BOOSTING HMMS WITH AN APPLICATION TO SPEECH RECOGNITION

Christos Dimitrakakis and Samy Bengio

IDIAP
CP952
1920 Martigny
Switzerland

ABSTRACT

Boosting is a general method for training an ensemble of classifiers with a view to improving performance relative to that of a single classifier. While the original AdaBoost algorithm has been defined for classification tasks, the current work examines its applicability to sequence learning problems, focusing on speech recognition. We apply boosting at the phoneme model level and recombine expert decisions using multi-stream techniques.

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. Specifically, we are applying boosting to HMM training at the phoneme classification level. This results in an ensemble of models, where each model has been trained on a different part of the data, namely on a subset of the phoneme segments. Ensemble performance is then examined at the phoneme classification level. Furthermore, we investigate a number of multi-stream techniques to recombine the separate models for use in continuous speech recognition. 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.

This work was carried out in the framework of the Swiss National Center of Competence in Research (NCCR) on Interactive Multimodal Information Management (IM2).

2. BOOSTING

Boosting algorithms [1, 2, 3] 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 [3], 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: $p(m_i|S) = \frac{p(S|m_i)p(m_i)}{p(S)}$. 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 [4], 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, so the Viterbi algorithm is used to find the maximum-likelihood state sequence 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_{si} states $\mathcal{Q}_i = \{q_{i,j} | j \in [1, n_{si}]\}$ 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 $S = (S^1, S^2, ..., S^{n_e})$.

We consider the case of state-locked multi-stream decoding, where all sub-models are forced to be at the same state. Thus, 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 [1] for an overview). However, the amount of research in the application of boosting to sequence learning has been comparatively small: In [6], boosting was used in a speech recognition task. That particular system was HMM-based with ANNs for computing the posterior 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 [7], 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. Unlike previous approaches, we apply boosting at the phoneme level, by treating the problem as a phoneme classification task. This results in a number of base models for each phoneme. We explore the use of various methods to combine the models for use in a continuous speech recognition task.

4.1. Model Training At The Phoneme Level

The simplest way to apply boosting to HMM training is to cast the problem into a 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 training 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.

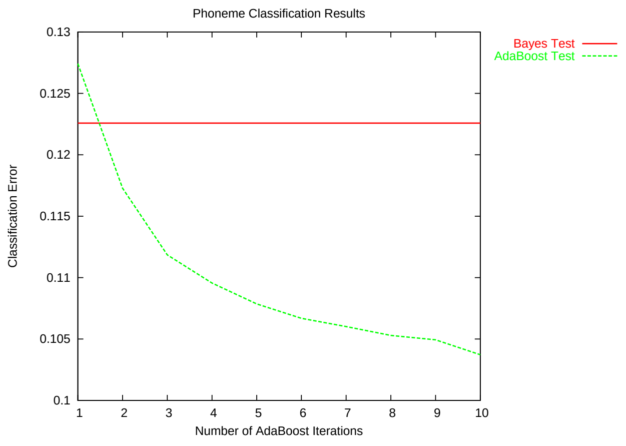

Fig. 1. Classification errors in testing for a boosted ensemble of Bayes Classifiers as the number of experts is increased at each iteration of boosting. For reference, the corresponding errors for a Bayes Classifier trained on the complete training set are also included.

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 [8]. This data set was converted from the original raw audio data into a set of features based on Mel-Frequence Cepstrum Coefficients (MFCC) [4] (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 [1] in a left-to-right topology and the distributions corresponding to each state were modelled with a Gaussian Mixture Model employing ten Gaussian distributions. [2]

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. In this case the classification error is continuously reduced, reaching an error of 10.3%, which, with 95% confidence, is statistically significantly better than the baseline system.

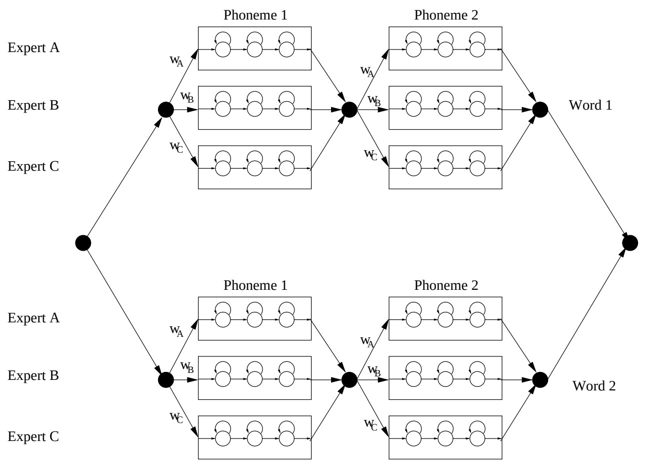

Fig. 2. Single-path decoding for two vocabulary words consisting of two phonemes each. When there is only one expert the decoding process is done normally. In the multiple expert case, phoneme models from each expert are connected in parallel. The transition probabilities leading from the anchor states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

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

[2]These values are within the range commonly employed in phoneme recognition tasks where the data is composed of MFCC features. There was no attempt to perform cross-validation in order to choose those hyperparameters optimally.

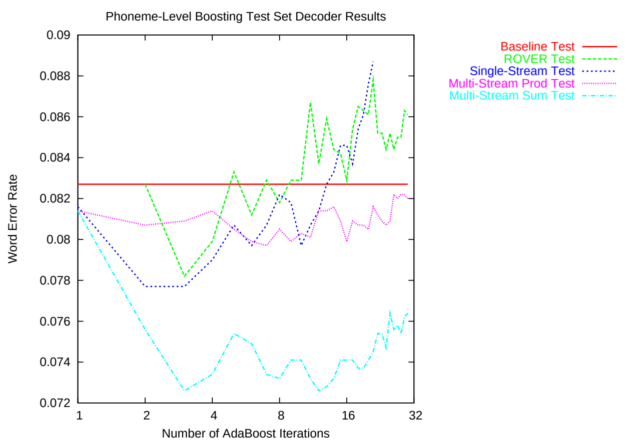

Fig. 3. Word error rates in testing when training with segmentation information. Decoding results are shown for a single HMM, and for single-path and multi-stream (product and sum) recombination schemes.

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 on unsegmented data. We explore three different recombination techniques:'

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. We experimented with two types of multi-stream decoding: exponentially weighted product (equation 2) and weighted sum (equation 3). In both cases the stream weights are given by equation 4.

Experimental results comparing the performance of the above techniques to that of an HMM using segmentation information for training are shown in Figure 3. An improvement on the baseline system was observed for all models, with the weighted-sum multi-stream model performing markedly better. However, the performance deteriorates after 3 iterations. This could be attributed to the effects of noise[3] and the use of state-locked techniques for multi-stream decoding. Application of the same models to the training data yielded substantially better and continuous performance increases, with the weighted-sum model reaching an error of 2.7%, compared to the 7% achieved by the baseline system.

[3]The AdaBoost algorithm is quite robust with respect to noise, but for high noise levels a regularisation method is necessary [1].

6. CONCLUSION AND FUTURE WORK

In this paper we presented a method for the application of boosting to complete HMMs, rather than at the frame level. State-locked multi-stream decoding techniques were investigated for model recombination in a continuous speech recognition task. As was expected, the application of boosting to phoneme classification resulted in a significant performance increase, in both the test and the training error. The same was true of the training error in the continuous speech recognition task. The test error was lower than that of the baseline system with a statistical significance level of 86%.

This lack of good generalisation in decoding might be remedied by pursuing alternative techniques to the state-locked multi-stream methods, such as a Monte Carlo approximation to the full likelihood estimation. Another subject of future research is the application of boosting at the sentence level. This poses additional difficulties, since treating the problem as a classification task is no longer feasible. However it may be possible to apply recently developed boosting techniques for regression problems [1, 9] by defining an appropriate distance metric to be minimised by the regression process, such as the word error rate.

7. REFERENCES

[1] Ron Meir and Gunnar Rätch, "An introduction to boosting and leveraging," in *Advanced Lectures on Machine Learning*, LNCS, pp. 119–184. Springer, 2003.

[2] Robert E. Schapire and Yoram Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297, 1999.

[3] 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*, vol. 55, no. 1, pp. 119–139, 1997.

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

[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*, Pont--Mousson, France, Apr. 1997, pp. 95–98.

[6] H. Schwenk, "Using boosting to improve a hybrid HMM/neural network speech recogniser," in *Proc. ICASSP '99*, 1999, pp. 1009–1012.

[7] G. Cook and A. Robinson, "Boosting the performance of connectionist large vocabulary speech recognition," in *Proc. ICSLP '96*, Philadelphia, PA, 1996, vol. 3, pp. 1305–1308.

[8] R. A. Cole, K. Roginski, and M. Fanty, "The ogi numbers database," Tech. Rep., Oregon Graduate Institute, 1995.

[9] Jerome H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, 2001.