m_i for each class c_i. For each model $m_i \in M = \{m_1, m_2, ..., m_{n_c}\}$, we can calculate $p(S|m_i)$ for a given sequence S. Assuming the distribution of c_i is modelled by m_i the class posterior can also be calculated, according to the Bayes rule: $p(m_i|S) = \frac{p(S|m_i)p(m)}{p(S)}$. The most likely model m^* is then selected, with $m^* = \arg\max_j p(m_j|S)$.

3.2 Sequence Decoding

For the simple case where the objective is to determine the class that a particular sequence belongs to, the Bayes Classifier approach described in section 3.1 is adequate. For other applications, such as continuous speech recognition [10], this approach is no longer feasible. In the former case we know that S belongs to one of a finite number of phoneme classes. In the latter case, S corresponds to a sequence of segments of different classes, with the number and position of segments being unknown. Ideally, one would like to exhaustively search through the complete space of possible class sequences that would correspond to S and select the one whose likelihood is the highest. However, this is too expensive in computation time and the Viterbi [14] approximation is used instead.

Assume a sequence S and a set of HMMs $\mathcal{M} = \{m_i | i \in [1, n_c]\}$, where each model $m_i \in \mathcal{M}$ has a set of n_{s_i} states $\mathcal{Q}_i = \{q_{i,j} | j \in [1, n_{s_i}]\}$ and an emission distribution $p(y|q_{i,j})$ at each state. Then for each sequence sample $s_k \in S$ it is possible to calculate the likelihood of s_k given state $q_{i,j} \in \mathcal{Q}$, denoted by $p(s_k|q_{i,j})$. The states that make up each model m_i correspond to speech data within a short time-frame, while the models themselves may correspond to simple phonetic classes, such as phonemes. The departure from the previous application of HMMs to sequence classification is that instead of evaluating $p(m_j|S)$ for each model, the models are incorporated into a larger HMM, with states $\mathcal{Q} = \{\mathcal{Q}_i | i \in [1, n_s]\}$ whose transitions represent the allowed transitions from one phonetic class to the next. These transitions define the vocabulary and language model from which allowed words and sequences of words are obtained. Then, given $p(s_k|q_{i,j}) \forall m_i \in \mathcal{M}, q_{i,j} \in \mathcal{Q}_i, s_k \in S$, we can calculate the likelihood of any given path $V = (v_1, v_2, .., v_n)$, where each $v_k \in V$ corresponds to one of the possible $q_{i,j}$ and the set of all possible paths of length n is $\mathcal{Q}^n$. This is simply

$$p(V) = \prod_{k=1}^{n} p(s_k|v_k) p(v_k|v_{k-1}). \tag{1}$$

The aim of the Viterbi search is to find an optimal path

$$V^* = \arg \max_{V \in \mathcal{Q}^n} p(V).$$

The process is optimal in the sense that the maximally-likely state sequence is found. However, this is not the same as finding the maximally-likely sequence of words.

3.3 Multi Stream Decoding

When the information is coming from multiple streams, or when there are multiple hypothesis models available, it can be advantageous to employ multi-stream decoding techniques [5]. In multi-stream decoding each sub-unit model m_j is comprised of n_e sub-models $m_j = \{m_j^i | i \in [1, n_e]\}$ associated with the sub-unit level at which the recombination of the input streams should be performed. Furthermore the data S is split into a number of vector components called *streams* such as $S = (S^1, S^2, ..., S^{n_e})$.

The problem formulation remains as in the previous section, but this time we are considering likelihoods of combinations of states, rather than of single states. More specifically, we wish to find the path $V = (v_1, v_2, ..., v_n)$ where each $v \in V$ is a vector of states across models: $v_k = (q_k^i | i \in [1, n_e])$ and

$$p(s_k|v_k) = \prod_{i=1}^{n_e} p(s_k^i|q_k^i)^{w_i} \tag{2}$$

or, in an alternative formulation:

$$p(s_k|v_k) = \sum_{i=1}^{n_e} p(s_k^i|q_k^i) w_i \tag{3}$$

where path likelihoods are calculated according as previously (equation 1) and w_i represents the reliability of expert e_i.

4 Boosting HMMs

Boosting had originally been defined for classification tasks, and recently it was generalised to the regression case (See [9] for an overview). However, the amount of research in the application of boosting to sequence learning has been comparatively small: In [13], boosting was used in a speech recognition task. That particular system was HMM-based with ANNs for computing the a posteriori phoneme probabilities at each state. The boosting itself was performed at the ANN level, using AdaBoost with confidence-rated predictions and in which the sample loss function was the frame error rate. The resulting decoder system differed from a normal HMM/ANN hybrid in that each ANN was replaced by a mixture of ANNs. Boosting was also applied in the same context in [2], but in this case the example selection was done at the sentence level, through the use of a sample loss function that considered a classification as correct if its word error rate was below a certain threshold.

The work presented here explores the use of boosting for HMMs that employ Gaussian Mixture Models at the state level. The following sections describe a number of boosting strategies for example and model selection. In the first strategy a set of HMMs is used as Bayes Classifiers at the phoneme level. The second strategy involves example and model selection at the sentence level. However, this complicates the use of boosting, since the task is no longer a classification task. Regression boosting methods are not directly applicable either, since there is no distance measure between the target and the hypothesis at the word level. We attempt to alleviate this problem by introducing methods for casting the word error rate as a bounded scalar quantity. Finally, different strategies for combining the base models are explored.

4.1 Model Training At The Phoneme Level

The simplest way to apply boosting to HMM training is to cast the problem into the classification framework. This is possible at the phoneme classification level, where each class c_i corresponds to one of the possible phonemes. As long as the available data are annotated in time so that subsequences

containing single phoneme data can be extracted, it is natural to adapt each Hidden Markov Model m_i to a single class c_i out of the possible n_c, and combine the models into a Bayes Classifier in the manner described in section 3.1. A Bayes Classifier can then be used as an expert in the AdaBoost framework.

More specifically, each example d_k in the training dataset $\mathcal{D}$ will be a sequence segment corresponding to data from a single phoneme $c_i \in \mathcal{C}$. So each example d_k would be of the form $d_k = (S_k, y_k)$, with $y_k \in \mathcal{C}$ and S_k being a subsequence of features corresponding to single phoneme data. At each iteration j of the AdaBoost algorithm, a new classifier e_j is created, which consists of a set of Hidden Markov Models $\{m_1^j, m_2^j, ..., m_{n_c}^j\}$. Each model m_i^j is adapted to the set of examples $\{d_k \in D_j | y_k = c_i\}$.

It is naturally expected that this type of training will improve performance in phoneme classification tasks and this is tested in section 5.1. Different ways of combining the ensembles resulting from this training in order to perform sequence recognition are described in section 5.2.

4.2 Model Training At The Sentence Level

An alternative way to apply boosting is to consider examples at the sentence level instead of the phoneme level. One way to do this is to attempt to cast the problem as a classification task. In this case, the output space is the space of possible sentences and it is not viable to construct a separate model for each possible sentence. Furthermore, it might be desirable to recognise sentences that do not appear in the training set. Thus, a Bayes Classifier approach is not possible. However, it is possible to use standard speech recognition techniques [10].

Assume $\mathcal{Y}$ is the set of possible sentences. Let us define an example d_k as the pair (S_k, y_k), where S_k is an input sequence and $y_k \in \mathcal{Y}$ describes the sequence to be recognised. Given a model M, composed of several phoneme HMMs such as to allow for all legal sentences in $\mathcal{Y}$, it is possible to perform decoding on S_k as described in section 3.2. The result of this decoding is a hypothesis h_k, with $h_k \in \mathcal{Y}$. It is then possible to view the result of the decoding process as a prediction with sample loss $l(d_k) = [h_k \neq y_k]$, a threshold loss function. Then the AdaBoost algorithm can be applied normally. Although straightforward, such an approach is problematic if none of our models are good enough to produce accurate hypothesis frequently. It is possible that most of the decoded sentences differ from the desired outcome, resulting in a loss higher than 0.5.

One way to get around this problem is to define an appropriate distance metric between two sequences. In that case, boosting techniques defined for regression problems [9, 3, 4, 8, 15, 11] can also be used, though this avenue has not been explored in this paper. A distance metric frequently used in the speech recognition literature is the word error rate:

$$WER = \frac{N_{ins} + N_{sub} + N_{del}}{N_{words}}$$

where, given a sentence y with N_{words} words, N_{ins} is the number of insertions, N_{sub} the number of substitutions and N_{del} the number of deletions of words required to create sentence h from sentence y.

This leaves us with the question of determining the loss function from the word error rate. In fact, an absolute error measure might not be necessary since all that we need to do at each boosting iteration is to assign more weight to the hardest examples and less weight to the easiest ones. In order to define a relative sample loss function from the word error rate, the following steps are performed:

After training an expert e_j on $\mathcal{D}_j$, the WER_k is calculated for each example $d_k \in \mathcal{D}_j$. Then the examples are ranked in N_r ranks[1] according to their WER, yielding a rank $r(d_k) \in [1, N_r]$ for each example. The loss of each example is defined as $l(d_k) = r(d_k)/N_r$. The question of what kind of overall loss is minimised by the above ranking scheme has not been addressed in this work

[1] Some examples might have the same word error rate, so they would be in the same rank.

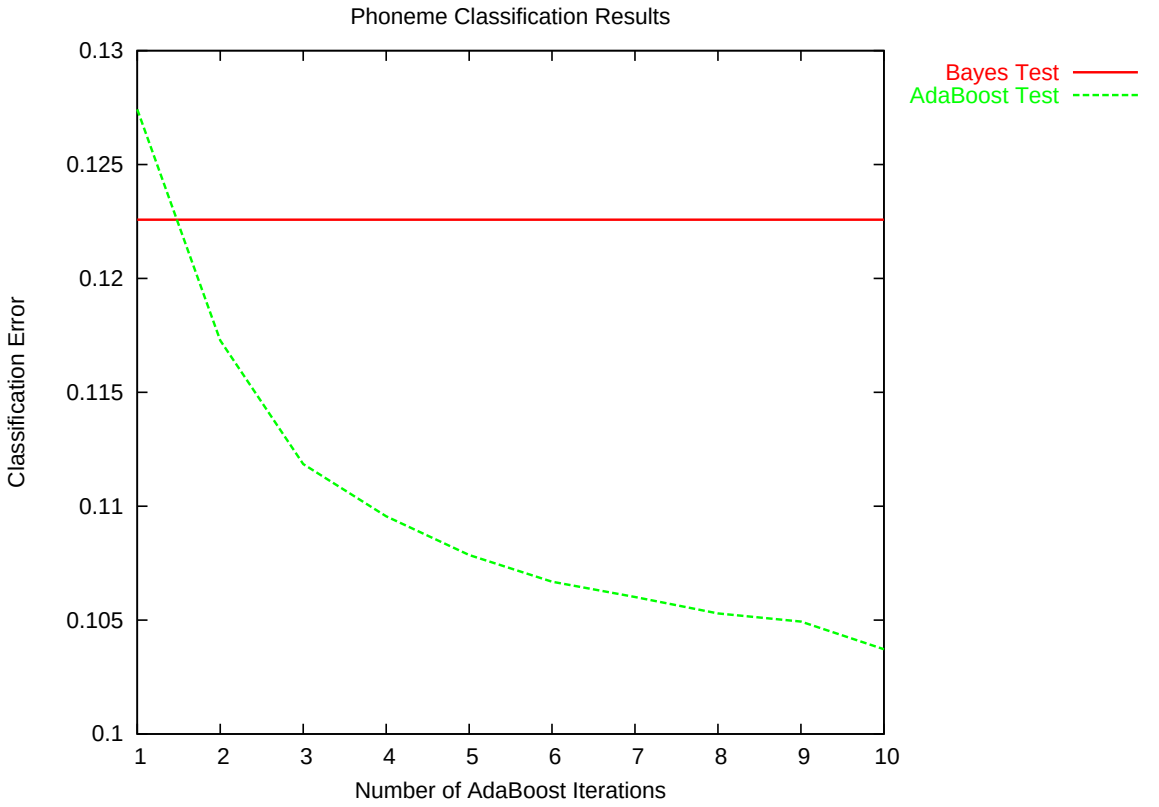

Figure 1: Classification errors in testing for a boosted ensemble of Bayes Classifiers as the number of experts is increased at each iteration of boosting. For reference, the corresponding errors for a Bayes Classifier trained on the complete training set are also included.

5 Experimental Results

5.1 Phoneme Classification

An experiment was performed to test whether it is possible to apply boosting to a sequence classification problem, namely a phoneme classification task. The phoneme data was based on a pre-segmented version of the OGI Numbers 95 (N95) data set [1]. This data set was converted from the original raw audio data into a set of features based on Mel-Frequence Cepstrum Coefficients (MFCC) [10] (with 39 components, consisting of three groups of 13 coefficients, namely the static coefficients and their first and second derivatives) that were extracted from each frame. The data contains 27 distinct phonemes, consisting of 3233 training utterances and 1206 test utterances. The segmentation of the utterances into their consisting phonemes resulted in 35562 training segments and 12613 test segments, totalling 486537 training frames and 180349 test frames respectively. Since the data was pre-segmented, the classification could be performed using a Bayes Classifier composed of 27 Hidden Markov Models, each one corresponding to one class. Given a particular sequence S, model likelihoods are calculated and the Bayes Classifier emits a class label which depends on which Hidden Markov Model had the highest posterior class probability. The HMMs themselves were composed of five hidden states [2] in a left-to-right topology and the distributions corresponding to each state were modelled with a Gaussian

[2] including two non-emitting states: the initial and final states

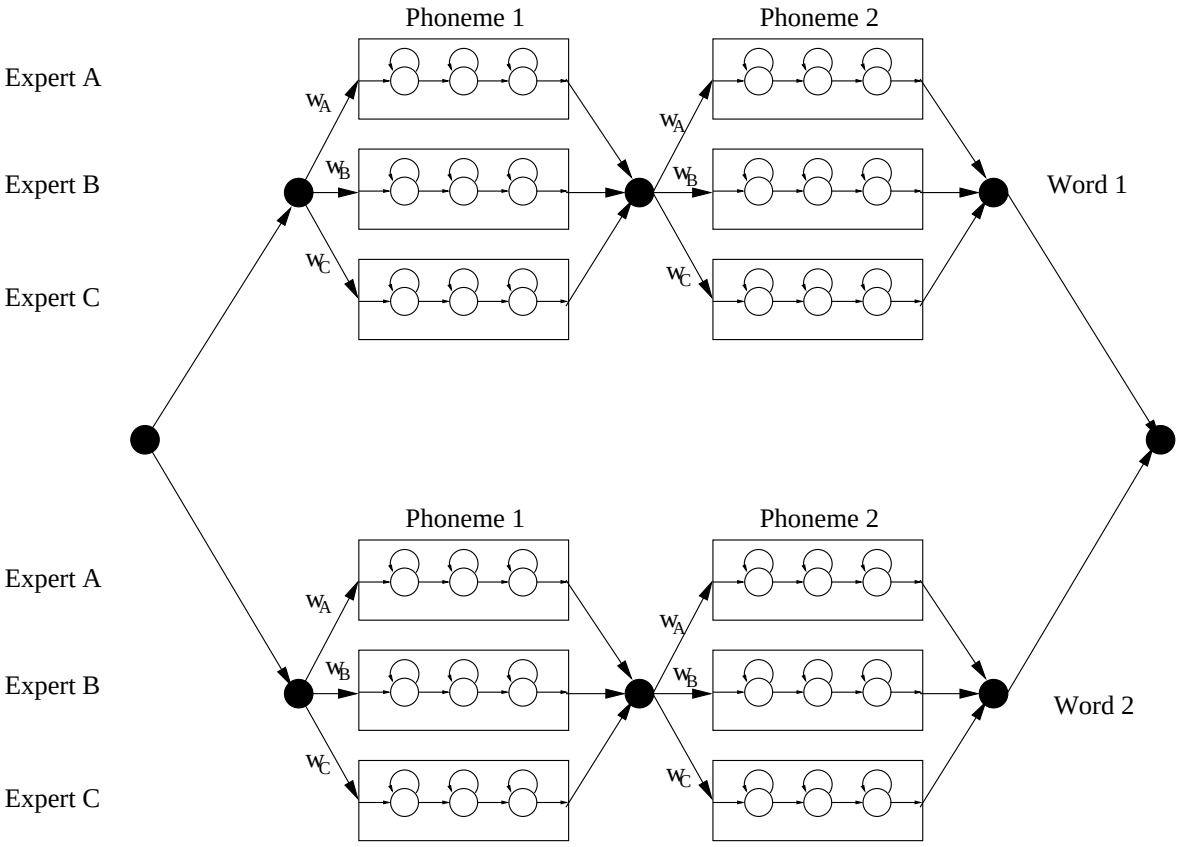

Figure 2: Single-path decoding for two vocabulary words consisting of two phonemes each. When there is only one expert the decoding process is done normally. In the multiple expert case, phoneme models from each expert are connected in parallel. The transition probabilities leading from the anchor states to the Hidden Markov Model corresponding to each experts are calculated from the expert weights β_i of each expert.

Mixture Model employing ten Gaussian distributions. [3]

It then becomes possible to apply the boosting algorithm by using Bayes Classifiers as the experts. The N95 data was pre-segmented into training examples, so that each one was a segment containing a single phoneme. Thus, bootstrapping was performed by sampling through these examples. Furthermore, the classification error of each classifier is used to calculate the weights necessary for the weighted voting mechanism. The data that was used for testing was also segmented to subsequences consisting of single phoneme data, so that the models could be tested on the phoneme classification tasks. The results, shown in Figure 1, were validated against the performance of a single Bayes Classifier that was trained on the complete data set.

5.2 Continuous Speech Recognition

The approach described in section 5.1 is only suitable when the data is segmented at the phoneme level both during training and testing. However we can still employ boosting by training with segmented data to produce a number of expert models which can then be recombined during decoding

[3] These values are within the range commonly employed in phoneme recognition tasks where the data is composed of MFCC features. Since this preliminary work was a proof of concept, there was no attempt to perform cross-validation in order to choose those hyper-parameters optimally.

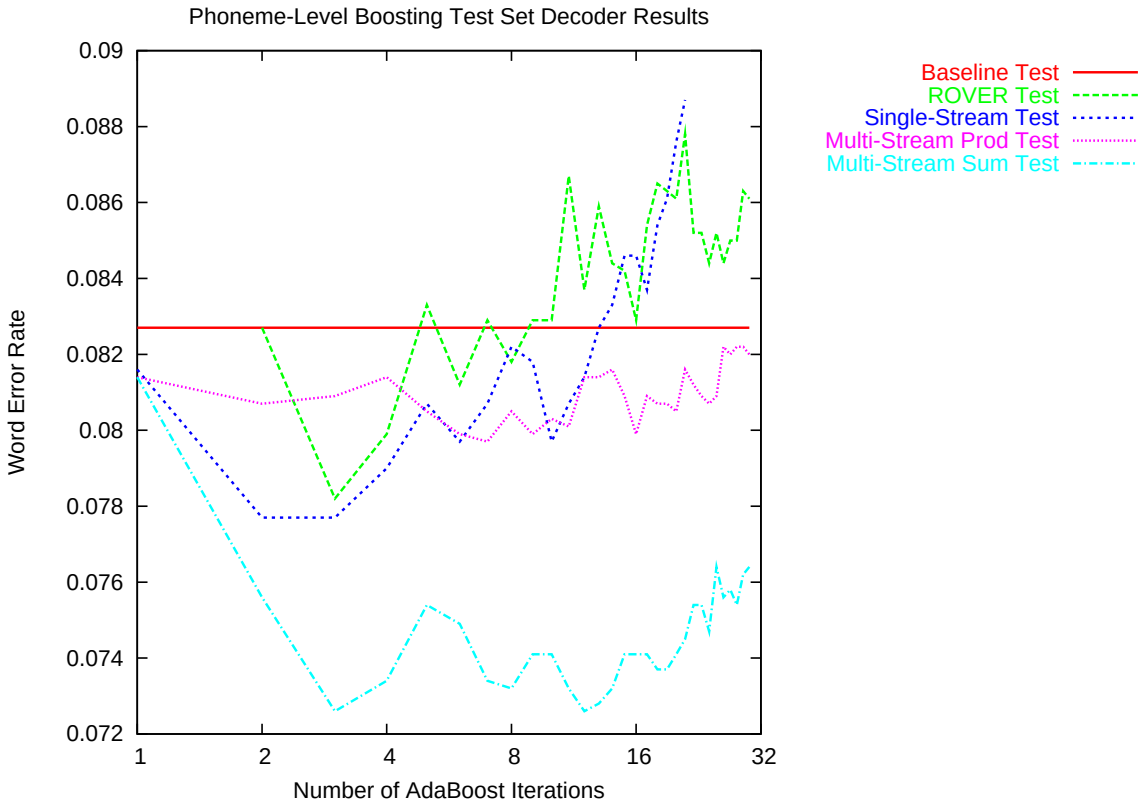

Figure 3: Word error rates in testing when training with segmentation information. Decoding results are shown for a single HMM, and for single-path, multi-stream (product and sum) and ROVER recombination schemes.

on unsegmented data.

The first technique employed for sequence decoding uses an HMM comprising of all phoneme models created during the boosting process, connected in the manner shown in Figure 2. Decoding is performed using the Viterbi algorithm in order to find a sequence of states maximising the likelihood of the sequence. Only single state likelihoods are considered in this case, so only single models are contributing to the final decision. The transition probabilities leading from anchor states to each model are calculated from the boosting weights β_i so that they sum to one and represent the confidence weight of each expert according to:

$$w_i = \frac{\beta_i}{\sum_j^{n_e} \beta_j}. \tag{4}$$

The models may also be combined using multi-stream decoding. This has the advantage of utilising information from all boosted models. The type of multi-stream decoding used was synchronised at the state level, according to either equation 2 or 3,where the stream weights are given by equation 4.

Another method of combining the models created from boosting is to perform decoding separately using each model and then attempt to combine their outputs. Such a technique is employed by ROVER [6], which uses dynamic programming to find an alignment between the sequences of words output by each expert. Experimental results comparing the performance of the above techniques to that of an HMM using segmentation information for training are shown in Figure 3. While there was a consistent performance increase in training (not shown), this did not result in very good generalisation abilities: while model performance was better than that of the baseline system, it was only improving

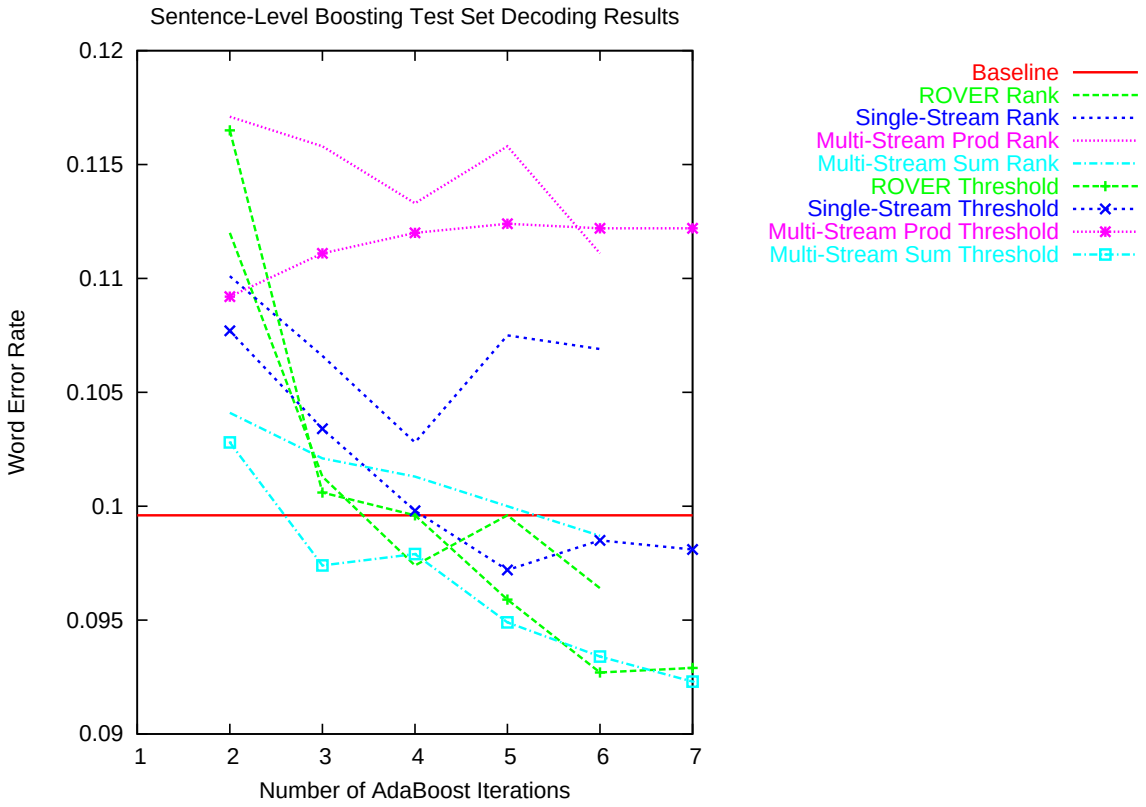

Figure 4: Word error rates in testing when training at the sentence level. Decoding results are shown for a single HMM and for single-path, multi-stream and ROVER recombination schemes for the boosted models. Two types of boosted models are shown, the Rank models, which have been trained with a ranking loss function, and the Threshold models, which have been trained with a thresholded loss function.

for the first 2-3 boosting iterations.

Training was also performed on the sentence level, using no prior segmentation information for the training data. Both the thresholded method and the ranking scheme were investigated as sample loss functions and the resulting models were combined using the techniques outlined above. Since at the sentence level no segmentation information is available, *embedded training* was used to directly adapt the models to target sentences. The results, shown in Figure 4 were compared with the performance of a single HMM. At the sentence level of example selection, boosting fails to improve training performance after six or seven iterations and stops. Generalisation results tend to be better for the weighted sum multi-stream decoding and the ROVER recombination scheme, while in general the thresholded loss function performs better.

6 Conclusion and Future Work

The experimental results indicate that boosting at the phoneme level can significantly improve phoneme classification performance, while also increasing the performance for sentence recognition. While there is only one way to combine the experts for phoneme classification, the task of sentence recognition does not present an obvious (optimal, yet tractable) method for the combination of experts. Indeed, the performance varies significantly depending on the combination method used. An interesting subject

of further research would be to pursue better methods for expert combination during decoding, such as an alternative to the current multi-stream methods, or a Monte Carlo approximation to the full likelihood estimation.

In the case of boosting at the sentence level, there was no consistently significant performance difference with the baseline. This indicates the need for a better formulation of boosting in general sequence recognition tasks, rather than sequence classification tasks. That would include the formulation of sample loss functions and overall cost functions for indefinite length sequences.

References

[1] R. A. Cole, K. Roginski, and M. Fanty. The ogi numbers database. Technical report, Oregon Graduate Institute, 1995.

[2] G. Cook and A. Robinson. Boosting the performance of connectionist large vocabulary speech recognition. In *Proc. ICSLP '96*, volume 3, pages 1305–1308, Philadelphia, PA, 1996.

[3] Harris Drucker. Improving regressors using boosting techniques. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pages 107–115, 1997.

[4] Nigel Duffy and David Helmbold. Boosting methods for regression. *Machine Learning*, 47(2-3):153–200, 2002.

[5] Stéphane Dupont, Hervé Bourlard, and Christophe Ris. Robust speech recognition based on multi-stream features. In *Proc. of ESCA/NATO Workshop on Robust Speech Recognition for Unknown Communication Channels*, pages 95–98, Pont--Mousson, France, April 1997.

[6] J.G. Fiscus. A post-processing system to yield reduced error word rates: Recognizer output voting error reduction (ROVER). In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 347–354, 1997.

[7] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[8] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), 2001.

[9] Ron Meir and Gunnan Rätch. An introduction to boosting and leveraging. In *Advanced Lectures on Machine Learning*, LNCS, pages 119–184. Springer, 2003.

[10] Lawrence R. Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. PTR Prentice-Hall, Inc., 1993.

[11] Gunnar Rätsch, Manfred Warmuth, Sebastian Mika, Takashi Onoda, Steven Lemm, and Klaus-Robert Müller. Barrier boosting. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*. COLT, Morgan Kaufmann, 2000.

[12] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297, 1999.

[13] H. Schwenk. Using boosting to improve a hybrid HMM/neural network speech recogniser. In *Proc. ICASSP '99*, pages 1009–1012, 1999.

[14] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

[15] Richard S. Zemel and Toniann Pitassi. A gradient-based boosting algorithm for regression problems. In *NIPS*, pages 696–702, 2000.