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.

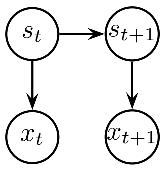

Figure 1: Graphical representation of a hidden Markov model, with arrows indicating dependencies between variables. The observations x_t and the next state s_{t+1} only depend on the current state s_t.

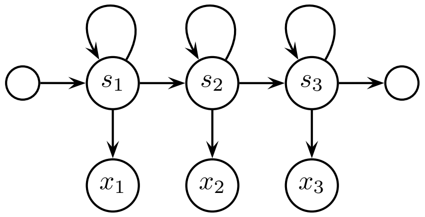

Figure 2: Graphical representation of a phoneme model with 3 emitting states, as well as initial and terminal non-emitting states. The arrows depict dependencies between specific states. All the phoneme models used in this paper employed the above topology.

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 $\mathcal{D}$:

$$\mu^* = \operatorname*{arg\,max}_{\mu \in \mathcal{M}} p(\mu|\mathcal{D}).$$

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.

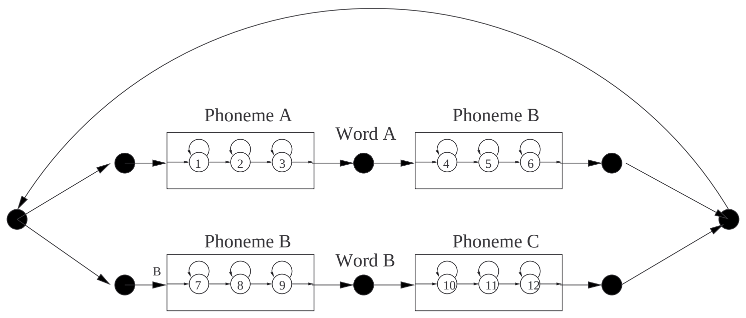

Figure 3: A hidden Markov model for speech recognition. The figure depicts how models of three phonemes, A, B, C are used to construct a single hidden Markov model for distinguishing between two different words. The states are indexed uniquely. Black circles indicate non-emitting states.

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.

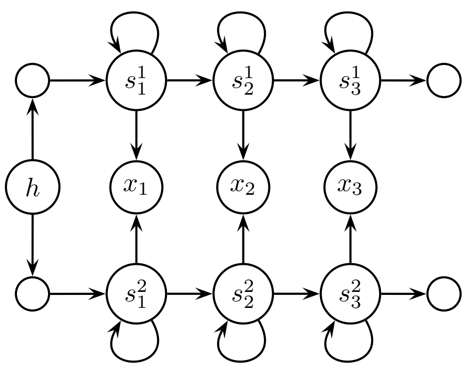

Figure 4: A phoneme mixture model. The generating model depends on the hidden variable h, which determines the mixing coefficients between model 1 and 2. The random variable h may in general depend on other variables. The distribution of the observation is a mixture between the two distributions predicted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

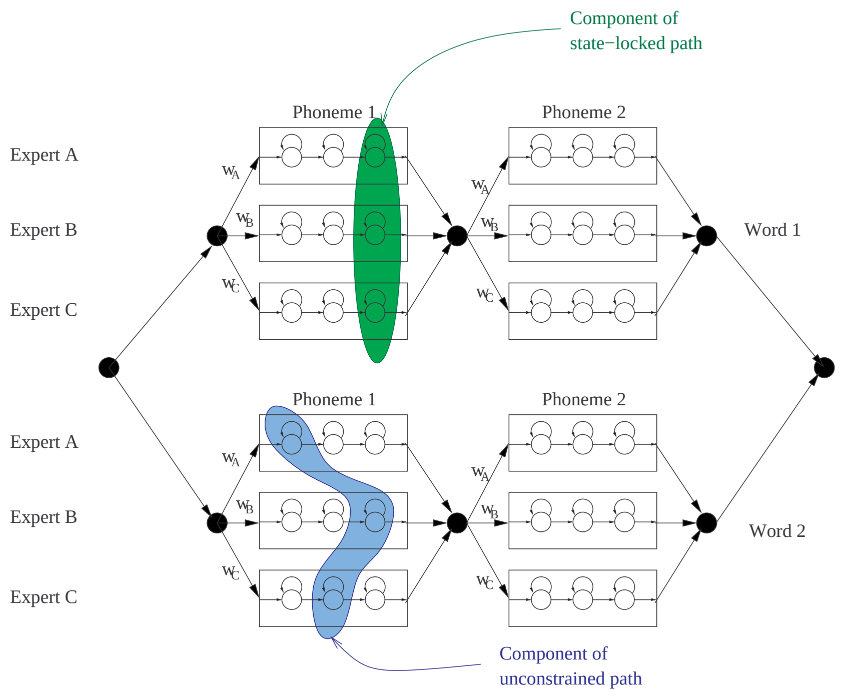

Figure 5: Single-path multi-stream decoding for two vocabulary words consisting of two phonemes each. When there is only one expert the decoding process is done normally. In the multiple expert case, phoneme models from each expert are connected in parallel. The transition probabilities leading from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