Minimum error classification training of HMMs— Implementation details and experimental results

David Rainton and Shigeki Sagayama

*ATR Interpreting Telephony Research Laboratories,
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-02 Japan*

(Received 21 February 1992)

This paper describes an implementation of minimum error training for continuous gaussian mixture density HMMs. Instead of maximising the conditional probability of producing a set of training data, as in the conventional HMM maximum likelihood approach, we train to minimise the number of recognition errors. The most important aspect of this work is the use of a first order differentiable “loss” function, the minimisation of which is directly related to the minimisation of the recognition error rate. The performance of the resulting minimum error HMMs was compared against that of conventional maximum likelihood HMMs in a continuous speech recognition task using the ATR 5,240 Japanese word data base. The results were impressive. For example, for 10 mixture 5 state Baum-Welch trained HMMs, after minimum error training word error rates reduced from 20.6% to 3.0% on the closed training set and 23.2% to 13.2% on the open test set. Furthermore, 3 mixture minimum error HMMs performed better than 10 mixture maximum likelihood HMMs. In fact in every performance measure made the minimum error HMMs proved to be superior.

Keywords: Speech recognition, Hidden Markov modelling, Discriminant training, Minimum error classification

PACS number: 43.72. Ne

1. INTRODUCTION

This paper describes minimum error (ME) classification training of hidden Markov models (HMMs), followed by an experimental comparison with conventional maximum likelihood models.

Although the application of minimum error classification to speech recognition is relatively new, several different implementations have already been described in the recent literature.^{1,2)} In both cited references, the authors derive functional approximations to the empirical error rate (hereafter referred to as “loss functions”), which are then minimised using gradient descent based techniques. The main practical difference between the two papers lies in the choice of loss functions, both of which are specific to an HMM classifier structure.

Recently a more general definition of loss has

been introduced within the framework of an adaptive discriminant learning paradigm, known as Generalized Probabilistic Descent (GPD).³⁻⁶⁾ It is this definition of loss that is used in this paper, where we show how to simplify and apply it within an HMM framework.

To avoid any possible confusion between our work and that described in Refs. 3-6), we should point out that the name “GPD” has been used loosely by the above authors to cover both the definition of a smooth continuous loss function and an adaptive learning algorithm. In this paper, while we use the loss function defined in Refs. 3-6) we do not rely on the convergence proofs provided for the adaptive learning algorithm. Instead we assume the availability of all the training data, thus we are free to minimise the loss using any of the many well known multidimension non-linear mini-

minimisation algorithms.⁸⁾

2. MINIMUM ERROR CLASSIFICATION

2.1 An Introduction

Briefly, minimum error classification training attempts to minimise a function of the recognition errors on a closed data set, rather than, as in the case of conventional Baum-Welch maximum likelihood training, attempting to maximise the conditional probability of generating the data set. Importantly, the two are not equivalent, since maximum likelihood training is not guaranteed to minimise the recognition error rate; modelling and estimation errors ensure that it is only an indirect means of doing so.

Given that the most important performance measure for any speech recognition system is the number of recognition errors that it makes, the motivation for minimum error training is obvious.

The most important aspect of minimum error classification is the definition of a suitable objective function, the minimisation of which is directly related to the minimisation of the empirical error rate. Once the loss has been defined, then assuming the availability of all the training data, we have a straightforward non-linear multidimensional minimisation problem, *i.e.* we have an unknown non-linear objective function (the loss) which depends on a number of independent variables (the classifier parameters).⁷⁾ The problem is to find those values of the classifier parameters which minimise the loss.

Given any computable loss function (*i.e.* it need not necessarily be continuous) we could minimise it using the so called "annealing" methods. Thus for example, we could trivially define the loss as equal to the number of recognition errors made on the closed training set. The recogniser parameters could then be adjusted using simulated annealing to minimise this quantity. Such a definition of loss however has the disadvantage of being discontinuous, thus we cannot compute the first order derivative of the loss with respect to the classifier parameters. This in turn prevents us from using any of the usually more powerful minimisation algorithms, such as steepest descent and conjugate gradients, which require computation, not only of the loss function itself, but also its first order derivative. As stated previously, for the work presented in this paper we use a loss function defined in Ref. 3). The beauty of the chosen definition is that it is a smooth

continuous function, whose first order derivative is easily computed for several common classifier structures, including the continuous mixture density HMM. Thus given this definition of a continuous loss function and availability of all the training data, we can apply any of the standard minimisation techniques to locally minimise the loss. Note that there is nothing probabilistic about this procedure; assuming that we use an appropriate minimisation technique, such as steepest descent or conjugate gradients, then the loss is guaranteed to reduce at each learning iteration. However, in order to avoid possible confusion later on, we should point out that in fact with the very simple gradient descent algorithm used in the implementation presented here, the loss did not always decrease with each iteration. That however was due simply to the poor nature of the descent algorithm, and can not be given a probabilistic interpretation.

The aim of this paper is to show the interested reader exactly how to implement a minimum error classification algorithm. Thus we provide, wherever possible, explicit formulations and specific implementation details. For the theory, and detailed arguments justifying minimum error classification itself, we refer the reader to the appropriate references.^{3,4)}

The starting point for this paper is the definition of a very general loss function. Given this definition of loss, we then show, in a step by step fashion, exactly how to simplify and apply it within an HMM framework, specifically a continuous gaussian mixture density HMM framework. The hope being that with such an approach alternative implementations should become readily apparent to the reader.

2.2 At What Level Should We Minimise the Recognition Errors?

There exists in speech recognition a clear hierarchy of recognition tasks. The most common is that which places phonemes at the bottom, isolated words in the middle and then sentences at the top. As we move up the hierarchy the number of classes at each level increases, in the worst case exponentially. In practice however, the rate of increase is usually constrained by the use of a grammar, mapping only certain class sequences at one level in the hierarchy to the next.

Because the number of classes always increases

as we move up the hierarchy, we usually decide to implement the basic classifier structure at the lowest possible level. Thus in the case of an HMM speech recogniser, we usually use phoneme HMMS. However, simply because we define our classifier structure at one level does not necessarily mean that it is best to optimise it at that same level. The level we optimise at should be defined by the recognition task, not the classifier structure, since errors made at lower levels may not be observable at a higher task level. For example, in an isolated word recognition task, using phoneme HMMS and an associated grammar, it is the word recognition error rate we should train to minimise, not the phoneme recognition rate. It is important to realise that the two are not equivalent.

In this paper we refer to two different class types, the "HMM class," whose level is defined by the choice of classifier structure, and the "token class," whose level is defined by the recognition task. The distinction between the two is important since optimisation is performed at the token class level, not the HMM class level. Thus we have both the advantage of a relatively small classifier (*e.g.* 26 phoneme HMMS), while directly minimising only those errors that we are interested in (*i.e.* word recognition errors).

2.3 The Token and HMM Classes

First we define the "HMM class." Each HMM class is a member of the set λ , *i.e.*

$$\lambda = \{\lambda_1, \dots, \lambda_i, \dots, \lambda_I\} \quad (1)$$

where I is the number of HMMS. For example, in this paper we use 26 phoneme HMMS, thus $I=26$. Each λ_i is a vector of HMM parameters completely characterising the i 'th HMM model. In the case of continuous gaussian mixture density HMMS, this vector contains four distinct types of parameters. One parameter type is

$$a_{i,p,q} \quad 1 \leq p, q \leq S \quad (2)$$

where $a_{i,p,q}$ is the state transition probability from state p to state q in the i 'th HMM. The total number of states is S . The gaussian mixtures are characterised by the remaining three parameter types. First the mixture weights

$$w_{i,s,m} \quad \begin{matrix} 2 \leq s \leq S-1 \\ 1 \leq m \leq M \end{matrix} \quad (3)$$

where $w_{i,s,m}$ is the mixture weight for the m 'th

mixture in the s 'th state of the i 'th HMM. The number of mixtures per state is M . It is assumed that the first and last model states are non-emitting (*i.e.* do not have associated gaussian probability density functions), hence s ranges only between 2 and $S-1$, instead of 1 and S .

The gaussian means are characterised by the vectors $\mu_{i,s,m}$, each mean vector containing D components

$$\begin{matrix} 2 \leq s \leq S-1 \\ \mu_{i,s,m,d} \quad 1 \leq m \leq M \\ 1 \leq d \leq D \end{matrix} \quad (4)$$

where $\mu_{i,s,m,d}$ is the d 'th element in the mean vector corresponding to the m 'th gaussian in the s 'th state of the i 'th model. D is equal to the observation vector size. Finally the gaussian variances are characterised by the vectors $\sigma_{i,s,m}^2$, whose components are

$$\begin{matrix} 2 \leq s \leq S-1 \\ \sigma_{i,s,m,d}^2 \quad 1 \leq m \leq M \\ 1 \leq d \leq D \end{matrix} \quad (5)$$

This assumes diagonal covariance HMMS, however extension to the full covariance case is trivial.

As was stated in the previous subsection, the HMM classes can be defined at any level of the recognition hierarchy, although it is usual to restrict them to either the phoneme or word level. Thus usually each individual λ_i represents either a phoneme or word HMM.

The token, or training data set $\mathcal{O}$, is defined as being comprised of K different "token classes," *i.e.*

$$\mathcal{O} = \{\mathcal{O}_1, \dots, \mathcal{O}_k, \dots, \mathcal{O}_K\} \quad (6)$$

where $\mathcal{O}_k$ is the sequence of training tokens from the k 'th token class. If there are N_k tokens per token class then the k 'th token class is represented by the set

$$\mathcal{O}_k = \{\mathcal{O}_{k,1}, \dots, \mathcal{O}_{k,n}, \dots, \mathcal{O}_{k,N_k}\} \quad (7)$$

Thus for example in the case of the isolated word recogniser described in this paper, each token class is a single word of which there is only one realisation (*i.e.* $N_k=1$ for all k).

$\mathcal{O}_{k,n}$ is a sequence of observation vectors produced by the n 'th token in the k 'th class, *i.e.*

$$\mathcal{O}_{k,n} = \mathbf{o}_{k,n,1}, \dots, \mathbf{o}_{k,n,t}, \dots, \mathbf{o}_{k,n,T_{k,n}} \quad (8)$$

where $T_{k,n}$ is the number of observations in the token.

Finally, $\mathbf{o}_{k,n,t}$ is the D dimensional observation

vector at time t , in the n 'th training token, from the k 'th class, *i.e.*

$$\mathbf{o}_{k,n,t} = [o_{k,n,t,1}, \dots, o_{k,n,t,d}, \dots, o_{k,n,t,D}]. \quad (9)$$

Again, in the case of the recogniser described in this paper $\mathbf{o}_{k,n,t}$ are 26 dimensional vectors, comprising of 12 cepstral coefficients, 12 delta cepstral coefficients, a power and a delta power coefficient.

The mapping between HMM class sequences and the token classes is defined by a grammar $\mathcal{G}(\cdot)$, *i.e.*

$$\mathcal{G}(\mathcal{C}_k) \mapsto \Lambda_k \quad (10)$$

where Λ_k is the "composite HMM" corresponding to the k 'th token class. Each composite HMM Λ_k is a network of individual HMMs from λ , representing all possible parses for the k 'th token class. Thus for the previously described isolated word recogniser, each composite HMM is a word HMM produced from a network of phoneme HMMs.

2.4 Definition of Loss

For minimum error classification we require an objective or loss function to minimise. The approach pursued in this paper requires that this loss function satisfy two important criterion. First, it must be a smooth differentiable function of the HMM parameters, with an easily computable first order derivative. This facilitates the use of gradient descent techniques in loss minimisation. Secondly, the loss must be directly related to the number of recognition errors that the classifier makes. Thus minimising the loss will be directly equivalent to minimising the number of recognition errors.

One loss function which satisfies both of the above criterion is the empirical loss $L(\mathcal{C}, \lambda)$ defined in GPD,²⁾ *i.e.*

$$L(\mathcal{C}, \lambda) = \sum_{k=1}^K \left\{ \sum_{n=1}^{N_k} l(d(\mathcal{C}_{k,n}, \lambda)) \right\} \quad (11)$$

where $l(\cdot)$ is the individual token loss for the n 'th token from the k 'th token class. This token loss is in turn defined in terms of the misclassification measure $d(\mathcal{C}_{k,n}, \lambda)$, *i.e.*

$$l(d(\mathcal{C}_{k,n}, \lambda)) = \frac{1}{1 + \exp(-\alpha[d(\mathcal{C}_{k,n}, \lambda) - \beta])}. \quad (12)$$

The real valued constants α and β control the slope and mid point of the sigmoid, and are determined experimentally. The misclassification measure $d(\mathcal{C}_{k,n}, \lambda)$ is a measure of the penalty incurred when correctly classifying the n 'th token in the k 'th token class, *i.e.*

$$d(\mathcal{C}_{k,n}, \lambda) = -g(\mathcal{C}_{k,n}, \Lambda_k) + \ln \left(\left[\frac{1}{K-1} \sum_{c, c \neq k} \exp(\eta g(\mathcal{C}_{k,n}, \Lambda_c)) \right]^{1/\eta} \right) \quad (13)$$

where $g(\mathcal{C}_{k,n}, \Lambda_c)$ is the "discriminant function"

$$g(\mathcal{C}_{k,n}, \Lambda_c) = \ln \left(\left[\sum_{\Theta_c} \{P(\mathcal{O}_{k,n}, \Theta_c | \Lambda_c)\}^\epsilon \right]^{1/\epsilon} \right). \quad (14)$$

Θ_c is any allowable model/state sequence for the Λ_c 'th composite HMM. The state sequence Θ_c defines both the allowable sequences of individual HMMs within the composite HMM, and the allowed state sequences within each individual HMM itself, *i.e.*

$$\Theta_c = \theta_{c,1}, \psi_{c,1}, \dots, \theta_{c,t}, \psi_{c,t}, \dots \quad (15)$$

where $\psi_{c,t}$ is the HMM class at time t and $\theta_{c,t}$ is the state within that HMM at time t .

The joint probability of the observation sequence $\mathcal{O}_{k,n}$ and the model/state sequence Θ_c is defined by the standard equation

$$P(\mathcal{O}_{k,n}, \Theta_c | \Lambda_c) = \prod_{t=1}^{T_{k,n}} a_{\psi_{c,t}, \theta_{c,t-1}, \theta_{c,t}} b_{\psi_{c,t}, \theta_{c,t}}(\mathbf{o}_{k,n,t}). \quad (16)$$

Note that if $\theta_{c,t-1}$ is the last state in the model $\psi_{c,t}$, then $a_{\psi_{c,t}, \theta_{c,t-1}, \theta_{c,t}}$ is a model, not a state transition probability (for the work here we assume that all model transition probabilities are some constant value). The probability of producing the observation vector $\mathbf{o}_{k,n,t}$, in the s 'th state of the i 'th HMM class is given by the equation

$$b_{i,s}(\mathbf{o}_{k,n,t}) = \sum_{m=1}^M w_{i,s,m} \mathcal{N}(\mathbf{o}_{k,n,t}, \boldsymbol{\mu}_{i,s,m}, \boldsymbol{\sigma}_{i,s,m}^2) \quad (17)$$

where $\mathcal{N}(\mathbf{o}_{k,n,t}, \boldsymbol{\mu}_{i,s,m}, \boldsymbol{\sigma}_{i,s,m}^2)$ denotes a multivariate Gaussian with mean $\boldsymbol{\mu}_{i,s,m}$ and variance vector $\boldsymbol{\sigma}_{i,s,m}^2$, *i.e.*

$$\mathcal{N}(\mathbf{o}_{k,n,t}, \boldsymbol{\mu}_{i,s,m}, \boldsymbol{\sigma}_{i,s,m}^2) = \prod_{d=1}^D \frac{1}{\sqrt{(2\pi)\sigma_{i,s,m,d}^2}} \exp \left(-\frac{(o_{k,n,t,d} - \mu_{i,s,m,d})^2}{2\sigma_{i,s,m,d}^2} \right). \quad (18)$$

With these equations we can directly express the loss in terms of the HMM parameters $w_{i,s,m}$, $\mu_{i,s,m,d}$ and $\sigma_{i,s,m,d}^2$.

2.5 Simplifying the Loss Definition

The definition of loss given in the previous subsection is a very general one, and prohibitively expensive to compute. In practice however we can

make several simplifications. Although the exact effect of these simplifications is unknown, it is reasonable to assume that they have a negative effect on the performance of the resulting classifier, the significance of which can only really be determined experimentally.

First, we consider a simplification of the discriminant function given in Eq. (14). Equation (14) can be regarded as a form of generalised likelihood, which can be reduced to two well known instances. When $\xi=1$ Eq. (14) reduces to the standard HMM full likelihood probability, generally computed using the forward-backwards algorithm. When $\xi \rightarrow \infty$ Eq. (14) reduces to the likelihood of the best path, usually computed using the Viterbi algorithm. It is this latter definition of the discriminant function that we shall use from now on, *i.e.*

$$g(\mathcal{C}_{k,n}, \Lambda_c) = \ln(P(\mathcal{C}_{k,n}, \Theta_c | \Lambda_c)) \quad (19)$$

where Θ_c is now redefined as the Viterbi model/state sequence for the c 'th composite HMM. Substituting for the log probability in Eq. (19) using Eq. (16) gives

$$g(\mathcal{C}_{k,n}, \Lambda_c) = \sum_{t=1}^{T_{k,n}} \ln(a_{\phi_{c,t}, \theta_{c,t-1}, \theta_{c,t}}) + \sum_{t=1}^{T_{k,n}} \ln(b_{\phi_{c,t}, \theta_{c,t}}(\mathbf{o}_{k,n,t})) \quad (20)$$

The next simplification we can make is in the definition of the misclassification measure in Eq. (13). In fact strictly speaking, we make both a modification and a simplification. The simplification is obtained by letting $\eta \rightarrow \infty$, so Eq. (13) reduces to

$$d(\mathcal{C}_{k,n}, \lambda) = -g(\mathcal{C}_{k,n}, \Lambda_k) + g(\mathcal{C}_{k,n}, \Lambda_c)$$

$$\text{where } c = \operatorname{argmax}_{p, p \neq k} g(\mathcal{C}_{k,n}, \Lambda_p) \quad (21)$$

The modification is that we drop the inequality $p \neq k$, thus the final misclassification measure used in this paper is given by the equation

$$d(\mathcal{C}_{k,n}, \lambda) = -g(\mathcal{C}_{k,n}, \Lambda_k) + g(\mathcal{C}_{k,n}, \Lambda_c)$$

$$\text{where } c = \operatorname{argmax}_p g(\mathcal{C}_{k,n}, \Lambda_p) \quad (22)$$

The effect of dropping the inequality is to ensure that the loss is zero when any token is correctly classified. This ensures that the HMMS are only updated using incorrectly classified tokens.

An alternative approach is to set η to an intermediate value. This will have the effect of allowing only the N Best tokens to dominate in Eq. (14), the exact value of N being determined by the choice of

η . While computationally slightly more expensive, such an approach may well prove more robust, as not only will the recogniser learn to correct recognition mistakes, but it will also attempt to improve the separability between correctly classified tokens. We are presently investigating this question.

2.6 Gradient Descent Minimisation of the Loss Function

To reduce the loss function during training we used a very simple gradient descent algorithm. It is not the "method of steepest descent," since the loss function is not guaranteed to reduce at each iteration. The only virtue of the algorithm described here was its implementational simplicity. In future it is intended that this algorithm will be replaced with a more powerful descent technique such as the conjugate gradient method.

The descent method used was an extremely simple iterative process. At each iteration the individual HMM parameters were adjusted according to the set of equations

$$w_{i,s,m}^{n+1} = w_{i,s,m}^n - \epsilon_w(n) \frac{\partial L(\mathcal{C}, \lambda)}{\partial w_{i,s,m}^n} \quad (23)$$

$$\mu_{i,s,m,d}^{n+1} = \mu_{i,s,m,d}^n - \epsilon_\mu(n) \frac{\partial L(\mathcal{C}, \lambda)}{\partial \mu_{i,s,m,d}^n} \quad (24)$$

$$\sigma_{i,s,m,d}^{2,n+1} = \sigma_{i,s,m,d}^{2,n} - \epsilon_\sigma(n) \frac{\partial L(\mathcal{C}, \lambda)}{\partial \sigma_{i,s,m,d}^{2,n}} \quad (25)$$

where n indicates the iteration number. The functions $\epsilon_w(n)$, $\epsilon_\mu(n)$ and $\epsilon_\sigma(n)$ control the learning rate, according to the equations

$$\epsilon_w(n) = \epsilon(n) \frac{\max_{p,q,r} w_{p,q,r}^n}{\max_{p,q,r} \Delta w_{p,q,r}^{n+1}} \quad (26)$$

$$\epsilon_\mu(n) = \epsilon(n) \frac{\max_{p,q,r} |\mu_{p,q,r}^n|}{\max_{p,q,r} |\Delta \mu_{p,q,r}^{n+1}|} \quad (27)$$

$$\epsilon_\sigma(n) = \epsilon(n) \frac{\max_{p,q,r} |\sigma_{p,q,r}^{2,n}|}{\max_{p,q,r} |\Delta \sigma_{p,q,r}^{2,n+1}|} \quad (28)$$

where the Delta terms in the denominators are the maximum values of the respective gradients at the $n+1$ 'th iteration. The function $\epsilon(\cdot)$ is monotonic decreasing as the iteration number n increases and always returns a small positive real number. For our experiments we used a function of the form

$$\epsilon(n) = \frac{a}{b + nc} \quad (29)$$

where a , b and c are experimentally determined constants, selected in an *a priori* fashion. Generally speaking, if a is too large and b and c too small then learning will be erratic, and the loss will fluctuate wildly. If a is too small, and b and c too large, then no learning will occur. For the experiments described later we selected the values $a=1$, $b=100$ and $c=1$. No attempt was made to optimise the choice of these values.

After updating at each iteration any negative mixture weights are set to zero and then all weights are re-normalised to sum to one within each HMM state. Obviously a problem occurs if all the weights have been reduced to zero prior to normalisation, however in practice, if the update scaling constants are small enough than this should never occur. The normalisation is required to maintain a probabilistic interpretation of the HMM scores. However, whether such an interpretation is actually necessary is an open question, and one for which further research is required.

Theoretically, the gradient descent can be begun with the HMM parameter vectors initialised to any arbitrary set of values, *e.g.* randomly initialised for instance. However, since minimum error training is only guaranteed to reach a local minimum of the loss function, it intuitively makes sense to begin training from the best possible starting point. Thus in this paper we assume, unless otherwise stated, that the HMMs are always initialised using conventional maximum likelihood training, prior to minimum error training. Thus the overall training procedure is really a combination of maximum likelihood followed by minimum error. Prior to minimum error classification, the models were initialised first using k -means clustering, then re-estimated using forward-backward re-estimation, and finally fine tuned using embedded, or connected training. The software used for this was the HTK software written by Cambridge University.

Finally, one more important point to note here is that we do not update the HMM transition probabilities using minimum error classification training. The reason for this being that they have only a small influence on the overall log probability.

2.7 Differentiating the Loss

In order to perform the gradient descent described in the previous subsection we need to be able to differentiate the loss wrt each different HMM

parameter type. The computation of these derivatives is facilitated by application of the chain rule, *i.e.* if $\alpha_{j,s,m}$ is any parameter from the m 'th mixture of the s 'th state of the j 'th HMM then

$$\frac{\partial L(\mathcal{O}, \lambda)}{\partial \alpha_{j,s,m}} = \sum_{k=1}^K \sum_{n=1}^{N_k} \frac{dl(d_k(\mathcal{O}_{k,n}, \lambda))}{dd_k(\mathcal{O}_{k,n}, \lambda)} \cdot \left[-\frac{\partial g(\mathcal{O}_{k,n}, \Lambda_k)}{\partial \alpha_{j,s,m}} + \frac{\partial g(\mathcal{O}_{k,n}, \Lambda_c)}{\partial \alpha_{j,s,m}} \right] \\ c = \underset{p}{\operatorname{argmax}} g(\mathcal{O}_{k,n}, \Lambda_p). \quad (30)$$

Using the short hand notation d_k for $d_k(\mathcal{O}_{k,n}, \lambda)$ we get

$$\frac{dl(d_k)}{dd_k} = \frac{\alpha l(d_k)^2}{\exp(\alpha(d_k - \beta))}. \quad (31)$$

Both differentials in the square brackets in Eq. (30) can be evaluated as follows

$$\frac{\partial g(\mathcal{O}_{k,n} | \Lambda_q)}{\partial \alpha_{j,s,m}} = \sum_{t=1}^{T_{k,n}} \delta(\psi_{q,t} - j) \delta(\theta_{q,t} - s) \cdot \frac{\partial \ln(b_{j,s}(\mathbf{o}_{k,n,t}))}{\partial \alpha_{j,s,m}} \\ \frac{\partial \ln(b_{j,s}(\mathbf{o}_{k,n,t}))}{\partial \alpha_{j,s,m}} = \frac{1}{b_{j,s}(\mathbf{o}_{k,n,t})} \frac{\partial b_{j,s}(\mathbf{o}_{k,n,t})}{\partial \alpha_{j,s,m}} \quad (32)$$

If $\alpha_{j,s,m}$ is equal to the weight vector $w_{j,s,m}$ then

$$\frac{\partial b_{j,s}(\mathbf{o}_{k,n,t})}{\partial \alpha_{j,s,m}} = \mathcal{N}(\mathbf{o}_{k,n,t}, \boldsymbol{\mu}_{j,s,m}, \boldsymbol{\sigma}_{j,s,m}^2). \quad (33)$$

If $\alpha_{j,s,m}$ is equal to the mean vector $\boldsymbol{\mu}_{j,s,m}$ then computing the differential on the rhs of Eq. (32), wrt the d 'th component of the vector, can be accomplished as follows

$$\frac{\partial b_{j,s}(\mathbf{o}_{k,n,t})}{\partial \mu_{j,s,m,d}} = w_{j,s,m} \frac{\partial \mathcal{N}(\mathbf{o}_{k,n,t}, \boldsymbol{\mu}_{j,s,m}, \boldsymbol{\sigma}_{j,s,m}^2)}{\partial \mu_{j,s,m,d}}. \quad (34)$$

Using the standard relationship

$$\frac{\partial f(x)}{\partial x} = f(x) \frac{\partial \ln f(x)}{\partial x} \quad (35)$$

and the fact that

$$\ln(\mathcal{N}(\mathbf{o}_{k,n,t}, \boldsymbol{\mu}_{j,s,m}, \boldsymbol{\sigma}_{j,s,m}^2)) \\ = -\frac{1}{2} \left[D \ln(2\pi) + \sum_{d=1}^D \ln \sigma_{j,s,m,d}^2 \right. \\ \left. + \sum_{d=1}^D \frac{(\mathbf{o}_{k,n,t,d} - \mu_{j,s,m,d})^2}{\sigma_{j,s,m,d}^2} \right] \quad (36)$$

we get

$$\frac{\partial b_{j,s}(\mathbf{o}_{k,n,t})}{\partial \mu_{j,s,m,d}} = w_{j,s,m} \mathcal{N}(\mathbf{o}_{k,n,t}, \boldsymbol{\mu}_{j,s,m}, \boldsymbol{\sigma}_{j,s,m}^2) \cdot \frac{(\mathbf{o}_{k,n,t,d} - \mu_{j,s,m,d})}{\sigma_{j,s,m,d}^2}. \quad (37)$$

Similarly for the variance

$$\frac{\partial b_{j,s}(\mathbf{o}_{k,n,t})}{\partial \sigma_{j,s,m,d}^2} = w_{j,s,m} \mathcal{N}(\mathbf{o}_{k,n,t}, \boldsymbol{\mu}_{j,s,m}, \boldsymbol{\sigma}_{j,s,m}^2) \cdot \frac{1}{2\sigma_{j,s,m,d}^2} \left[\frac{(\mathbf{o}_{k,n,t,d} - \mu_{j,s,m,d})^2}{\sigma_{j,s,m,d}^2} - 1 \right]. \quad (38)$$

3. EXPERIMENTAL WORK

A preliminary evaluation of the minimum error training algorithm described in the previous section was conducted using the ATR 5,240 Japanese word data base. Only data from one speaker (speaker MAU) was used, with just one realisation of each word. Training (*i.e.* both the initial maximum likelihood and the following minimum error training) was conducted on the even words, while testing was on the odd words.

All feature vectors comprised of 12 cepstral coefficients, 12 delta cepstral coefficients, a power and a delta power coefficient (*i.e.* 26 dimensions in all). Computation was done using a 25.6 ms hamming window with a 5 ms shift. The cepstral coefficients were computed via an intermediate 12th order LPC analysis.

We used 24 phoneme models from the standard ATR labelling scheme, *i.e.* /a/, /b/, /ch/, /d/, /e/, /g/, /h/, /i/, /j/, /k/, /m/, /n/, /ng/, /o/, /p/, /r/, /s/, /sh/, /t/, /ts/, /u/, /w/, /z/, /zh/, plus a silence model /sil/ and a closure model /cl/ (*i.e.* 26 models in all). For recognition purposes however, the silence and closure models were treated as identical. The grammar used both for training and recognition was totally unconstrained, *i.e.* any phoneme could follow any phoneme, and the phoneme string could be of any length.

As described in the last section, prior to minimum error training the HMMs were initialised using a conventional maximum likelihood algorithm, *i.e.* *k*-means initialisation, Baum-Welch re-estimation and finally embedded training.

The type of HMM used was a continuous gaussian mixture density model with no explicit duration modelling. All models were left to right 5 state models, with no skips. The first and last states were non-emitting. Both 3 and 10 mixture models were tested. The results are shown in Figs. 1 and 2. In both figures the horizontal axes show the number of minimum error training iterations, beginning with the standard maximum likelihood

models at iteration 1. Solid lines show the change in recognition performance on the closed training set as the number of minimum error iterations increase. The recognition results on the open test set are shown only for the initial maximum likelihood models and the best minimum error models. These results are indicated by the triangles. Both graphs show 3 types of recognition error:

Fig. 1 Recognition results for the 3 mixture HMMs.

Fig. 2 Recognition results for the 10 mixture HMMs.

Fig. 3 A comparison of the 3 and 10 mixture HMMs.

words correct, phonemes correct and phoneme accuracy. These are defined as per usual, phonemes correct including only substitution and deletion errors, while phoneme accuracy additionally includes insertion errors. Finally, Fig. 3 shows a direct comparison of the performance of the 3 and 10 mixture models on the open test set data. Of particular interest here is the fact that the 3 mixture minimum error model actually performs better than the 10 mixture maximum likelihood model.

4. CONCLUSIONS

The results need little comment; it is obvious that the minimum error HMMs perform much better than the maximum likelihood models. It is important to remember when looking at the results that these were preliminary experiments only, no attempt was made to optimise the choice of minimum error training parameters, *e.g.* learning rate, *etc.* Also we chose the simplest possible minimum error classification implementation. Thus the potential for improvement is large. In future we intend to extend upon this work by employing conjugate gradient descent algorithms, N Best discriminant functions and explicit HMM duration models, optimised using the minimum error criterion.

ACKNOWLEDGEMENTS

The authors would like to thank S. Katagiri and C. H. Lee for introducing them to the study of

generalised probabilistic descent and helpful discussions concerning its application to HMMs.

REFERENCES

- 1) A. Ljolje, Y. Ephraim, and L. R. Rabiner, "Estimation of hidden Markov model parameters by minimizing empirical error rate," Proc. ICASSP-90, 709-712 (1990).
- 2) H. Franco and A. Serralheiro, "Training HMMs using a minimum error approach," Proc. ICASSP-91, 357-360 (1991).
- 3) S. Katagiri, C. H. Lee, and B. H. Juang, "A generalized probabilistic descent method," Proc. Autumn Meet. Acoust. Soc. Jpn. 2-p-6, 141-142 (1990).
- 4) S. Katagiri, C. H. Lee, and B. H. Juang, "New discriminative algorithms based on the generalized probabilistic descent method," Proc. IEEE-SP Workshop Neural Networks for Signal Processing, Princeton, Sept. (1991).
- 5) S. Katagiri, C. H. Lee, and B. H. Juang, "Discriminative multi layer feedforward networks," Proc. IEEE-SP Workshop Neural Networks for Signal Processing, Princeton, Sept. (1991).
- 6) P. C. Chang, S. H. Chen, and B. H. Juang, "Discriminative analysis of distortion sequences in speech recognition," Proc. ICASSP-91, 549-552 (1991).
- 7) L. E. Scales, *Introduction to Non-linear Optimization*, Macmillan Computer Science Series (Macmillan, London, 1985).

David Rainton was born in Cambridge, England, in 1963. He received a B.Sc. Hons. in Applied Physics and Electronics, from Durham University in 1985. In 1987 he obtained an MPhil. in Computer Speech and Language Processing from Cambridge University, England, and in 1989 received a Ph. D. from Cambridge, for his research on time-frequency spectral analysis of speech. For the past 2 years, he has been a visiting researcher at ATR Interpreting Telephony Research Laboratories, Japan, engaged in speech recognition research activities. His research interests include speech recognition, signal processing, spectral analysis and array processing techniques.

Shigeki Sagayama was born in Hyogo Prefecture, Japan, in 1948. He received BE and ME degrees from University of Tokyo, Tokyo, Japan, in 1972 and 1974, respectively, both in Mathematical Engineering and Instrumentation Physics. He joined Electrical Communications Laboratories, Nippon Telegraph and Telephone Public Corporation, in 1974. Since 1990, he has been with ATR Interpreting Telephony Research Laboratories, where he is

D. RAINTON and S. SAGAYAMA: MINIMUM ERROR CLASSIFICATION TRAINING OF HMMS

currently in charge of Head, Speech Processing Department. His research interests include speech recognition, speech synthesis, signal modeling, and natural language. He is a member of the Acoustical Society of Japan, the

Institute of Electronics, Information and Communications Engineers, Japan, and the IEEE. He received the Invention Prize from the Institute of Invention, Japan, in 1991.