Phoneme and Sentence-Level Ensembles for Speech Recognition*

Christos Dimitrakakis
FIAS, Frankfurt, Germany
christos.dimitrakakis@gmail.com

Samy Bengio
Google, Mountain View, CA, USA
bengio@google.com

March 17, 2011

Abstract

We address the question of whether and how boosting and bagging can be used for speech recognition. In order to do this, we compare two different boosting schemes, one at the phoneme level, and one at the utterance level, with a phoneme level bagging scheme. We control for many parameters and other choices, such as the state inference scheme used. In an unbiased experiment, we clearly show that the gain of boosting methods compared to a single hidden Markov model is in all cases only marginal, while bagging significantly outperforms all other methods. We thus conclude that bagging methods, which have so far been overlooked in favour of boosting, should be examined more closely as a potentially useful ensemble learning technique for speech recognition.

1 Introduction

This paper examines the application of ensemble methods to hidden Markov models (HMMs) for speech recognition. We consider two methods: bagging and boosting. Both methods feature a fixed mixing distribution between the ensemble components, which simplifies the inference, though it does not completely trivialise it.

This paper follows up on and consolidates previous results [10, 11, 12] that focused on boosting. The main contributions are the following. Firstly, we use an unbiased model testing methodology to perform the experimental comparison between the various different approaches. A larger number of experiments, with additional experiments on tri-phones, shed some further light on previous results [11, 12]. Secondly, the results indicate that, in an *unbiased* comparison,

*This work was supported in part by the IST program of the European Community, under the PASCAL Network of Excellence, IST-2002-506778, funded in part by the Swiss Federal Office for Education and Science (OFES) and the Swiss NSF through the NCCR on IM2, and the EU-FP7 project IM-CLeVeR

at least for the dataset and features considered, bagging approaches enjoy a significant advantage to boosting approaches. More specifically, bagging consistently exhibited a significantly better performance than either any of the boosting approaches examined. Furthermore, we were able to obtain state-of-the art results on this dataset using a simple bagging estimator on tri-phone models. This indicates that perhaps a shift towards bagging and perhaps more generally, empirical Bayes methods, may be advantageous for any further advances in speech recognition.

Section 2 introduces notation and provides some background to speech recognition using hidden Markov models. In addition, it discusses multi-stream methods for combining multiple hidden Markov models to perform speech recognition. Finally, it introduces the ensemble methods used in the paper, bagging and boosting, in their basic form.

Section 3 discusses related work and their relation to our contributions, while section 4 gives details about the data and the experimental protocols followed.

In the speech model considered, words are hidden Markov models composed of concatenations of phonetic hidden Markov models. In this setting it is possible to employ mixture models at any temporal level. Section 5 considers mixtures at the phoneme model level, where data with a phonetic segmentation is available. We can then restrict ourselves to a sequence classification problem in order to train a mixture model. Application of methods such as bagging and boosting to the phoneme classification task is then possible. However, using the resulting models for continuous speech recognition poses some difficulties in terms of complexity. Section 5.1 outlines how multi-stream decoding can be used to perform approximate inference in the resulting mixture model.

Section 6 discusses an algorithm, introduced in [12], for word error rate minimisation using boosting techniques. While it appears trivial to do so by minimising some form of loss based on the word error rate, in practice successful application additionally requires use of a probabilistic model for inferring error probabilities in parts of misclassified sequences. The concepts of expected label and expected loss are introduced, of which the latter is used in place of the conventional loss. This integration of probabilistic models with boosting allows its use in problems where labels are not available.

Sections 7 and 8 conclude the paper with an extensive comparison between the proposed models. It is clearly shown that the neither of the boosting approaches employed manage to outperform a simple bagging model that is trained on pre-segmented phonetic data. Furthermore, in a follow-up experiment, we find that the performance of bagging when using tri-phone models achieves state of the art results for the dataset used. These are significant findings, since most of the recent ensemble-based hidden Markov model research on speech recognition has focused invariably on boosting.

2 Background and notation

Sequence learning and sequential decision making deal with the problem of modelling the relationship between sequential variables from a set of data, and then using the models to make decisions. In this paper, we examine two types of sequence learning tasks: sequence classification and sequence recognition.

The sequence classification task entails assigning a sequence to one or more of a set of categories. More formally, we assume a finite label set $\mathcal{Y}$ and a possibly uncountably infinite observation set $\mathcal{X}$. We denote the set of sequences of length n as $\mathcal{X}^n \triangleq \times^n \mathcal{X}$, and the null sequence set by $\mathcal{X}^0 \triangleq \emptyset$. Finally, we denote the set of all sequences by $\mathcal{X}^* \triangleq \bigcup_{n=0}^{\infty} \mathcal{X}^n$. We observe sequences $x = x_1, x_2, \ldots$, with $x_i \in \mathcal{X}$ and $x \in \mathcal{X}^*$ and we use $|x|$ to denote the length of a sequence x, while $x_{t:T} = x_t, x_{t+1}, \ldots, x_T$ denotes subsequences. In sequence classification, each $x \in \mathcal{X}^*$ is associated with a label $y \in \mathcal{Y}$. A sequence classifier $f \in \mathcal{F}$, is a mapping $f : \mathcal{X}^* \to \mathcal{Y}$, such that $f(x)$ corresponds to the predicted label, or classification decision, for the observed sequence x.

We focus on probabilistic classifiers, where the predicted label is derived from the *conditional* probability of the class given the observations, or *posterior class probability* $\mathbf{P}(y \mid x)$, with $x \in \mathcal{X}^*$, $y \in \mathcal{Y}$, where we make no distinction between random variables and their realisations. More specifically, we consider a set of models $\mathcal{M}$ and an associated set of observation densities and class probabilities $\{\, p(x \mid y, \mu), \mathbf{P}(y \mid \mu) \, : \, \mu \in \mathcal{M} \,\}$ indexed by μ. The posterior class probability according to model μ can be obtained by using Bayes' theorem:

$$\mathbf{P}(y \mid x, \mu) = \frac{p(x \mid y, \mu)\, \mathbf{P}(y \mid \mu)}{p(x \mid \mu)}. \tag{1}$$

Any model μ can be used to define a classification rule.

Definition 2.1 (Bayes classifier)**.** *A classifier $f_\mu : \mathcal{X}^* \to \mathcal{Y}$ that employs (1) and makes classification decisions according to:*

$$f_\mu(x) = \arg\max_{y \in \mathcal{Y}} \mathbf{P}(y \mid x, \mu), \tag{2}$$

is referred to as a Bayes classifier, or a Bayes decision rule.

Formally, this task is exactly the same as non-sequential classification. The only practical difference is that the observations are sequences. However, care should be taken as this makes the implicit assumption that the costs of all incorrect decisions are equal.

In *sequence recognition*, we attempt to determine a sequence of events from a sequence of observations. More formally, we are given a sequence of observations x and are required to determine a sequence of labels $y \in \mathcal{Y}^*$, i.e. the sequence $y = y_1, y_2, \ldots, y_k$, $|y| \leq |x|$, with maximum posterior probability $\mathbf{P}(y \mid x)$. In practice, models are used for which it is not necessary to exhaustively evaluate the set of possible label sequences. One such simple, yet natural, class is that of hidden Markov models.

2.1 Speech recognition with hidden Markov models

Definition 2.2 (Hidden Markov model)**.** *A hidden Markov model (HMM) is a discrete-time stochastic process, with state variable s_t in some discrete space $\mathcal{S}$, and an observation variable $x_t \in \mathcal{X}$, such that:*

$$\mathbf{P}(s_t | s_{t-1}, s_{t-2}, \ldots) = \mathbf{P}(s_t | s_{t-1}) \tag{3}$$

$$\mathbf{P}(x_t | s_t, x_{t-1}, s_{t-1}, x_{t-2}, \ldots) = \mathbf{P}(x_t | s_t). \tag{4}$$

The model is characterised by the observation distribution $\mathbf{P}(x_t | s_t)$, the transition distribution $\mathbf{P}(s_t | s_{t-1})$ and the initial state distribution $\mathbf{P}(s_1) \equiv \mathbf{P}(s_1 | s_0)$. These dependencies are shown graphically in Fig. 1.

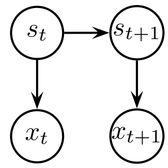


Figure 1: Graphical representation of a hidden Markov model, with arrows indicating dependencies between variables. The observations x_t and the next state s_{t+1} only depend on the current state s_t.

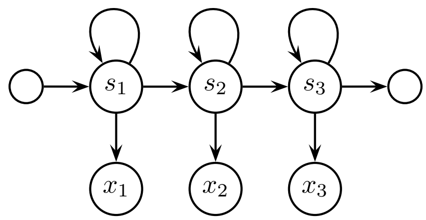


Figure 2: Graphical representation of a phoneme model with 3 emitting states, as well as initial and terminal non-emitting states. The arrows depict dependencies between specific states. All the phoneme models used in this paper employed the above topology.

Training consists of two steps. First, selecting a class of hidden Markov models $\mathcal{M}$, with each model $\mu \in \mathcal{M}$ corresponding to a pair of transition and observation densities $\mathbf{P}(s_t|s_{t-1}, \mu)$, $\mathbf{P}(x_t|s_t, \mu)$. The second step is to select a model from $\mathcal{M}$. By additionally defining a prior density $p(\mu)$ over $\mathcal{M}$, we can try to find the maximum *a posteriori* (MAP) model $\mu^* \in \mathcal{M}$, given a set of observation sequences $\mathcal{D}$:

$$\mu^* = \arg\max_{\mu \in \mathcal{M}} p(\mu|\mathcal{D}).$$

The class $\mathcal{M}$ is restricted to models with a particular number of states and allowed transitions between states. In this paper, the optimisation is performed through expectation maximisation.

The most common way to apply such models to speech recognition is to associate each state s with phonological units $a \in \mathcal{A}$, such as phonemes, syllables, or words, through a distribution $\mathbf{P}(a|s)$, which takes values in $\{0, 1\}$ in usual practice: thus, each state is mapped to only one phoneme. This is done by modelling each phoneme as a small HMM (Fig. 2) and combining them into a larger HMM, such as the one shown in Figure 3, with a set of parallel chains such that each chain maps to one word; for example, given that we are in the state $s = 4$ at some time t, then we are also definitely (i.e. with probability 1) in Word A and Phoneme B at time t. In general, if we can determine the probabilities for sequences of states, we can also determine the most probable sequence of words or phonemes. That is, given a sequence of observations $x_{1:T}$ we calculate the state distribution $\mathbf{P}(s_{1:T}|x_{1:T})$ and subsequently a distribution over phonologies, to wit the probabilities of possible word, syllable or phoneme sequences. Thus, the problem of recognising word sequences is reduced to the problem of state estimation.

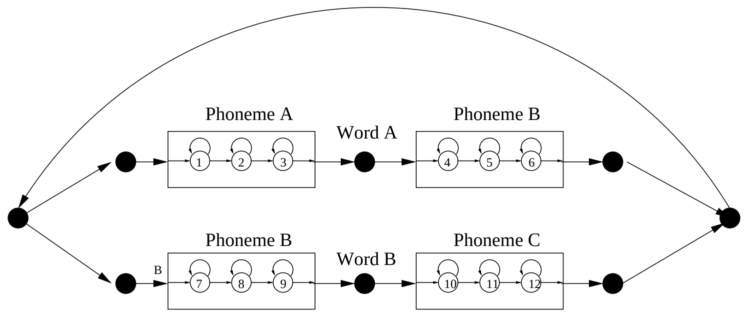


Figure 3: A hidden Markov model for speech recognition. The figure depicts how models of three phonemes, A, B, C are used to construct a single hidden Markov model for distinguishing between two different words. The states are indexed uniquely. Black circles indicate non-emitting states.

2.2 Multi-stream decoding

When we wish to combine evidence from n different models, state estimation is significantly harder, as the number of effective states is $|\mathcal{S}|^n$. However, multi-stream decoding techniques can be used as an approximation to the full mixture model [29]. Such techniques derive their name from the fact that they were originally used to combine models which had been trained on different streams of data or features [27]. In this paper, we instead wish to combine evidence from models trained on different samples of the same data.

In multi-stream decoding each sub-unit model corresponding to a phonological unit a is comprised of n sub-models $a = \{ a_i : i \in [1, n] \}$ associated with the sub-unit level at which the recombination of the input streams should be performed. For any given a, and a distribution over models $\pi(a_i \mid a)$, the observation density conditioned on the unit a can be written as

$$\pi(x \mid a) = \sum_{i=1}^{n} p(x \mid a_i)\pi(a_i \mid a), \tag{5}$$

where $\pi(a_i \mid a)$ can be seen as a weight for expert i. This may vary across a, but herein we consider the case where the weight is fixed, i.e. $\pi(a_i \mid a) = w_i$ for all a. We consider *state-locked* multi-stream decoding, where all sub-models are forced to be at the same state. This can be viewed as creating another Markov model with emission distribution:

$$\pi(x_t \mid s_t, a) = \sum_{i=1}^{n} p(x_t \mid s_t, a_i)\pi(a_i \mid a). \tag{6}$$

An alternative is the exponentially weighted product of emission distributions.

$$\pi(x_t \mid s_t, a) = \prod_{i=1}^{n} p(x_t \mid s_t, a_i)^{\pi(a_i|a)}. \tag{7}$$

However this approximation does not arise from (5), but from assuming a factorisation of the observations $p(x_t \mid s_t) = \prod_{i=1}^{n} p(x_t^i \mid s_t)$, which is useful when there is a different model for different parts of the observation vector.

Multi-stream techniques are hardly limited to the above. For example Misra et al. [28] describes a system where π is related to the entropy of each sub-model, while Ketabdar et al. [20] describes a multi-stream method utilising state posteriors. We, however, shall concentrate on the two techniques outlined above, as well as a single-stream technique to be described in Section 5.1.

2.3 Ensemble methods

We investigate the use of ensemble methods in the class of *static* mixture models for speech recognition. Such methods construct an aggregate model from a set of base hypotheses $M \triangleq \{ \mu_i : i = 1, \ldots, N \}$. Each hypothesis μ_i indexes a set of conditional distributions $\{ \mathbf{P}(\cdot|\cdot, \mu_i) : i = 1, \ldots, N \}$. To complete the model, we employ a set of weights $W \triangleq \{ w_i : i = 1, \ldots, N \}$ corresponding to the probability of each base hypothesis, so that $w_i \triangleq \mathbf{P}(\mu_i)$. Thus, we can form a mixture model, assuming $\mathbf{P}(\mu_i \mid x) = \mathbf{P}(\mu_i)$ for all $x \in \mathcal{X}^*$:

$$\mathbf{P}(\cdot \mid \cdot, M, W) = \sum_{i=1}^{N} w_i \, \mathbf{P}(\cdot \mid \cdot, \mu_i). \tag{8}$$

Two questions that arise when training such models are how to select M and W. In this paper, we consider two different approaches, bagging and boosting.

2.3.1 Bagging

Bagging [5] can be seen as a method for sampling the model space $\mathcal{M}$. We first require a learning algorithm $\Lambda : (\mathcal{X}^* \times \mathcal{Y})^* \to \mathcal{M}$ that maps[1] from a *dataset* $D \in (\mathcal{X}^* \times \mathcal{Y})^*$ of data pairs (x, y) to models $\mu \in \mathcal{M}$. We then sample N datasets D_i from a distribution $\mathcal{D}$, for $i = 1, \ldots, N$. For each D_i, the learning algorithm Λ generates a model $\mu_i \triangleq \Lambda(D_i)$. The models $M \triangleq \{ \mu_i : i = 1, \ldots, N \}$ can be combined into a mixture with $w_i = \frac{1}{N}$ for all i:

$$\mathbf{P}(y|x, M, W) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{P}(y|x, \mu_i). \tag{9}$$

In Bagging, D_i is generated by sampling with replacement from the original dataset D, with $|D_i| = |D|$. Thus, D_i is a *bootstrap replicate* of D.

2.3.2 Boosting

Boosting algorithms [16, 25, 32] are another family of ensemble methods. The most commonly used boosting algorithm for classification is Ada-Boost [16]. Though many variants of Ada-Boost for multi-class classification problems exist, in this work we will use Ada-Boost.M1.

An Ada-Boost ensemble is a mixture model composed of N models μ_i and weights w_i, as in the previous section. The models and weights are created in an iterative manner. At iteration j, the model $\mu_j \triangleq \Lambda(D_j)$ is created from a *weighted* bootstrap sample D_j of the training dataset $D = \{ d_i : i \in [1, n] \}$, with $d_i = (x_i, y_i)$. The probability of adding example d_i to the bootstrap

[1] While we restrict our selves to the deterministic case for simplicity, bagging is applicable to stochastic learning algorithms as well.

replicate D_j is denoted as $p_j(d_i)$, with $\sum_i p_j(d_i) = 1$. At the end of iteration j of Ada-Boost.M1, β_j is calculated according to:

$$\beta_j = \ln \frac{1 - \varepsilon_j}{\varepsilon_j}, \tag{10}$$

where $\varepsilon_j \triangleq \sum_i p_j(d_i)\ell(d_i)$ is the empirical expected loss of the j-th, with $\ell(d_i) \triangleq \mathbb{I}\{h_i \neq y_i\}$ being the *sample loss* of example d_i, where $\mathbb{I}\{\cdot\}$ is an indicator function. At the end of each iteration, sampling probabilities are updated according to:

$$p_{j+1}(d_i) = \frac{p_j(d_i)\exp(\beta_j \ell(d_i))}{Z_j}, \tag{11}$$

where $Z_j \triangleq \sum_i p_j(d_i)\exp(\beta_j \ell(d_i))$, is a normalisation factor. Thus, incorrectly classified examples are more likely to be included in the next bootstrap data set. The final model is a mixture with N components μ_i and weights $w_i \triangleq \beta_i / \sum_{j=1}^{N} \beta_j$.

3 Contributions and related work

The original Ada-Boost algorithm had been defined for classification and regression tasks, with the regression case receiving more attention recently (see [25] for an overview). In addition, research in the application of boosting to sequence learning and speech recognition has intensified [6, 26, 37, 40]. The application of other ensemble methods, however, has been limited to random decision trees [7, 35]. In our view, bagging [5] is a method that has been somewhat unfairly neglected and we present results that show that it can outperform boosting in an unbiased experiment.

One of the simplest ways to apply ensemble methods to speech recognition is to employ them at the state level. For example Schwenk [34] proposed a HMM/artificial neural network (ANN) system, with the ANNs used to compute the posterior phoneme probabilities at each state. Boosting itself was performed at the ANN level, using Ada-Boost with confidence-rated predictions, using the frame error rate as the sample loss function. The resulting decoder system differed from a normal HMM/ANN hybrid in that each ANN was replaced by a mixture of ANNs that had been provided via boosting. Thus such a technique avoids the difficulties of performing inference on mixtures, since the mixtures only model instantaneous distributions. Zweig and Padmanabhan [42] appear to be using a similar technique, based on Gaussian mixtures. The authors additionally describe a few boosting variants for large-scale systems with thousands of phonetic units. Both papers report mild improvements in recognition.

One of the first approaches to utterance-level boosting is due to Cook and Robinson [9], who employed a boosting scheme where the sentences with the highest error rate were classified as 'incorrect' and the rest 'correct', irrespective of the absolute word error rate of each sentences. The weights of all frames constituting a sentence were adjusted equally and boosting was applied at the frame level. This however does not manage to produce as good results as the other schemes described by the authors. In our view, which is partially supported by the experimental results in Section 6, this could have been partially due to the lack of a temporal credit assignment mechanism such as the one we present. An early example of a non-boosting approach for the reduction of word error rate is [2], which employed a "corrective training scheme".

In related work on utterance-level boosting, Zhang and Rudnicky [39] compared use of the posterior probability of each possible utterance for adjusting the weights of each utterance with a "non-boosting" method where the same weights are adjusted according to some function of the word error rate. In either case, utterance posterior probabilities are used for recombining the experts. Since the number of possible utterances is very large, not all possible utterances are used, but an N-best list. For recombination, the authors consider two methods: Firstly, choosing the utterance with maximal sum of weighted posterior (where the weights have been determined by boosting). Secondly, they consider combining via ROVER, a dynamic programming method for combining multiple speech recognisers [see 15]. Since the authors' use of ROVER entails using just one hypothesis from each expert to perform the combination, in [40] they consider a scheme where the N-best hypotheses are reordered according to their *estimated* word error rate. In further work [41] the authors consider a boosting scheme for assigning weights to frames, rather than just to complete sentences. More specifically, they use the currently estimated model to obtain the probability that the correct word has been decoded at any particular time, i.e. the posterior probability that the word at time t is a_t given the model and the sequence of observations. In our case we use a slightly different formalism in that we calculate the expectation of the loss according to an independent model.

Finally, Meyer and Schramm [26] propose an interesting boosting scheme with a weighted sum model recombination. More precisely, the authors employ Ada-Boost.M2 at the utterance level, utilising the posterior probability of each utterance for the loss function. Since the algorithm requires calculating the posterior of every possible class (in this case an utterance) given the data, exact calculation is prohibitive. The required calculation however can be approximated by calculating the posterior only for the subset of the top N utterances and assuming the rest are zero. Their model recombination scheme relies upon treating each expert as a different pronunciation model. This results in essentially a mixture model in the form of equation 5, where the weight of each expert is derived from the boosting algorithm. They further robustify their approach through a language model. Their results indicate a slight improvement (in the order of 0.5%) in a large vocabulary continuous speech recognition experiment.

More recently, an entirely different and interesting class of complementary models were proposed in [6, 7, 35]. The core idea is the use of randomised decision trees to create multiple experts, which allows for more detailed modelling of the strengths and weaknesses of each expert, while [6] presents an extensive array of methods for recombination during speech recognition. Other recent work has focused on slightly different applications. For example, a boosting approach for language identification was used in [23, 37], which utilised an ensemble of Gaussian mixture models for both the *target class* and the *anti-model*. In general, however, bagging methods, though mentioned in the literature, do not appear to be used and recent surveys, such as [6, 17, 38] do not include discussions of bagging.

3.1 Our contribution

This paper presents methods and results for the use of both boosting and bagging for phoneme classification and speech recognition. Apart from synthesising and extending our previous results [11, 12], the main purpose of this paper is to present an *unbiased* experimental comparison between a large number of methods, controlling for the appropriate choice of hyper-parameters and using a principled statistical methodology for the evaluation of the significance of the results. If this is not done, then it is possible to draw incorrect conclusions.

Section 5 describes our approach for phoneme-level training of ensemble methods (boosting and bagging). In the phoneme classification case, the formulation of the task is essentially the same as that of static classification; the only difference being that the observations are sequences rather than single values. As far as we know, our past work [11] is the only one employing ensemble methods at the phoneme level. In Section 5, we extend our previous results by comparing boosting and bagging in terms of both classification and recognition performance and show, interestingly, that bagging achieves the same reduction in recognition error rates as boosting, even though it cannot match boosting's classification error rate reduction. In addition, the section compares a number of different multi-stream decoding techniques.

Another interesting way to apply boosting is to use it at the sentence level, for the purposes of explicitly minimising the word error rate. Section 6 presents a boosting-based approach to minimise the word error rate originally introduced in [12].

Finally, Section 7 presents an extensive, unbiased experimental comparison, with separate model selection and model testing phase, between the proposed methods and a number of baseline systems. This shows that the simple phoneme-level bagging scheme outperforms all of the other boosting schemes explored in this paper them significantly. Finally, further results using tri-phone models, indicate state-of-the-art performance is achievable for this dataset using bagging, but not boosting.

4 Data and methods

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-Frequency Cepstrum Coefficients (MFCC) [30] (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 (or 80 tri-phones in the tri-phone version of the dataset) that compose 30 dictionary words. There are of 3233 training utterances and 1206 test utterances, containing 12510 and 4670 words respectively. The segmentation of the utterances into their constituent phonemes resulted in 35562 training segments and 12613 test segments, totalling 486537 training frames and 180349 test frames respectively. The feature extraction and phonetic labelling is described in more detail in [19].

4.1 Performance measures

The comparative performance measure used depends on the task. For the phoneme classification task, the *classification error* is used, which is the percentage of misclassified examples in the training or testing data set. For the speech recognition task, the *word error rate* (12) is used:

$$WER = \frac{N_{\text{ins}} + N_{\text{sub}} + N_{\text{del}}}{N_{\text{words}}}, \tag{12}$$

where N_{ins} is the number of word insertions, N_{sub} the number of word substitutions and N_{del} the number of word deletions. These numbers are determined by finding the minimum number of insertions, substitutions, or deletions necessary to transform the target utterance into the emitted utterance for each example and then summing them for all the examples in the set.

4.2 Bootstrap estimate for speech recognition

In order to establish the significance of the reported results, we employ a bootstrap method [c.f. 14]. More specifically, we use the approach suggested by Bisani and Ney [3] for speech recognition. It amounts to using the results of speech recognition on a test set of sentences as an empirical distribution of errors. Using this method, we obtain a bootstrap estimate of the probability distribution of the difference in word error rate ΔW between two systems, from B bootstrap samples ΔW_k of the word error rate difference:

$$\mathbf{P}(\Delta W > u) = \int_u^\infty p(\Delta W)\, d\Delta W \approx \frac{1}{B} \sum_{k=1}^{B} \mathbb{I}\left\{\Delta W_k > u\right\}, \tag{13}$$

where $\mathbb{I}\{\cdot\}$ is an indicator function. This approximates the probability that system A is better than system B by more than u. See [14] for more on the properties of the bootstrap and [36] for the convergence of empirical processes and their relation to the bootstrap.

4.3 Parameter selection

The models employed have a number of hyper-parameters. In order to perform unbiased comparisons, we split the training data into a smaller training set of 2000 utterances and a hold-out set of 1233 utterances. For the preliminary experiments performed in Sections 5 and 6, we train all models on the small training set and report the performance on both the training and the hold-out set. For the experiments in Section 7, each model's hyperparameters are selected independently on the hold-out set. Then the model is trained on the complete training set and evaluated in the independent test set.

For the classification task (Sec. 5), we used pre-segmented data. Thus, the classification could be performed using a Bayes classifier composed of 27 Hidden Markov Models, each one corresponding to one class. Each phonetic HMM was composed of the same number of hidden states,[2] in a left-to-right topology and the distributions corresponding to each state were modelled with a Gaussian

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

mixture model, with each Gaussian having a diagonal covariance matrix. In Sec. 5.2, we select the number of states per phoneme from $\{1, 2, 3, 4, 5\}$, and the mixture components from $\{10, 20, 30, 40\}$ in the hold-out set for a single HMM and then examine whether bagging or boosting can improve the classification or speech recognition performance.

In all cases, the diagonal covariance matrix elements of each Gaussian were clamped to a lower limit of 0.2 times the global variance of the data. For continuous speech recognition, transitions between word models incurred an additional likelihood penalty of $\exp(-15)$ while calculating the most likely sequence of states. Finally, in all continuous speech recognition tasks, state sequences were constrained to remain in the same phoneme for at least three acoustic frames.

For phoneme-level training, the adaptation of each phoneme model was performed in two steps. Firstly, the acoustic frames belonging to each phonetic segment were split into a number of equally-sized intervals, where the number of intervals was equal to the number of states in the phonetic model. The Gaussian mixture components corresponding to the data for each interval were initialised via 25 iterations of the K-means algorithm (see, for example [4]). After this initialisation was performed, a maximum of 25 iterations of the EM algorithm were run on each model, with optimisation stopping earlier if at any point in time t, the likelihood L_t satisfied the stopping criterion $(L_t - L_{t-1})/L_t < \epsilon$, with $\epsilon = 10^{-5}$ being used in all experiments that employed EM for optimisation.

For the utterance-level training described in Section 6, the same initialisation was performed. The inference of the final model was done through expectation maximisation (using the Viterbi approximation) on concatenated phonetic models representing utterances. Note that performing the full EM computation is costlier and does not result in significantly better generalisation performance, at least in this case. The stopping criterion and maximum iterations were the same as those used for phoneme-level training.

Finally, the results in Section 7 present an unbiased comparison between models. In order to do this, we selected the parameters of each model, such as the number of Gaussians and number of experts, using the performance in the hold out set. We then used the selected parameters to train a model on the full training dataset. The models were evaluated on the separate testing dataset and compared using the bootstrap estimate described in Section 4.2.

5 Phoneme level bagging and boosting

A simple way to apply ensemble techniques such as bagging and boosting is to cast the problem into the classification framework. This is possible at the phoneme level, where each class $y \in \mathcal{Y}$ corresponds to a phonemes. As long as the available data are annotated so that subsequences containing single phoneme data can be extracted, it is natural to adapt each hidden Markov model μ_y to a single class y out of the possible $|\mathcal{Y}|$, where $|\cdot|$ denotes the cardinality of the set, and combine the models into a Bayes classifier in the manner described in Section 2. Such a Bayes classifier can then be used as an expert in an ensemble.

In both cases, each example d in the training dataset D is a sequence segment corresponding to data from a single phoneme. Consequently, each example d has the form $d = (x, y)$, with $x \in \mathcal{X}^*$ being a sub-sequence of features corresponding to single phoneme data and $y \in \mathcal{Y}$ being a phoneme label.

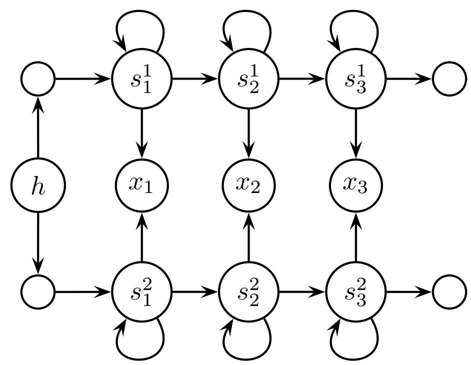


Figure 4: A phoneme mixture model. The generating model depends on the hidden variable h, which determines the mixing coefficients between model 1 and 2. The random variable h may in general depend on other variables. The distribution of the observation is a mixture between the two distributions predicted by the two hidden models, mixed according to the mixture model h.

Both methods iteratively construct an ensemble of N models. At each iteration j, a new classifier h_j is created, consisting of a set of hidden Markov models: $h_j = \{\mu_1^j, \mu_2^j, ..., \mu_{|\mathcal{Y}|}^j\}$. Each model μ_y^j is adapted to the set of examples $\{d_k \in \mathcal{D}_j \mid y_k = y\}$, where $\mathcal{D}_j$ is a bootstrap replicate of $\mathcal{D}$. In order to make decisions, the experts are weighted by the mixture coefficients $w_i \triangleq \pi(h_i)$. The only difference between the two methods is the distribution that $\mathcal{D}_j$ is sampled from and the definition of the coefficients.

For **bagging**, $\mathcal{D}_j$ is sampled *uniformly* from $\mathcal{D}$ and the probability over the mixture components is also uniform, i.e. $\pi(h_i) = N^{-1}$.

For **boosting**, $\mathcal{D}_j$ is sampled from $\mathcal{D}$ using the distribution defined in equation (11), in p. 7, while the expert weights are defined as $\pi(h_i) = \beta_i / \sum_j \beta_j$, where β is given by (10), in p. 7. The Ada-Boost method used was Ada-Boost.M1.

Since previous studies in non-sequential classification problems had shown that an increase in generalisation performance may be obtained through the use of those two ensemble methods, it was expected that they would have a similar effect on performance in phoneme classification tasks. This is tested in Section 5.2. While using the resulting phoneme classification models for continuous speech recognition is not straightforward, we describe some techniques for combining the ensembles resulting from this training in order to perform sequence recognition in Section 5.1.

5.1 Continuous speech recognition with mixtures

The approach described is easily suitable for phoneme classification, since each phonetic model is now a mixture model (Fig. 4), which can be used to classify phonemes given pre-segmented data. However, the phoneme mixtures can also be combined into a speech recognition mixture. Thus, we can still employ ensemble methods for the full speech recognition problem by training with segmented data to produce a number of expert models which can then be recombined during decoding on unsegmented data.

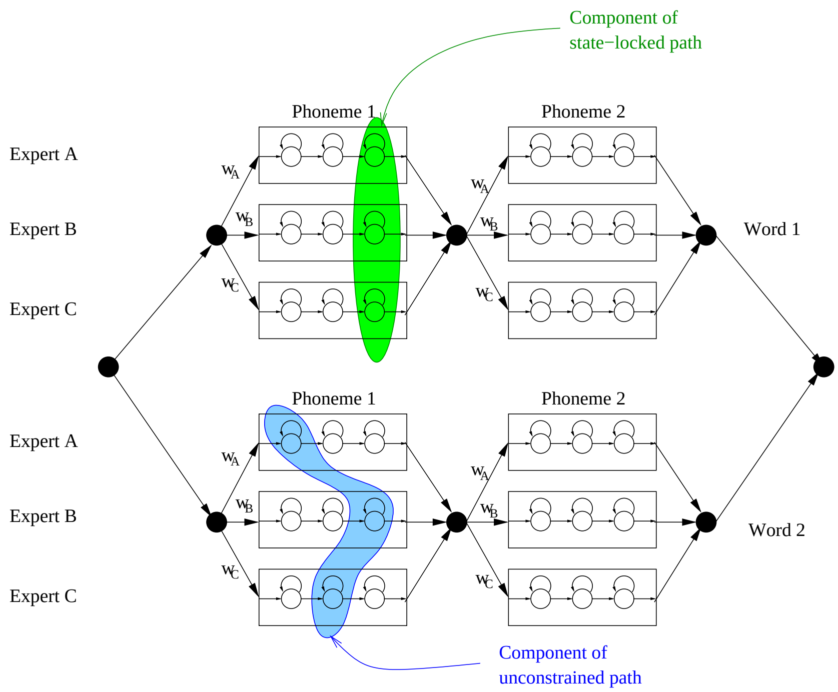


Figure 5: Single-path multi-stream decoding for two vocabulary words consisting of two phonemes each. When there is only one expert the decoding process is done normally. In the multiple expert case, phoneme models from each expert are connected in parallel. The transition probabilities leading from the anchor states to the Hidden Markov Model corresponding to each experts are the weights w_i of each expert.

The first technique employed for sequence decoding uses an HMM comprising all phoneme models created during the boosting process, connected in the manner shown in Figure 5. Each phase of the boosting process creates a sub-model i, which we will refer to as *expert* for disambiguation purposes. Each expert is a classification model that employs one hidden Markov model for each phoneme. For some sequence of observations, each expert calculates the posterior probability of each phonetic class given the observation and its model. Two types of techniques are considered for employing the models for inferring a sequence of words.

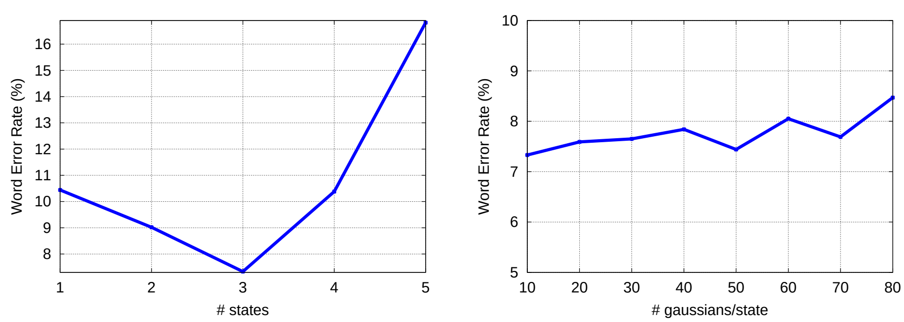
In the *single stream* case, decoding is performed using the Viterbi algorithm in order to find a sequence of states maximising the posterior probability of the sequence. A normal hidden Markov model is constructed in the way shown in Figure 5, with each phoneme being modelled as a mixture of expert models. In this case we are trying to find the sequence of states $\{s_t = s_i^j\}$ with maximum likelihood. The transition probabilities leading from anchor states (black circles in the figure) to each model are set to $w_i = \pi(h_i)$.

This type of decoding would have been appropriate if the original mixture had been inferred as a type of switching model, where only one sub-model is responsible for generating the data at each point in time and where switching between models can occur at anchor states.

The models may also be combined using *multi-stream* decoding (see section 2.2). The advantage of such a method is that it uses information from all models. The disadvantage is that there are simply too many states to be considered. In order to simplify this, we consider multi-stream decoding synchronised at the state level, i.e. with the constraint that $P(s_t^i \neq s_t^j) = 0$ if $j \neq i$. This corresponds to equation (5), where the weight of stream i are again w_i.

5.2 Experiments with boosting and bagging phoneme level models

The experiments described in this section were performed with a fixed number of states for all phonemes, as well as with a fixed number of Gaussians per state. The selection of these hyper-parameters was performed on a hold-out set, as described in Section 4. The hold-out set results are shown in Figure 6. After selecting those hyper-parameters, we perform an exploratory comparison[3] of the performance of boosting and bagging as the number of mixture components are increased, for the tasks of phoneme classification and speech recognition. For the latter problem, we also examine the relative merits of different decoding techniques.



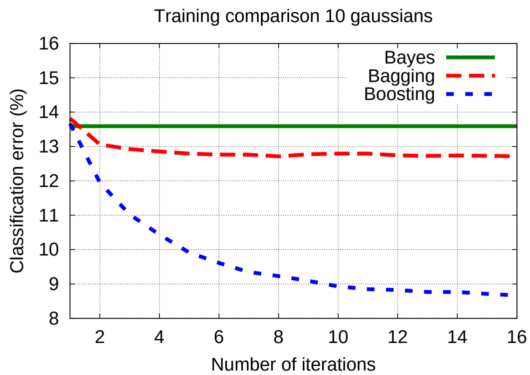
(a) Hold-out set, 10 Gaussians / state

(b) Hold-out set, 3 states / phoneme

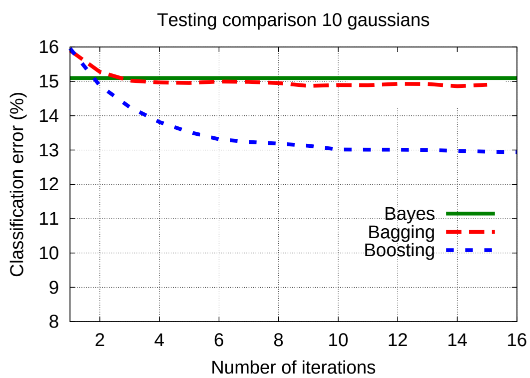
Figure 6: In the experiments reported in this section, the number of states and number of Gaussian mixtures per state were tuned on a hold-out set prior to the analysis. Figure 6a displays the word error rate performance of an HMM with 10 Gaussians per state when the number of emitting states per phoneme is varied, with rather dramatic effects. Figure 6b displays the word error rate performance of an HMM with 3 emitting states as the number of Gaussians per state varies. In this case, the effect on generalisation is markedly lower.

Since the available data includes segmentation information it makes sense to first limit the task to training for phoneme classification. This enables the direct application of ensemble training algorithms by simply using each segment as a training example.

[3]An experiment that uses an unbiased procedure to select the number of experts independently for boosting and bagging, is described in Section 7.



(a) Training



(b) Holdout

Figure 7: Classification errors for a bagged and a boosted ensemble of Bayes Classifiers as the number of experts is increased. For reference, the corresponding errors for a single Bayes Classifier trained on the complete training set are also included. There were 10 Gaussians per state and 3 states per phoneme for all models.

Two methods were examined for this task: bagging and boosting. At each iteration of either method, a sample from the training set was made according to the distribution defined by either algorithm and then a Bayes classifier composed of $|\mathcal{Y}|$ hidden Markov models one for each phonetic class $y \in \mathcal{Y}$, was trained.

It then becomes possible to apply the boosting and bagging algorithms 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. Bootstrapping was performed by sampling through these examples. The classification error of each classifier was used to calculate the boosting weights. The test data was also segmented in subsequences consisting of single phoneme data, so that the models could be tested on the phoneme classification tasks.

Figure 7 compares the classification performance of bagging and boosting as the number of experts increases with that of the Bayes classifier trained on the full training data. As can be seen in Fig. 7a, both bagging and boosting manage to reduce the phoneme classification error considerably in the training,

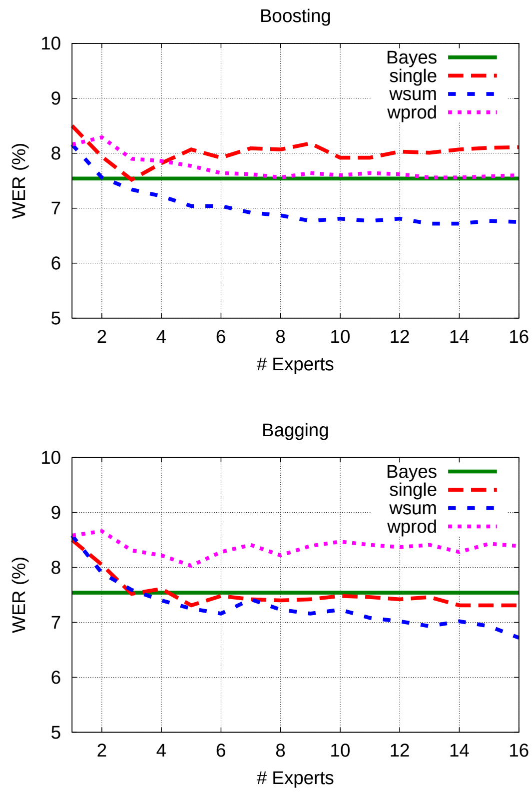


Figure 8: Generalisation performance on the hold-out set in terms of word error rate after training with segmentation information. Results are shown for both boosting and bagging, using three different methods for decoding. Single-path and multi-stream. Results are shown for three different methods single-stream (**single**), and state-locked multi-stream using either a weighted product (**wprod**) or weighted sum (**wsum**) combination.

with boosting continuing to make improvements until the maximum number of iterations. For bagging, the improvement in classification was limited to the first 4 iterations, after which performance remained constant. The situation was similar when comparing the models in the hold-out set, shown in Figure 7b. There, however, bagging failed to improve upon the baseline system.

Finally, an exploratory comparison between the models on the task of continuous speech recognition was made. This was necessary, in order to decide on a method for performing decoding when dealing with multiple models. The three relatively simple methods of single stream and multi-stream decoding (the latter employing either weighted product or weighted sum) were evaluated on the hold-out set. As can be seen in Figure 8, the weighted sum method consistently performed the best for both bagging and boosting. This was expected since it was the only method with some justification in our particular case, as it arises out of constraining the full state inference problem on the mixture. The multi-stream product method is not justified here, since each model had exactly the same observation variables. The single-stream model could perhaps be justified under the assumption of a switching model, where a different expert can be responsible for the observations in each phoneme. That might explain the fact that its performance is not degrading in the case of bagging, as the components of each mixture should be quite similar to each other, something which is definitely not the case with boosting where each model is trained on a different distribution of the data.

A fuller comparison between bagging and boosting at the phoneme level will be given in Section 7, where the number of Gaussian units per state and the number of experts will be independently tuned on the hold-out set, and evaluated on a separate test set. There, it will be seen that with an *unbiased* hyperparameter selection, bagging actually outperforms boosting.

6 Expectation boosting for WER minimisation

It is also possible to apply ensemble training techniques at the utterance level. As before, the basic models used are HMMs that employ Gaussian mixtures to represent the state observation distributions. Attention is restricted to boosting algorithms in this case. In particular, we shall develop a method that uses boosting to simultaneously utilise information about the complete utterance, together with an estimate about the phonetic segmentation. Since this estimate will be derived from bootstrapping our own model, it is unreliable. The method developed will take into account this uncertainty.

More specifically, similarly to [9], sentence-level labels (sequences of words without time indications) are used to define the error measure that we wish to minimise. The measure used is related to the word error rate, as defined in (12). In addition to a loss function at the sentence level, a probabilistic model is used to define a distribution for the loss at the frame level. Combined, the two can be used for the greedy selection of the next base hypothesis. This is further discussed in the following section.

6.1 Boosting for word error rate minimisation

In the previous section (and [11]) we have applied boosting to speech recognition at the phoneme level. In that framework, the aim was to reduce the *phoneme classification error* in pre-segmented examples. The resulting boosted phoneme models were combined into a single speech recognition model using *multi-stream* techniques. It was hoped that we could reduce the word error rate as a side-effect of performing better phoneme classification and three different approaches were examined for combining the models in order to perform continuous speech recognition. However, since the measure that we are trying to improve is the word error rate and since we did not want to rely on the existence of segmentation information, minimising the word error rate directly would be desirable. This section describes such a scheme using boosting techniques.

We describe a training method we introduced in [12], specific to boosting and hidden Markov models (HMMs), for word error rate reduction. We employ a score that is exponentially related to the word error rate of a sentence example. The weights of the frames constituting a sentence are adjusted depending on our expectation of how much they contribute to the error. Finally, boosting is applied at the sentence and frame level simultaneously. This method has arisen from a two-fold consideration: firstly, we need to directly minimise the relevant measure of performance, which is the word error rate. Secondly, we need a way to more exactly specify which parts of an example most probably have contributed to errors in the final decision. Using boosting it is possible to focus training on parts of the data which are most likely to give rise to errors, while at the same time doing it in such a manner as take into account the actual performance measure. We find that both aspects of training have an important effect.

Section 6.1.1 describes word error rate-related loss functions that can be used for boosting. Section 6.1.2 introduces the concept of *expected error*, for the case when no labels are given for the examples. This is important for the task of word error rate minimisation. Previous sections on HMMs and multi-stream decoding, described how the boosted models are combined for performing the speech recognition task. Experimental results are outlined in section 6.2. We conclude with an experimental comparison between different methods in Section 7, followed by a discussion.

6.1.1 Sentence loss function

A commonly used measure of optimality for speech recognition tasks is the word error rate (12). We would like to minimise this quantity using boosting techniques. In order to do this, a dataset is considered where each example is a complete sentence and where the *loss* $\ell(d)$ for each example d is given by some function of the word error rate for the sentence.

The word error rate for any particular sentence can take values in $[0, \infty)$, while the Ada-Boost algorithm that is employed herein requires a sample loss function with range $[-1, 1]$. For this reason we employ the *ad hoc*, but reasonable, mapping $\ell : [0, \infty) \to [-1, 1)$

$$\ell(x) = 1 - 2e^{-\eta x}, \tag{14}$$

where x is the word error rate. When $\ell(x) = -1$, an example is considered as classified correctly and when $\ell(x) = 1$, the example is considered to be classified incorrectly. This mapping includes a free parameter $\eta > 0$. Increasing the parameter η increases the sharpness of the transition, as shown in Figure 9. This function is used for $\ell(\cdot)$ in equation (11).

While this scheme may well result in some improvement in word recognition with boosting, while avoiding relying on potentially erroneous phonetic labels, there is some information that is not utilised. Knowledge of the required sequence of words, together with the obtained sequence of words for each decoded sentence results in a set of errors that are fairly localised in time. The following sections discuss how it is possible to use a model that capitalises on such knowledge in order to define a distribution of errors over time.

6.1.2 Error expectation for boosting

In traditional supervised settings we are provided with a set of examples and labels, which constitute our training set, and thus it is possible to apply algorithms such as boosting. However this becomes problematic when labels are noisy [see for example 31]. Such an example is a typical speech recognition data set. Most of the time such a data set is composed of a set of sentences, with a

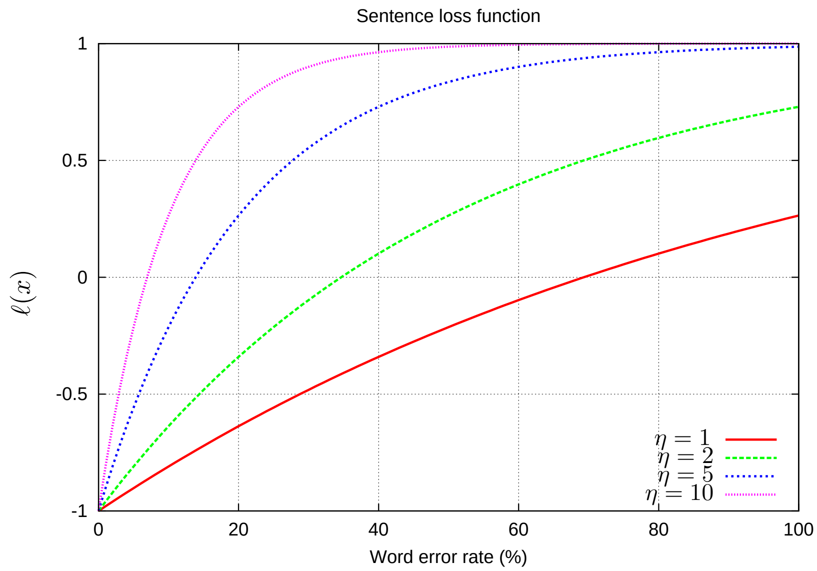


Figure 9: The sentence loss function (14) for $\eta \in \{1, 2, 5, 10\}$.

corresponding set of transcriptions. However, while the transcriptions may be accurate as far as the intention of the speakers or the hearing of the transcriber is concerned, subsequent translation of the transcription into phonetic labels is bound to be error prone, as it is quite possible for either the speaker to mispronounce words, or for the model that performs the automatic segmentation to make mistakes. In such a situation, adapting a model so that it minimises the errors made on the segmented transcriptions might not automatically lead into a model that minimises the word error rate, which is the real goal of a speech recognition system.

For this purpose, the concept of error expectation is introduced. Thus, rather than declaring with absolute certainty that an example is incorrect or not, we simply define $\ell(d_i) = \mathbf{P}(y_i \neq h_i)$, so that the sample loss is now the probability that a mistake was made on example i and we consider y_i to be a random variable. Since boosting can admit any sample loss function[16], this is perfectly reasonable and it's possible to use this loss as a sample loss in a boosting context. The following section discusses some cases for the distribution of y in the following section which are of relevance to the problem of speech recognition.

6.1.3 Error distributions in sequential decision making

In sequential decision making problems the knowledge about the correctness of decisions is delayed. Furthermore, it frequently lacks detailed information concerning the temporal location of errors. A common such case is knowing that we have made one or more errors in the time interval $[1, T]$. This form occurs in a number of settings. In the setting of individual sentence recognition a sequence of decisions is made which corresponds to an inferred utterance.

When this is incorrect, there is little information to indicate where mistakes were made.

In such cases it is necessary to define a model[4] for the probability of having made an erroneous decision at different points in times t, given that there has been at least one error in the interval $[1, T]$. Let us denote the probability of having made an error at time $t \in [1, T]$ by $\mathbf{P}(y_t \neq h_t \mid y_{1:T} \neq h_{1:T})$. A trivial example of such a model is to assume that the error probability is uniformly distributed. This can be expressed via the flat prior:

$$\mathbf{P}(y_t \neq h_t \mid y_{1:T} \neq h_{1:T}) \propto 1/T. \tag{15}$$

Another useful model is to assume an exponential prior such that:

$$\mathbf{P}(y_t \neq h_t \mid y_{1:T} \neq h_{1:T}) \propto \lambda^{t-T}, \quad \lambda \in [0, 1), \tag{16}$$

such that the expectation of an error near the end of the decision sequence is much higher. This is useful in tasks where it is expected that the decision error will be temporally close to the information that an error has been made. Ultimately, such models incorporate very little knowledge about the task, apart from this simple temporal structure.

In this case we focus on the application of speech recognition, which has some special characteristics that can be used to more accurately estimate possible locations of errors. For the case of labelled sentence examples it is possible to have a procedure that can infer the location of an error in time. This is because correctly recognised words offer an indication of where possible errors lie. Assume some procedure that creates an indicator function I_t such that $I_t = 1$ for instances in time where an error could have been made. We can then estimate the probability of having an error at time t as follows:

$$\mathbf{P}(y_t \neq h_t \mid y_{1:T} \neq h_{1:T}) = \frac{\gamma^{I_t}}{\sum_{k=1}^{T} \gamma^{I_k}}, \tag{17}$$

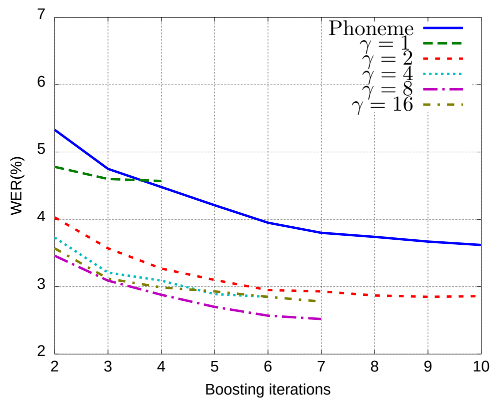
where the parameter $\gamma \in [1, \infty)$ expresses our confidence in the accuracy of I_t. A value of 1 will cause the probability of an error to be the same for all moments in time, irrespective of the value of I_t, while when γ approaches infinity we have absolute confidence in the inferred locations. Similar relations can be defined for an exponential prior and they can be obtained through the convolution of (16) and (17).

In order to apply boosting to temporal data, where classification decisions are made at the end of each sequence, we use a set of weights $\{\psi_t^{(i)}\}_i$ corresponding to the set of frames in an example sentence. At each boosting iteration j the weights are adjusted through the use of (17), resulting in the following recursive relation:

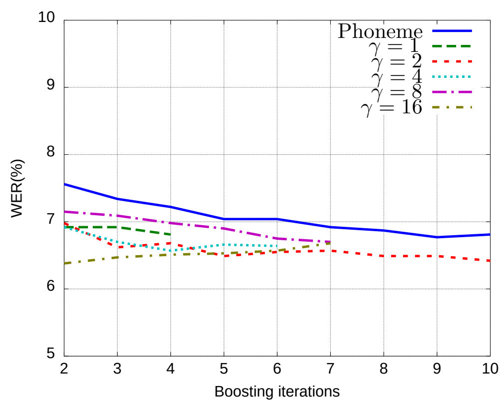
$$\psi_t^{(j+1)} = \frac{\psi_t^{(j)} \gamma^{I_t}}{\sum_{k=1}^{T} \psi_k^{(j)} \gamma^{I_k}} \tag{18}$$

In this manner, the loss incurred by the whole sentence is distributed to its constituent frames, although the choice is rather ad-hoc. A different approach was investigated by Zhang and Rudnicky [41], where the loss on the frames was related to the probability of the relevant word being uttered at time t, but their results do not indicate that this is a better choice compared to the simpler utterance-level training scheme that they also propose in that paper.

[4] Even if no model is explicitly defined, there is always one implied.



(a) Training



(b) Holdout

Figure 10: Word error rates for various values of γ, compared with the phoneme boosting approach, for the training and the holdout set.

6.2 Experiments with expectation boosting

We experimented on the OGI Numbers 95 (N95) data set [8] (details about the setup and dataset are given in Section 4). The experiment was performed as follows: firstly, a set of HMMs μ_0, composed of one model per phoneme, was trained using the available phonetic labels. This has the role of a starting point for the subsequent expert models. At each boosting iteration t we take the following steps: firstly, we sample with replacement from the distribution of training sentences given by the Ada-Boost algorithm. We create a new expert μ_t, initialised with the parameters of μ_0. The expert is trained on the sentence data using EM with the Viterbi approximation in the expectation step to calculate the expectation. The frames of each sequence carry an importance weight ψ_t, computed via (18), which is factored into the training algorithm by incorporating it in the posterior probability of the model h given the data x and time t, which, if we assume independence of x and t, can be written as:

$$p(h \mid x, t) \propto p(h \mid x) p(h \mid t) p(h).$$

In this case, $p(h \mid t)$ will correspond to our importance weight ψ_t.

After training, all sequences are decoded with the new expert. The weights of each sentence is increased according to (14), with $\eta = 10$. This value was chosen so that any sentence decodings with more than 50% error rate would be considered nearly completely erroneous (see Figure 9). For each erroneously decoded sentence we calculate the edit distance using a shortest path algorithm. All frames for which the inferred state belonged to one of the words that corresponded to a substitution, insertion, or deletion are then marked. The weights of marked frames are adjusted according to (17). The parameter γ corresponds to how smooth we want the temporal credit assignment to be.

In order to evaluate the combined models we use the multi-stream method described in equation (6), where the weight w_i of each stream i is $w_i \triangleq \pi(h_i) = \beta_i / \sum_j \beta_j$.

Experimental results comparing the performance of the above techniques to that of an HMM using segmentation information for training are shown in Figure 10a, for the training data and figure 10b for the test data. The figures include results for our previous results with boosting at the phoneme level. We have included results for values of $\gamma \in \{1, 2, 4, 8, 16\}$. Although we do not improve significantly upon our previous work with respect to the generalisation error, we found that on the training set, while boosting with pre-segmented phoneme examples had previously resulted in a reduction of the error to 3% after approximately 30 iterations (not shown), the sentence example training, combined with the error probability distribution over frames, converged to the same error after approximately 6 iterations. The situation was similar in the hold out set, with the new approach converging to a good generalisation error at 10 iterations, while the previous approach required 16 iterations to reach the same performance. A drawback of the new approach, however, is the need to specify two new hyper-parameters: (a) the shape of the cost function and (b) the shape of the expected error distribution. As mentioned previously, for (a) we are using (14) with $\eta = 10$ and for (b) we have chosen a mix between a uniform distribution and an indicator function, with γ being a free parameter. Choosing γ is not trivial (i.e. it cannot be chosen in the training set), since large values can lead to overfitting, while values that are too small seem to provide no benefit. For the experiments described in Section 7 we will be holding out part of the training set in order to select an appropriate value for γ.

The main interesting feature of the utterance-level approach is that we are minimising the word error rate directly, which is the real objective. Secondly, we don't require segmentation information during training. Lastly, the temporal probability distribution, derived from the word errors and the state inference, provides us with a method to assign weights to parts of the decoded sequence. Its importance becomes obvious when we compare the performance of the method for various values of γ. When the distribution is flat (i.e. when $\gamma = 1$), the performance of the model drops significantly. This supports the idea of using a probabilistic model for the errors over training sentences.

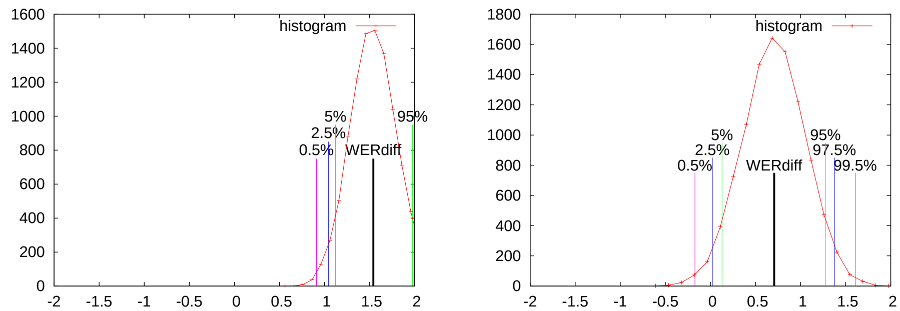
7 Generalisation performance comparison

In a real-world application one would have to use the training set for selecting hyper-parameters to use in unknown data. To perform such a comparison between methods, the training data set was split in two parts; holding out 1/3rd of it for validation. For each training method, we selected the hyperparameters θ having the best performance on the hold-out set. In our case, the hyperparameters are a tuple $\theta = (k, n, \gamma)$, where $k \in \{10, 20, 30, 40, 50\}$ is the number of Gaussians in the Gaussian mixture model, $n \in \{1, 2, 3, \ldots, 16\}$ is the number of experts and $\gamma \in \{1, 2, 4, 8, 16\}$ is the temporal credit assignment coefficient for the expectation-boosting method. The number of states was fixed to 3, since our exploratory experiment, described in Section 5, indicated that it was the optimal value by a large margin. For each method, the hyperparameters θ which provided the best performance in the hold-out set were used to train the model on the full training set. We then evaluated the resulting model on the independent test set. Full details on the data and methods setup are given in Sec. 4.

Model	Gaussians	Word error rate (%)
GMM	30	8.31
GMM embed	40	8.12
Boost GMM	10×30	7.41
HMM	10	7.52
HMM embed	10	7.04
Boost HMM	10×10	6.81
E-Boost HMM	7×10 (γ=8)	6.75
Bag HMM	16×20	5.97

Table 1: Test set performance comparison of models selected on a validation set. The second column indicates the number of Gaussians per phoneme. For ensemble methods, $n \times m$ denotes n models, each having m Gaussian components per state. *GMM* indicates a model consisting of a single Gaussian mixture for each phoneme. *HMM* indicates a model consisting of three Gaussian mixtures per phoneme. Thus for HMMs, the total number of Gaussians is three times that of the GMMs with an equal number of components per state. *Boost* and *Bag* models indicate models trained using the standard boosting and bagging algorithm respectively on the phoneme classification task, while *E-boost* indicates the expectation boosting algorithm for word error rate minimisation. Finally *embed* indicates that embedded training was performed subsequently to initialisation of the model.

We compared the following approaches. Firstly, a Gaussian mixture model (GMM), where the same observation distribution was used for all three states of the underlying phoneme hidden Markov model. This model was trained on the segmented data only. Secondly, a standard hidden Markov model (HMM) with three states per phoneme, also trained on the segmented data only. We also considered the same models trained on complete utterances, using *embedded training* after initialisation on the segmented data. The question we wanted to address, was whether (and which) ensemble methods could improve upon these baseline results in an unbiased setting. We first considered ensemble methods trained using segmented data, using the phoneme-level bagging and boosting described in Sec. 5. This included both bagging and boosting of HMMs, as well as boosting of GMMs, for completeness. In all cases the experimental setup was identical, and the only difference between the boosting and the bagging algorithms was that bagging used a uniform distribution for each bootstrap sample of the data and uniform weights on the expert models. Finally, at the utterance level, we used expectation boosting, which is described in Sec. 6.



(a) Bag vs HMM

(b) Boost vs HMM

Figure 11: Significance levels of word error rate difference between the phoneme-level bagging and boosting and HMM. he histograms are created from 10,000 bootstrap samples of the test data, as described in Section 4.2. *WERdiff* displays the average difference in word error rate performance between the two methods. Positive values indicate that the first method has lower error than the second. The percentage values indicate the tail mass of the histogram to that point.

Table 1 summarises the results obtained, indicating the number of Gaussians per phoneme and the word error rate obtained for each model. If one considers only those models that were created strictly using the classification task, that is without adapting word models, ensemble methods perform significantly better. Against the baseline *HMM embed* model, which is trained on full utterances, however, not all ensemble methods perform so well.

This can be seen clearly in Figures 11,12 (p. 24,26) which show the probability[5] that one model will be better than the other by a given amount. In particular, the estimated probability that *Boost* is better than *HMM embed*, shown in Figure 12a is merely 51% and the mean difference in performance is just 0.23%, while against the simple *HMM* the result, shown in Figure 11a, is statistically significant with a confidence of 91%. Slightly better performance is offered by *E-Boost*, with significance with respect to the *HMM* and *HMM embed* models at 98% and 65% respectively. Overall bagging works best, performing better than other methods with a confidence of at least 99% in all cases, while approximately 97.5% of the probability mass lies above the 0.5% differential word error rate compared to the baseline model, as can be seen in Figure 12a.

However these results are not quite near the state of the art on this database. Other researchers [1, 13, 18, 21, 22, for example], have achieved word error rates $5.0 \pm 0.3\%$, mainly through the use of different phonetic models. Accordingly, some further experiments were performed with Markov models using a more complex phonetic model (composed of 80 tri-phones, i.e. phonemes with contextual information). After performing the same model selection procedure as

[5]Calculated via the bootstrap estimate, detailed in Sec. 4.2.

above, a single such model achieved word error rates of $4.8 \pm 0.1\%$ (not shown in the table) which is in agreement with published state of the art results. This suggests that using a more complex model could be better than using mixtures of simpler models. Corresponding results for ensembles of tri-phone models indicated that the boosting-based approaches could not increase generalisation performance, achieving a word error rate of 5.1%. However, the simpler bagging approach managed to reach a performance of 4.5%. However, the performance differences are not really significant in this case.

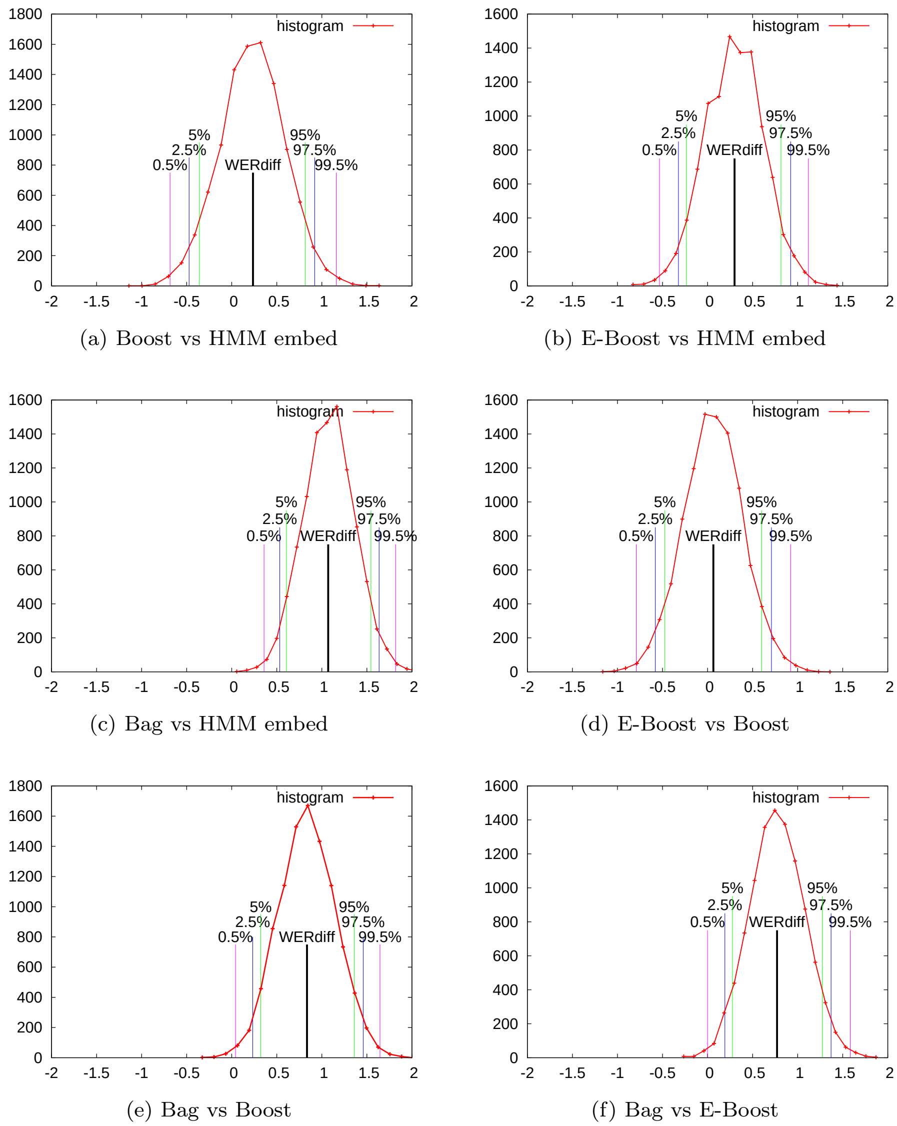
Nevertheless, it appears that in all cases, phoneme bagging is the most robust approach. The reasons for this are not apparent, but it is tempting to conclude that the label noise combined with the variance-reducing properties of bagging are at least partially responsible. Although it should be kept in mind that the aforementioned tri-phone results are limited in significance due to the small difference in performance between methods, they nevertheless indicate that in certain situations ensemble methods and especially bagging may be of some use to the speech recognition community.

8 Discussion

We presented some techniques for the application of ensemble methods to HMMs. The ensemble training was performed for complete HMMs at either the phoneme or the utterance level, rather than at the frame level. Using boosting techniques at the utterance level was thought to lead to a method for reducing the word error rate. Interestingly, this word error rate reduction scheme did not improve generalisation performance for boosting, while the simplest approach of all, bagging, performed the best.

There are a number of probable causes. The first one is that the amount of data is sufficiently large for ensemble techniques to have little impact on performance, i.e. there is enough data to train sufficiently good base models. The second is that the state-locked multi-stream decoding techniques were investigated for model recombination lead to an increase in generalisation error as the inference performed is very approximate. The third is that the boosting approach used is simply inappropriate. The first case mustn't be true, since bagging does achieve considerable improvements over the other methods. There is some evidence for the second case, since the GMM ensembles are the only ones that should not be affected by the multi-stream approximations and while a more substantial performance difference can be observed, it nevertheless is not much greater. The fact that bagging's phoneme mixture components are all trained on samples from the same distribution of data and that it outperforms boosting is also in agreement with this hypothesis. This leaves the possibility that the type of boosting training used is inappropriate, at least in conjunction with the decoding method used, open.

Future research in this direction might include the use of other approximations for decoding than constrained multi-stream methods. Such an approach was investigated by Meyer and Schramm [26], where the authors additionally consider the harder problem of large vocabulary speech recognition (for which even inferring the most probable sequence of states in a single model may be computationally prohibitive). It could thus be also possible to use the methods developed herein for large vocabulary problems by borrowing some of their



(a) Boost vs HMM embed

(b) E-Boost vs HMM embed

(c) Bag vs HMM embed

(d) E-Boost vs Boost

(e) Bag vs Boost

(f) Bag vs E-Boost

Figure 12: Significance levels of word error rate difference between the top four models. The histograms are created from 10,000 bootstrap samples of the test data, as described in Section 4.2. *WERdiff* displays the average difference in word error rate performance between the two methods. Positive values indicate that the first method has lower error than the second. The percentage values indicate the tail mass of the histogram to that point.

techniques. The first technique, also used by [39], relies on finding an n-best list of possible utterances, assuming there are no other possible utterances and then fully estimating the posterior probability of the n alternatives. The second technique, developed by Schramm and Aubert [33], combines multiple pronunciation models. In this case each model arising from boosting could be used in lieu of different pronunciation models. Another possible future direction is to consider different algorithms. Both Ada-Boost.M1, which was employed here, and Ada-Boost.M2, are using greedy optimisation for the mixture coefficients. Perhaps better optimisation procedures, such as those proposed by Mason et al. [24], may result in an additional advantage.

References

[1] Athineos, M., Hermansky, H., and Ellis, D. P. (2004). LP-TRAP: Linear predictive temporal patterns. In *Proceedings of the 8th International Conference on Spoken Language Processing (ICSLP)*, pages 949–952.

[2] Bahl, L., Brown, P., de Souza, P., and Mercer, R. (1988). A new algorithm for the estimation of hidden Markov model parameters. In *Proceedings of the IEEE Inernational Conference on Acoustics, Speech and Signal Processig (ICASSP)*, pages 493–496.

[3] Bisani, M. and Ney, H. (2004). Bootstrap estimates for confidence intervals in ASR performance evaluation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 409–412.

[4] Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford.

[5] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

[6] Breslin, C. (2008). *Generation and Combination of Complementary Systems for Automatic Speech Recognition*. PhD thesis, Cambridge University Endingeering Department and Darwin College.

[7] Breslin, C. and Gales, M. (2009). Directed decision trees for generating complementary systems. *Speech Communication*, 51(3):284–295.

[8] Cole, R. A., Roginski, K., and Fanty, M. (1995). The OGI numbers database. Technical report, Oregon Graduate Institute.

[9] Cook, G. and Robinson, A. (1996). Boosting the performance of connectionist large vocabulary speech recognition. In *Proceedings of the IEEE International Conference on Acoustic, Speech and Signal Porocessing (ICASSP)*, volume 3, pages 1305–1308, Philadelphia, PA.

[10] Dimitrakakis, C. (2006). *Ensembles for Sequence Learning*. PhD thesis, École Polytechnique Fédérale de Lausanne.

[11] Dimitrakakis, C. and Bengio, S. (2004). Boosting HMMs with an application to speech recognition. In *Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing (ICASSP)*, volume 5, pages 621–624.

[12] Dimitrakakis, C. and Bengio, S. (2005). Boosting word error rates. In *Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing (ICASSP)*, volume 5, pages 501–504.

[13] Doss, M. M. (2005). *Using Auxiliary Sources of Knowledge for Automatic Speech Recognition*. PhD thesis, École Polytechnique Fédérale de Lausanne, Computer Science Department, Lausanne, Switzerland. Thesis No. 3263.

[14] Efron, B. and Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*, volume 57 of *Monographs on Statistics & Applied Probability*. Chapmann & Hall.

[15] Fiscus, J. (1997). 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.

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

[17] Gales, M. and Young, S. (2007). The application of hidden Markov models in speech recognition. *Foundations and Trends® in Signal Processing*, 1(3):195–304.

[18] Hermansky, H. and Sharma, S. (1998). TRAPs - classifiers of temporal patterns. In *Proceedings of the 5th International Conference on Speech and Language Processing (ICSLP)*, pages 1003–1006.

[19] Johnny Mariéthoz and Samy Bengio (2004). A new speech recognition baseline system for Numbers 95 version 1.3 based on Torch. IDIAP-RR 04-16, IDIAP.

[20] Ketabdar, H., Bourlard, H., and Bengio, S. (2005a). Hierarchical multi-stream posterior based speech recognition system. IDIAP-RR 25, IDIAP.

[21] Ketabdar, H., Vepa, J., Bengio, S., and Bourlard, H. (2005b). Developing and enhancing posterior based speech recognition systems. In *Proceedings of the 6th annual Conference of the International Speech Communication Association (Interspeech)*, pages 1461–1464, Lisbon, Portugal.

[22] Lathoud, G., Magimai.-Doss, M., Mesot, B., and Bourlard, H. (2005). Unsupervised Spectral Subtraction for Noise-Robust ASR. In *Proceedings of the 2005 IEEE ASRU Workshop*, pages 343–348, San Juan, Puerto Rico.

[23] Man-Hung, S., Yang, X., and Gish, H. (2009). Discriminatively trained GMMs for language classification using boosting methods. *IEEE Transactions on Audio, Speech and Language Processing*, 17(1):187–197.

[24] Mason, L., Bartlett, P. L., and Baxter, J. (2000). Improved generalization through explicit optimization of margins. *Machine Learning*, 38(3):243–255.

[25] Meir, R. and Rätch, G. (2003). An introduction to boosting and leveraging. In *Advanced Lectures on Machine Learning*, volume 2600 of *LNCS*, pages 119–184. Springer.

[26] Meyer, C. and Schramm, H. (2006). Boosting HMM acoustic models in large vocabulary speech recognition. *Speech Communication*, 48:532–548.

[27] Misra, H. and Bourlard, H. (2005). Spectral Entropy Feature in Full-Combination Multi-Stream for Robust ASR. In *Proceedings of the the 9th European Conference on Speech Communication and Technology (Eurospeech)*, pages 2633–2636, Lisbon, Portugal.

[28] Misra, H., Bourlard, H., and Tyagi, V. (2003). New entropy based combination rules in HMM/ANN multi-stream ASR. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 741–744, Hong Kong.

[29] Morris, A., Hagen, A., Glotin, H., and Bourlard, H. (2001). Multi-stream adaptive evidence combination for noise robust ASR. *Speech Communication*, 34:25–40.

[30] Rabiner, L. R. and Juang, B.-H. (1993). *Fundamentals of Speech Recognition.* PTR Prentice-Hall, Inc.

[31] Rätsch, G., Onoda, T., and Müler, K.-R. (2001). Soft margins for adaboost. *Machine Learning*, 42(3):287–320.

[32] Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336.

[33] Schramm, H. and Aubert, X. L. (2006). Efficient integration of multiple pronunciations in a large vocabulary decoder. In *Proceedings of the IEEE International Conference on Acoustic, Speech and Signal Porocessing (ICASSP)*, volume 3, pages 143–146.

[34] Schwenk, H. (1999). Using boosting to improve a hybrid HMM/neural network speech recogniser. In *Proceedings of the IEEE International Conference on Acoustic, Speech and Signal Porocessing (ICASSP)*, pages 1009–1012.

[35] Siohan, O., Ramabhadran, B., and Kingsbury, B. (2005). Constructing ensembles of ASR systems using randomized decision trees. In *Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing (ICASSP)*, volume 1, pages 197–200.

[36] Van der Vaart, A. and Wellner, J. (1996). *Weak convergence and empirical processes: with applications to statistics*. Springer.

[37] Yang, X., Siu, M.-h., Gish, H., and Mak, B. (2007). Boosting with anti-models for automatic language identification. In *Proceedings of the 8th annual Conference of the International Speech Communication Association (Interspeech)*, pages 342–345.

[38] Zhang, R. (2007). *Making an Effective Use of Speech Data for Acoustic Modeling*. PhD thesis, Carnegie Mellon University.

[39] Zhang, R. and Rudnicky, A. I. (2003). Comparative study of boosting and non-boosting training for constructing ensembles of acoustic models. In *Proceedings of the 8th European Conference on Speech Communication and Technology (Eurospeech)*, pages 1885–1888.

[40] Zhang, R. and Rudnicky, A. I. (2004a). Apply n-best list re-ranking to acoustic model combinations of boosting training. In *Proceedings of the 8th International Conference on Spoken Language Processing (ICSLP)*, pages 1949–1952.

[41] Zhang, R. and Rudnicky, A. I. (2004b). A frame level boosting training scheme for acoustic modeling. In *Proceedings of the 8th International Conference on Spoken Language Processing (ICSLP)*, pages 417–420.

[42] Zweig, G. and Padmanabhan, M. (2000). Boosting Gaussian mixtures in an LVCSR system. In *Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing (ICASSP)*, pages 1527–1530.