Computer Speech and Language (1997) **11,** 321–343

Training mixture density HMMs with SOM and LVQ

Mikko Kurimo*

Neural Networks Research Centre, Helsinki University of Technology, P.O. Box 2200, FIN-02015 HUT, Finland

Abstract

The objective of this paper is to present experiments and discussions of how some neural network algorithms can help to improve phoneme recognition using mixture density hidden Markov models (MDHMMs). In MDHMMs, the modelling of the stochastic observation processes associated with the states is based on the estimation of the probability density function of the short-time observations in each state as a mixture of Gaussian densities. The Learning Vector Quantization (LVQ) is used to increase the discrimination between different phoneme models both during the initialization of the Gaussian codebooks and during the actual MDHMM training. The Self-Organizing Map (SOM) is applied to provide a suitably smoothed mapping of the training vectors to accelerate the convergence of the actual training. The codebook topology which is obtained can also be exploited in the recognition phase to speed up the calculations to approximate the observation probabilities. The experiments with LVQ and SOMs show reductions both in the average phoneme recognition error rate and in the computational load compared to the maximum likelihood training and the Generalized Probabilistic Descent (GPD). The lowest final error rate, however, is obtained by using several training algorithms successively. Additional reductions from the online system of about 40% in the error rate are obtained by using the same training methods, but with advanced and higher dimensional feature vectors.

1. Introduction

The hidden Markov models (HMMs) (Rabiner, 1989) have been widely used for the automatic speech recognition (ASR) since Baker (1975) and Jelinek (1976). Other successful applications have been, for example, the handwriting recognition (Cho and Kim, 1995) and the modelling of the protein chains (Baldi and Chauvin, 1996). The power of the HMMs rely on the ability to combine the modelling of stationary stochastic

* E-mail: mikko.kurimo@hut.fi.

0885–2308/97/040321 + 23 $25.00/0/la970034

processes producing observable short-time features and the temporal relationships between these processes (Juang and Rabiner, 1991).

The traditional way to train the HMMs is by the maximum likelihood principle (Liporace, 1982), but the best recognition results are often obtained by discriminative training. Due to the lack of general efficient analytical formulation of the discriminative training for HMMs, artificial neural networks (ANNs) are very often used to constitute hybrid models combining their classification power with the time-domain modelling capability of HMMs (e.g. Bourlard and Wellekens, 1988; Lippmann, 1989). As shown in Niles and Silverman (1990), the HMM system itself can also be presented as an ANN. The simple adaptive form of discriminative training, learning vector quantization (LVQ) (Kohonen, 1986, 1990) has been successfully applied first as hybrid systems of discrete HMMs (DHMMs) with an LVQ-trained codebook (Iwamida, Katagiri, McDermott and Tohkura, 1990; Torkkola *et al.*, 1991) and later also as continuous density HMM (CDHMM) hybrids (Kurimo and Torkkola, 1992a; Katagiri and Lee, 1993), where the mixture density codebooks can be trained by LVQ. The attempts with partly heuristic corrective training has also provided good results for both DHMMs (Bahl, Brown, de Souza and Mercer, 1988) and CDHMMs (Mizuta and Nakajima, 1990). Via the generalized probabilistic descent (GPD) (Katagiri, Lee and Juang, 1991) the discriminative training related to the LVQ2 (Kohonen, 1990) can be formulated in a theoretically justified manner. The GPD is also a more general framework that can be shown to lead to several efficient implementations (e.g. Juang and Katagiri, 1992; Komori and Katagiri, 1992; Rainton and Sagayama, 1992).

The speech recognition project that produced this paper, started after the 1991 version of the Finnish phonetic typewriter (Kohonen, 1991; Torkkola *et al.*, 1991) that successfully integrated the DHMMs to the codebooks trained by LVQ. The current paper combines the LVQ3 algorithm (Kohonen, 1990) with the segmental training by Viterbi decoding (Rabiner, Wilpon and Juang, 1986) and the latest mixture density HMM (MDHMM) versions (Hwang *et al.*, 1994; Kurimo, 1994b; Peinado, Segura, Rubio and Benitez, 1994). In addition to the traditional maximum likelihood (ML) training, the training method is compared to the LVQ2 based corrective tuning (Kurimo, 1994a) and the segmental GPD (Chou, Juang and Lee, 1992). The idea of integration of the local acoustic context (Mäntysalo, Torkkola and Kohonen, 1992) and differential features to the standard short-time cepstrum is also tested as a way to improve the model performance and extend the current online system to exploit the improving hardware facilities. This test is also used to verify the applicability of the suggested training method and the initialization by self-organizing maps (SOMs) for different and higher-dimensional feature vectors.

This paper is organized so that Section 2 introduces the framework of the MDHMMs and summarizes the training methods related to the segmental LVQ3 training, which is then presented in Section 3. The phoneme recognition tests are given in Section 4 and the conclusions in Section 5.

2. Mixture density HMMs

2.1. The temporal model

The temporal model in the HMMs is a relatively simple structure of successive states and a probabilistic model of their mutual transitions. The modelling of the stochastic

observation processes associated with the states of HMMs is based on the estimation of the probability density of the short-time observations in each state. In this work the HMMs are applied to model the short-time cepstral features of the speech signal. The assumption is that during the articulation of the different phonemes, there are some states producing characteristic short-time features, which can be modelled by rather stationary stochastic processes. The temporal relations of these states are quite simple, because they usually follow each other in the same order so that the unidirectional, left-to-right HMMs can approximate the signal sources with an appropriate accuracy, if the number of allowed states is sufficient.

In the hidden Markov model (Rabiner, 1989) the sequence of short-time observations $\boldsymbol{O}=(\boldsymbol{O}_1, \boldsymbol{O}_2, \ldots, \boldsymbol{O}_T)$ are used to study the states $\boldsymbol{q}=(q_0, q_1, \ldots, q_T)$ the system undergoes. The production of observations is assumed to be stochastic and it is characterized by a set of observation probability measures $B=\{b_i(\boldsymbol{O}_t)\}_{i=1}^{N}$, where $b_i(\boldsymbol{O}_t)=\Pr(\boldsymbol{O}_t \mid q_t=i)$. The HMMs are based on the basic assumptions that the observations produced by states are assumed to be independent of each other and the time t and that the probability of state transitions depends only on the previous state. Thus, the joint probability of state sequence $\boldsymbol{q}$ being generated by the Markov chain and the observation sequence $\boldsymbol{O}$ being generated by that state sequence, can be calculated by the product

$$\Pr(\boldsymbol{O}, \boldsymbol{q} \mid \pi, A, B)=\pi_{q0}\prod_{t=1}^{T} a_{qt-1qt}b_{qt}(\boldsymbol{O}_t), \tag{1}$$

where $\pi_i=\Pr(q_0=i)$ is the probability of the initial state and $A=[a_{ij}]$, where $a_{ij}=\Pr(q_t=j \mid q_{t-1}=i)$, $i, j=1, \ldots, N$, the state transition probability matrix. A hidden Markov model is then defined by the triple $\lambda=(\pi, A, B)$.

2.2. Tied mixture density models

The modelling of the production of observations in different states of the system divides the HMMs into two categories: *discrete* and *continuous observation density* HMMs (Rabiner, 1989). In CDHMMs the $b_i(\boldsymbol{O}_t)$'s in (1) are some parametric probability density functions or mixtures of them. The most common parametric pdf used in this context is the mixture Gaussian density

$$b_i(\boldsymbol{O}_t)=\sum_{m=1}^{M} c_{im}b_{im}(\boldsymbol{O}_t),\ i=1, \ldots, N, \tag{2}$$

where $b_{im}(\boldsymbol{O}_t)$ is a K-dimensional multivariate Gaussian density with covariance matrix $\boldsymbol{\Sigma}_{im}$ and mean vector $\boldsymbol{\mu}_{im}$. The constraints for the mixture weights are: $c_{im} \geq 0 \forall m$ and $\Sigma_{m=1}^{M} c_{im}=1$, $i=1, \ldots, N$. The convergence of the common Baum–Welch re-estimation algorithm (Juang, 1985) requires only $b_{im}(\boldsymbol{O}_t)$ to be a strictly log-concave or elliptically symmetric covering also many other applicable parametric pdfs in addition to the multivariate Gaussians considered in this work. However, the use of mixture Gaussians is clearly the dominant trend in this research field.

To speed up the density computations and to reduce the number of parameters to be estimated the covariance matrices $\boldsymbol{\Sigma}_{im}$ are approximated by diagonal matrices tied

over all mixtures (Paul, 1989; Kurimo, 1994b). This also provides the possibility to considerably increase the number of mixtures to allow better density modelling accuracy without getting an excessive number of parameters. A related approach using weighted Gaussian kernel functions in ANNs is called the radial basis function (RBF) network (Broomhead and Lowe, 1988; Singer and Lippmann, 1992). The RBF networks are usually trained by first establishing the kernel functions and only then determining the weights.

Because the continuity of the acoustic features and the overlapping of the output densities of different states, it seems reasonable not to estimate a totally different set of mixture Gaussians for each state, but to *use the same large Gaussian codebook for several (or all) states* and only estimate different mixture weights for each state. This intermediate solution between the discrete and continuous HMMs is called the *semi-continuous* HMM (SCHMM) (Huang and Jack, 1989). The tied Gaussians resembles the vector quantization codebook of DHMMs, except that the densities are smoothly overlapped rather than partitioned. In addition to the more efficient use of the mixtures, the SCHMM also provides a more efficient model training. Like in DHMMs, the parameter estimation can be divided into two distinct phases: first estimate the codebook and then use that fixed codebook to estimate the other parameters. As proposed in Kurimo (1993) the latter phase can also include the re-adjustment of the Gaussian codebooks to achieve an optimal combination of all the parameters. By the terminology used in this work the codebook estimation phase is called the density initialization and the re-estimation phase is the actual training.

In this work the typing of the mixture densities is applied so that the states belonging to the same HMM, i.e. representing the same phoneme, use a common codebook of Gaussian densities (Kurimo, 1994b). Thus there are as many sets of Gaussians as there are HMMs, which is a kind of intermediate for CDHMMs (different set for each state) and SCHMMs (only one large set of Gaussians). Related approaches to *phoneme-wise tied Gaussian codebooks* have been recently proposed (e.g. Digalakis and Murveit, 1994; Hwang *et al.*, 1994; Peinado *et al.*, 1994; Zavaliagkos, Schwarz, McDonald and Makhoul, 1995). In addition to the training convenience, the phoneme-wise tied Gaussian codebooks are appealing, because the system balances between a vast number of mixture mean vectors and covariances, like in large CDHMM systems, and an excessive amount of mixture weights, like in large semi-continuous HMM systems. From the CDHMM point of view, having common Gaussians for states of the same phoneme is a tempting approximation, since the output densities of successive states are often similar and highly overlapping. From the SCHMM point of view, the normally large set of very small weights can be reduced, because most of the large weights for a certain state are often nicely localized around Gaussians resembling to one phoneme (see Kurimo, 1993).

2.3. Comments on maximum likelihood training

The traditional approach for the HMM training is to formulate the inverse estimation problem, i.e. the selection of the model parameters so that the likelihood of the training data being produced by the model is maximized. This maximum likelihood (ML) estimation can be conveniently accomplished by an iterative two-step procedure called *Baum–Welch re-estimation* (Baum and Petrie, 1966) to select the model λ in order to maximize $\Pr(\boldsymbol{O} \mid \lambda)$ for the given training sequence $\boldsymbol{O}$. In short, the first step is to

determine the state probabilities for the observation sequence and the second is to estimate the model parameters using the observations weighted by the state probabilities. A simplified version used in this paper as a reference model is the *Viterbi training* (a.k.a. segmental K-means (Juang and Rabiner, 1990)). The maximization of the so-called state-optimized likelihood, which approximates the above by using only the probability of the most probable state sequence, leads to a more straightforward implementation with less computation and numerical difficulties of low probabilities, because the Viterbi search (Forney, 1973) can be used for the fast state segmentation and the estimation phase is then just a simple K-means-type batch epoch. Despite the approximation, the same convergence properties as in Baum–Welch are valid (Juang and Rabiner, 1990).

The ML training is not, however, an optimal method in terms of minimizing the classification error rate in recognition tasks. The ML also supposes that the optimal model can be represented as a suitable parameter combination belonging to the a priori known model family. If the model family of the true model is unknown and incorrectly approximated, the ML training may operate very ineffectively. If the training data is completely labelled, i.e. the association between the data and the classes is precisely known a priori during the training as in supervised learning, the conditional maximum likelihood estimator (Nadas, Nahamoo and Picheny, 1988) can be used to gain robustness of the a priori model. This estimator can be shown to lead to the maximization of the mutual information (MMI) between the observation and class labels (Bahl, Brown, de Souza and Mercer, 1986).

2.4. Comments on corrective training

In practical training problems, like ASR systems, it is not necessary to reach the absolute optimum, but to get sufficiently close to it to obtain an acceptable accuracy. The idea in corrective training methods is to try to recognize the training words as if they were unknown and, when the result is not satisfactory, to modify the parameters to directly minimize the original training objective, i.e. the misclassification rate.

In Bahl *et al.* (1988) the parameter modification rules are presented for DHMMs and in Mizuta and Nakajima (1990) for CDHMMs. Due to the quite heuristic nature of these parameter modifications, the convergence is uncertain, but appears to work very well, in practice, producing significantly fewer recognition errors than ML or MMI estimations. However, choosing the set of parameters that minimizes the error rate on training data, do not, in general, minimize the error rate on test data. Therefore, it is necessary to have a large training database as representative as possible for the intended application. Corrective training methods are fast, because they do not include any complex gradient calculations, but the speed and success of the convergence may depend much on the selection of the constants to control the size of the adjustments. The *LVQ based corrective tuning* for semi-continuous HMMs (Kurimo, 1994a), differs from corrective training (Mizuta and Nakajima, 1990) by allowing adjustments only for incorrectly recognized phonemes and binding the learning rate not to the difference of the HMM path probabilities of the whole sample word, but to a gradually decreasing function of the iteration count aiming at the stochastic convergence. A similar approach has also been presented for training DHMMs (Galindo, 1995).

The probability differences of the HMM paths can be used to derive a smooth and differentiable cost function for misrecognitions. By the *generalized probabilistic descent* (GPD) paradigm (Katagiri *et al.*, 1991; Komori and Katagiri, 1992) the minimization

of the cost can be used to approximate the real error rate minimization. The objective is to provide a framework to develop faster and more reliable minimum error classification training algorithms. In GPD, the misclassification measure $d_k(\boldsymbol{O}, \lambda)$ of $\boldsymbol{O} \in C_k$ is defined by the difference of the logarithmic classification probabilities of the correct class C_k and the incorrect ones with the largest discriminant values. The corresponding cost function $l_k(d_k(\boldsymbol{O}, \lambda))$ can be, for example, a sigmoid (Rainton and Sagayama, 1992). The GPD guarantees that by adjusting the model with $-\varepsilon \boldsymbol{U} \nabla l_k$, where ε is a small positive learning rate factor and $\boldsymbol{U}$ a positive definite matrix, at least a local minimum of classification error is achieved in a probabilistic sense (Amari, 1967).

In Komori and Katagiri (1992) the general form of the minimum error classification rule (Katagiri *et al.*, 1991) is heavily simplified, thus preserving the basic characteristics, to observe that the GPD is actually a generalization of LVQ2 (Kohonen, 1990) applied to training the reference models according to the most probable path obtained by dynamic programming. The simplifications are obtained by using $L\infty$ norm instead of the general L_p norm ($p>0$), changing the general distance measures to the minimum operator, and approximating the sigmoidal function by a piece-wise linear step function to give the window constraint of the LVQ2 learning law (Kohonen, Kangas, Laaksonen and Torkkola, 1992). Due to the generality of the minimum error classification theory, the results of the GPD formulation can be correspondingly used to justify the LVQ based training methods presented in this paper.

A common feature in all these corrective algorithms is the need for the traditional maximum likelihood approach to first train the initial models. If the initial model performs poorly, the convergence for good results by the error corrections can be slow and inefficient. One reason might be that the discriminative training includes the tuning of some incorrect reference vectors away from the observations, which can have unwanted effects by isolating some reference vectors, if they do not manage to get tuned towards any correct observations. Also, it is not normally sensible to use the corrective training to all HMM parameters (e.g. the state transition probabilities).

2.5. Why initialize HMMs?

A common problem for all practically applicable iterative methods used for HMM parameter estimation is the selection of suitable initial parameter values. Although the methods usually obey some hill-climbing property due to their probabilistic nature, it is intuitively clear that the better the starting point the higher the probability of reaching good models within a reasonable number of literations.

The difference in the computational load between a proper initialization and extra epochs of the actual training depends on the size of the phoneme models, but an approximate relation observed in this paper is that the whole time spent in the initialization phase, if small SOMs are trained for each phoneme, corresponds roughly to the time of only one training epoch of the standard Viterbi training. For higher feature dimensions and mixture densities, the difference of the amount of computations grows even larger.

When striving for better models by increasing modelling complexity (number of HMMs, states, mixture densities and feature dimensions), the roughness of the error surface and the number of local optima usually increases as the size of the training database is increased in order to reliably estimate the model parameters. So careful

initialization not only saves time but also leads to better models because, in practice, the search must be finished long before the global optimum is reached.

2.6. New initialization methods

By using a self-organizing map (Kohonen, 1990), a high-dimensional input space can be mapped into an ordered set of codebook vectors $\{\boldsymbol{m}_i\}_{i=1}^{M}$ so that the input topology is maintained, i.e. nearby feature vectors will be mapped to nearby codebook vectors. In this work, the SOMs are applied to adaptively produce a representative clustering of the feature vectors for each state of HMMs on which the mixture densities are built. The obtained codebook topology can also be used in later phases, for example, to speed up the calculations to approximate the observation probabilities by using only the best matching unit and its nearest neighbours (Kurimo and Somervuo, 1996).

For HMMs, SOMs have been applied to train smoother DHMM codebooks (Zhao and Rowden, 1991; Monte, 1992) and initialize the mixture densities for CDHMMs and SCHMMs (Kurimo and Torkkola, 1992b; Kurimo, 1993). With CDHMMs the SOM codebook vectors are transformed to mean vectors of the mixture densities and the initialization of the mixture weights and the covariance matrices is based on the clustering of training samples based on those mean vectors. For CDHMMs it is observed (Kurimo and Torkkola, 1992b) that initialization using SOMs leads to a near-minimum recognition error rate in at least three times less epochs of training. The reason why SOM smoothing gives better initialization than other mappings (e.g. by the K-means, often with even smaller distortion) is that the generalization to the test data tends to be better and also all the mixtures contribute better to the density modelling (Kim *et al.*, 1994; Kurimo and Somervuo, 1996).

The SOM adapts to the input space by using the input samples $\boldsymbol{x}$ in a random order one at a time and adjusting the nearest codebook vector $\boldsymbol{m}_c$ (*best-matching unit*, bmu) and its neighbours to be closer to $\boldsymbol{x}$:

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) + h_{ci}(t)[\boldsymbol{x}(t) - \boldsymbol{m}_i(t)], \tag{3}$$

where $t = 0,1,\ldots$ is a discrete time index. The function $h_{ci}(t)$ determines the *neighbourhood* around the bmu and its value can be, for example, the *learning rate* (alpha) $\alpha(t) \in (0,1)$, if the array distance between $\boldsymbol{m}_i$ and $\boldsymbol{m}_c$ is smaller than *neighbourhood radius* $r(t)$ and zero otherwise (Kohonen, 1995). $\alpha(t)$ and $r(t)$ decrease gradually (while t increases) to obtain the desired form of convergence.

For the phoneme-based tied density codebooks (Kurimo, 1994b) used for the mixture density HMMs in this work, the initialization of the mean vectors could be done by building a large SOM covering all phonemes and then splitting it into separate phoneme codebooks according to the labelling. A faster and better way, however, is to use the initially segmented data to train a small separate SOM for each phoneme (Kurimo and Somervuo, 1996). In this way, an equal number of mixtures and a proper representation for the variation in rare phonemes, such as /D/ and /Ö/, can be ensured.

For DHMMs it was concluded in Iwamida *et al.* (1990) that the concern should not be with reducing spectral distortion in the vector quantizer codebook design, but rather with providing discrimination information to minimize the number of misclassifications. It is reported both in Iwamida *et al.* (1990) and Utela (1992) that tuning the feature vector classifier with LVQ to increase the discrimination decreases the recognition error

rate significantly. It is also possible to apply LVQ to the initialization of CDHMMs and SCHMMs (Kurimo and Torkkola, 1992b; Kurimo, 1993). The motivation for LVQ initialization is to test whether a discriminative initialization might guide the ML training to find solutions with high discrimination between models. SCHMM initialization experiments reported in Kurimo (1993) show that applying LVQ2 or LVQ3 after SOM leads to fewer recognition errors than initialization by the SOM alone. LVQ training is applied in the normal way using data vectors from pre-segmented (either by hand or by another HMM) training samples (e.g. Section 3.2). Since all the candidate reference vectors from all classes participate in the winner search for all data vectors, this initialization for the MDHMMs is slower than in the suggested single-phoneme SOMs mentioned above.

3. Discriminative training by LVQ

3.1. Minimization of the classification error by LVQ

When multiple codebook vectors are used to represent each class of the observed feature vectors, the optimal training of the codebook vectors is an essentially different problem to the quantization error minimization of classical vector quantization. The minimization of misclassifications can be treated using Bayesian decision theory, in which each sample $\boldsymbol{x}$ is classified into the class C_k of highest likelihood $p(\boldsymbol{x} \mid \boldsymbol{x} \in C_k)P(C_k)$. Here, $P(C_k)$ is the prior probability of class C_k.

The *learning vector quantization* (LVQ) (Kohonen, 1990) generates a VQ codebook to approximate the Bayesian decision surfaces by classification using the label of the best matching codebook vector. For mixture density HMMs, LVQ is applied in the training of the codebook of Gaussians to maximize the differentiation accuracy near the border areas between the phonemes. Compared to mixture densities estimated to approximate the conditional output densities $p(\boldsymbol{x} \mid \boldsymbol{x} \in C_k)$ of each state of the HMMs, mixture densities designed for optimal classification may not produce as accurate probability estimates but instead aim at the generation of likelihoods that discriminate the states better. To recognize the phonemes correctly, the selection of the correct sequence of states is, after all, more important than the determination of the correct probabilities of different state sequences.

3.2. The basic LVQ3

For a randomly selected training vector $\boldsymbol{x}$, the *two* best-matching codebook vectors $\boldsymbol{m}_c$ and $\boldsymbol{m}_{c'}$ are adjusted by LVQ3 (Kohonen, 1990) as follows: if the sample vector $\boldsymbol{x}$ appears to be near the border between two classes and $\boldsymbol{m}_c$ and $\boldsymbol{x}$ belong to the same class, but $\boldsymbol{m}_{c'}$ to a different class:

$$\begin{aligned} \boldsymbol{m}_c(t+1) &= \boldsymbol{m}_c(t) + \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_c(t)] \\ \boldsymbol{m}_{c'}(t+1) &= \boldsymbol{m}_{c'}(t) - \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_{c'}(t)]. \end{aligned} \tag{4}$$

If $\boldsymbol{m}_{c'}$ is the one that belongs to the correct class the signs in front of $\alpha(t)$s in (4) are changed. If all $\boldsymbol{m}_c$, $\boldsymbol{m}_{c'}$ and $\boldsymbol{x}$ belong to the same class the update rule is:

$$\begin{aligned} \boldsymbol{m}_c(t+1) &= \boldsymbol{m}_c(t) + \varepsilon\alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_c(t)] \\ \boldsymbol{m}_{c'}(t+1) &= \boldsymbol{m}_{c'}(t) + \varepsilon\alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_{c'}(t)], \end{aligned} \tag{5}$$

where $\varepsilon \in (0,1)$ is a stabilizing constant factor. The value of ε should reflect the width of the adjustment window around the border between classes c and c' (Kohonen, 1990) so that with a narrow window the stabilizing learning steps (5) are small (i.e. ε is small). The teaching gain $\alpha(t) \in (0,1)$ is decreased monotonically.

3.3. The segmental LVQ3 training

The success of LVQ based tuning (Kurimo, 1994a) to enhance the recognition accuracy of mixture density HMMs first trained by another method introduces the question, why not save training effort by integrating the corrective training with the basic HMM training. This would, hopefully, stabilize the convergence of the corrective training by rewarding the correct results and applying discrimination only when the incorrect decision is strong enough to displace the correct one. The proposition here is that the K-means part (the adjustment of the model parameters) of the Viterbi training could be replaced by LVQ3 training. Since the normal Viterbi estimation is sometimes called segmental K-means training, this training method could in the same manner be called *segmental LVQ3 training* (Kurimo, 1996).

In addition to the determination of the most probable correct state segmentation (path) $\boldsymbol{q}^l$ for each training word l in the Viterbi training, the application of LVQ3 requires also the best rival path $\bar{\boldsymbol{q}}^l$ for the discrimination. The finding of the latter path simulates the recognition of an unknown word. The two paths are compared, and for the coinciding states the density functions are updated to increase the observation probability for the current acoustic observation as usual. If mismatching states are found, the density of the correct state is adjusted as above, but the density of the best incorrect state is adjusted to lower the likelihood of the observation. Combining the effects of these two different updates, maximization of discrimination and likelihood, the result corresponds to a kind of *batch version of LVQ3*.

The tuning of a mixture density function corresponding to a state HMM is performed by selecting the nearest Gaussian mixture and applying the LVQ3 rule to tune its mean vector and the corresponding weight. An attempt to briefly illustrate the consequences of the density tuning in one dimension is presented in Fig. 1. Due to the nature of the segmental training, i.e. the adaptation of the models is followed by a new segmentation of the data followed by a new adaptation and so on, the adaptation phase is done in the batch mode.

The batch formulation of the adjustment of the mean vector $\boldsymbol{\mu}_{jm}$ of each Gaussian kernel $m = 1, \ldots, M_j$ of codebook j corresponds to the weighted average of all the associated data vectors, where the representatives of incorrect states have a contribution equal to the vector of the same size, but opposite direction (note that $[\boldsymbol{O}_t^l + (2\boldsymbol{\mu}^{old} - \boldsymbol{O}_t^l)]/2 = \boldsymbol{\mu}^{old}$). A data vector $\boldsymbol{O}_t^l$ affects the adjustment of $\boldsymbol{\mu}_{jm}$, if the corresponding state q_t^l on the decoded correct path uses codebook j and the index o of the best-matching Gaussian in that codebook equals m, thus

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\delta(j \in q_t^l)\delta(m=o)[\delta(q_t^{-l} = \bar{q}_t^l)\boldsymbol{O}_t^l + \delta(\bar{q}_t^l \neq q_t^l)(2\boldsymbol{\mu}_{jm} - \boldsymbol{O}_t^l)]}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\delta(j \in q_t^l)\delta(m=o)}, \tag{6}$$

where

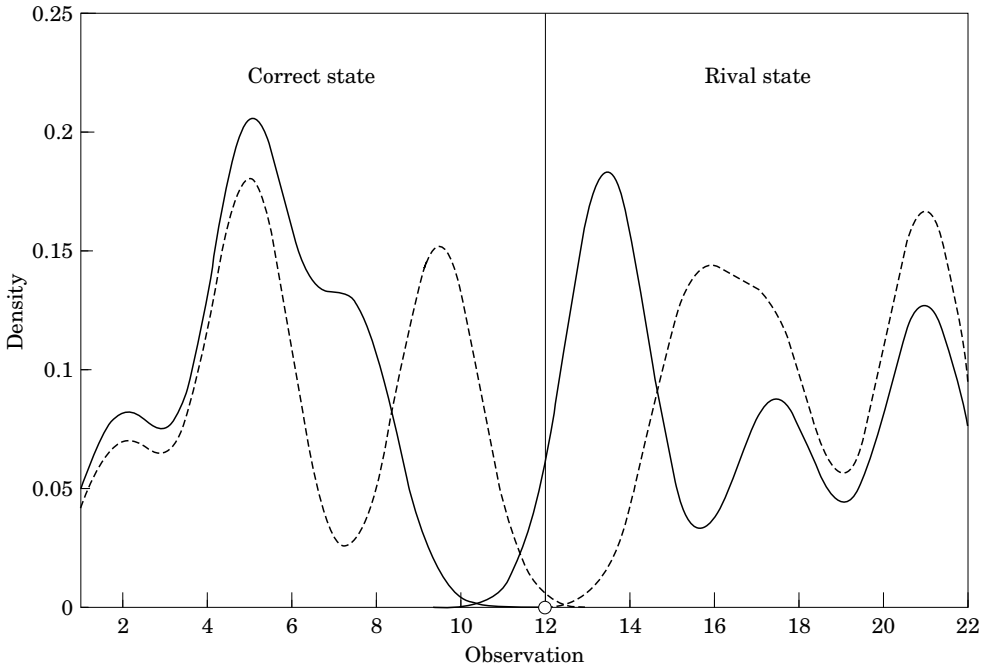

Figure 1. Adjusting the mixture densities of competing states for one observation (here, value 12). The parameters to be modified are the centroids of the nearest Gaussian for the correct state and, if a rival HMM state causes a misrecognition, also its corresponding centroid. The mixture weights of the modified mixtures are tuned respectively, but taking care of the normalization. The resulting new pdfs are shown dashed for this simple one-dimensional three-mixture case.

$$\delta(z)=\begin{cases}1, & \text{if } z \text{ is true,}\\ 0, & \text{if } z \text{ is false.}\end{cases} \tag{7}$$

For the mixture weight update, the idea is to increase the likelihood of a state by increasing the weight, if the Gaussian matches to a correct state and by decreasing the weight for a match to an incorrect state. To avoid any weight decreasing to zero or below, the decrease of a weight is substituted by increasing the other weights by the fraction representing their contribution (note that $\Sigma_{m=1,m\neq o}c_{im}=1-c_{io}$)

$$\hat{c}_{im}=\frac{\Sigma_{l=1}^{L}\Sigma_{t=1}^{T_l}\delta(i=q_t^l)[\delta(m=o)\delta(\bar{q}_t^l=q_t^l)+\delta(m\neq o)\delta(\bar{q}_t^l\neq q_t^l)c_{im}/(1-c_{io})]}{\Sigma_{l=1}^{L}\Sigma_{t=1}^{T_l}\Sigma_{m=1}^{M}\delta(i=q_t^l)\delta(m=o)}. \tag{8}$$

The summations are over all training samples: L is the number of training words, T_l the length of the current word in frames and $\{\boldsymbol{O}_t^l\}_{t=0}^{T_l}$ the corresponding sequence of feature vectors.

After the adjustments, the new parameter values λ are used to compute the paths again followed by the next iteration of parameter value update. The process can be iterated until the decided stopping criterion is fulfilled.

3.4. Comparisons to other corrective training methods

The main difference between segmental LVQ3 and corrective tuning using LVQ (Kurimo, 1994a) is best characterized by considering the difference between the underlying LVQ learning laws, LVQ3 and LVQ2. In tuning, the modifications of the output densities occur only for misclassified states, like in LVQ2, which makes the algorithm suitable mainly for fine tuning purposes. If misrecognitions occur frequently, the algorithm becomes slow and, perhaps, unstable, because reducing the likelihood of the incorrect models by tuning the densities away from the observations might have too much influence and disturb successful convergence. However, if the recognition occurs with few errors, the speed and stability increases and corrective tuning will be more efficient, because the learning is directed towards the likely trouble areas.

The second major difference compared to corrective tuning (Kurimo, 1994a) and also to Mizuta and Nakajima (1990) is the use of batch mode, i.e. adjustments are not applied after each misclassified vector, but they are stored in the memory and processed after the whole training data is checked. As a side effect, there will be less pre-specified control parameters for the learning and the order of the presentation of the training samples is guaranteed to have no effect.

The third difference to Kurimo, (1994a) is that the adjustments are not only applied to the Gaussian mean vectors, but also to the weight vectors and the transition probabilities. Corrective tuning was restricted to the mean vectors in order to avoid dramatic parameter changes due to a single recognition error.

Unlike segmental LVQ3, in discriminative training methods such as segmental GPD (Chou *et al.*, 1992) and ODT (optimal discriminative training) (Mitzuta and Nakajima, 1990), the extent of the whole word or sentence misclassification directly controls the magnitude of the parameter modification. However, the total misclassification extent does not necessarily represent well the situation in every part of the observation sequence. Another difference is that in segmental LVQ3, only adjustments to increase the difference between competing paths by increasing state likelihoods are allowed for phonemes that would already be correctly recognized.

4. Phoneme recognition experiments

4.1. Recognition system

The experimental evaluation is based on the speech recognition system of the Laboratory of Information and Computer Science at Helsinki University of Technology (Torkkola *et al.*, 1991). The system behaves like a phonetic typewriter to transcribe spoken Finnish words into phoneme sequences. The main difference here compared to the original system is the application of the continuous density HMMs instead of the discrete HMMs.

The short-time acoustical features used as observations of the speech signal are 20-component cepstra computed every 0·01 s from a 0·02 s partially overlapping window concatenated with the energy of the signal. The cepstral coefficients are obtained by a discrete cosine transform of a Mel-filtered power spectrum. For optimal classification ability, the cepstrum is weighted using a window of raised sinusoid. An example of a sequence of the obtained short-time features corresponding to the Finnish word "johdosta" is shown in Fig. 2.

The transformation of the most likely HMM path to the labels of the real word is a relatively straightforward procedure (in Finnish). Because each HMM corresponds

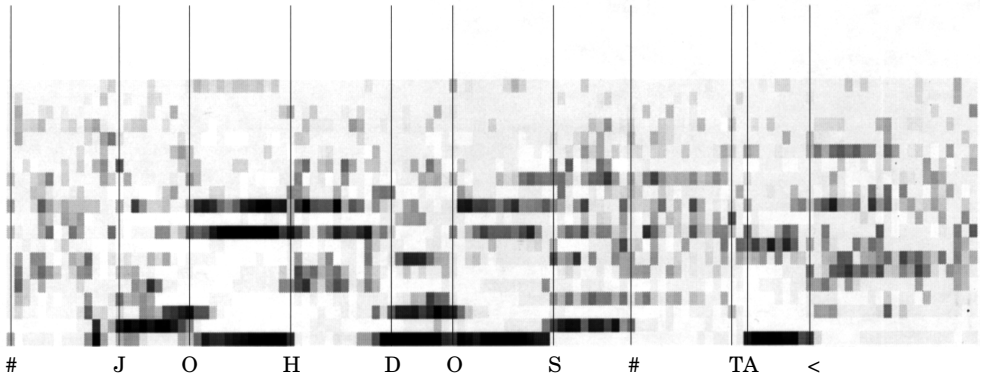

Figure 2. Short-time cepstra from a speech signal 10 ms apart. Time increases to right and quefrency (same as frequency for a spectrum) upward. The darkness of a pixel illustrates the value of the corresponding cepstral coefficient. The thin vertical lines show the starting points of the phonemes. /#/ means the silence before the word or a plosive and /</ the end of word.

to one phoneme the sequence of visited HMMs is directly the sequence of phonemes in the real word, which is quite near to the way the words are written in Finnish. The long phonemes (e.g. /AA/) are represented by the same HMMs as the short ones and their separation is based on the time spent in the corresponding HMM.

The comparison of the performance of different types of continuous density HMMs (Kurimo, 1994b) concluded that the phoneme-based tied HMMs provide the best configuration, when the combination of the number of parameters, the recognition time and the error rate was set as the selection criterion. The baseline phoneme model selected for this work is a 5-state unidirectional HMM with no skips and a density codebook of 70 mixture Gaussians per phoneme. In the first training phase the Gaussian mean vectors are trained by SOMs and in the second phase the training continues with embedded Viterbi (a.k.a. segmental K-means) training. The SOMs are arranged in a two-dimensional (10×7 units) hexagonal grid and for the rough organization (1000 samples) the neighbourhood radius decreases down from 8 and the learning rate down from 0·2. In the fine organization (10 000 samples), the radius goes down from 2 and the learning rate from 0·02. The final values for the radius and the learning rate are 1 and 0, respectively. As given in Kurimo (1994b) the total number of parameters for this baseline configuration is about 40 000 and the average error rate 5·7% (Data 90).

4.2. Speech material

The experiments were carried out using a set of 311 Finnish words and three native Finnish speakers (Data 90). From each speaker, four repetitions of the word set are available and the speaker dependent HMMs are trained with three sets per speaker leaving one set for testing. To verify the results with different data, seven speakers were chosen from a new database (Data 95). To get more variation, the models from the new data are trained with only the three first speech sets and tested with the fourth. The time spans between the first and the fourth speech sets range from one to seven months. The new data is recorded with a slightly higher sampling rate (16 vs. 12·8 kHz) and with longer and phonetically better balanced sessions (350 words), but the speakers used are not, on average, so experienced.

The recognition performance was determined by calculating the sum of changed, missing and extra phonemes in decoded phonetic transcriptions divided by the correct number of phonemes in the test material. The Matched-Pairs and the McNemar's test (Gillick and Cox, 1989) are used to determine the statistical significance of the difference in the average error rates. The Matched-Pairs test counts the difference in the number of errors in independent segments (a word, here) made by the compared algorithms. The significance level gives the risk of rejecting the hypothesis of no statistical difference. For example, the results are significantly different (at risk 0·05), if the difference in the number of errors is large enough to correspond to probability less than 0·05 in the normal distribution. The McNemar's test produces the corresponding risk level by comparing the number of differently recognized words to a chi-square distribution. The Matched-Pairs test result is given for most of the comparisons, but both results are provided, if necessary.

4.3. Comparison of initializations

The suitability of different initialization procedures for the HMM output densities are most reliably tested by running the whole parameter training several times and changing only the initialization method, then comparing the results. In addition to the final error rate it is also important that an appropriate recognizer performance is reached with as few training epochs as possible. In practice, the available training data is limited and often recycled for several epochs to give the algorithm time to converge. Thus, the number of epochs can also be a limiting factor for the learning of the optimal models.

Since the optimal performance is both difficult to define (0% error rate on this test data seems to be impossible with the limited training data) and difficult to reach (the error rates tend to decrease too slowly after the first few epochs), one suggestion is to set the lowest error rate obtained by the default training method as the benchmark and compare how fast the error rates get close enough starting from the different initializations.

The recognition tests are made for MDHMMs with Viterbi training by the segmental K-means after several alternative mean vector initializations (Table I). Starting with simple selection criteria (KNN, EVEN) requires at least five Viterbi training epochs to get even roughly near to the benchmark, whereas the advanced methods (KM, SOM) achieve that subgoal in only two or three epochs. If the training is continued to 10 epochs, the differences resulting from the initialization become small, but the benchmark still remains unreached for most of the models. In the comparison of the required Viterbi epochs to reach the benchmark result by the 95% confidence level the SOM based initializations beats clearly the rival methods (five vs. at least 10 epochs).

Another difference between the initializations is observed from the averaged learning curves (Fig. 3). While some initializations lead directly to the low rates (SOM, LVQ), others (KM, KNN) have a flat phase after an insufficient local minimum is achieved. The LVQ seems to give the smoothest learning curves, although the error rate is high at the beginning of the training.

4.4. Comparison of training algorithms

The comparisons between the average test set error rates of the proposed segmental LVQ3 and the conventional HMM training (the segmental K-means) and also the

TABLE I. Average test set error rates (Data 90 on Test I) for segmental K-means trained MDHMMs using alternative initializations for the Gaussian mean vectors. SOM learning is done in two phases (shorter and longer) for each phoneme. LVQ is trained for all phonemes at the same time and a short OLVQ1 phase precedes the main LVQ3 training. The other initializations are RAN = random vectors evenly distributed in the area of corresponding data vector components, EVEN = randomly selected reference vectors from the correct classes, KNN = correctly KNN-classified ($K = 5$) reference vectors and KM = K-means clustered vectors. The last column shows the number of Viterbi epochs required to reach the threshold for the good accuracy defined here to be the point, when the result is in average not statistically different (risk less than 0·05 by Matched-Pairs (MP) and McNemar (MN)) from the best result obtained by long Viterbi training (5·6% and 5·4% for Test I and II, respectively). The Test II performed on the high dimensional context vectors ("context80" in Section 4.6) and on the database Data 95, was performed to verify the comparison between the SOM and K-means initializations

Initialization	Error rate %		Epochs required	
Test I	3 ep.	5 ep.	MP	MN
RAN	10·5	8·2	>10	>10
EVEN	7·2	6·5	>10	>10
KNN	7·0	6·3	>10	>10
KM	6·4	6·0	>10	>10
SOM	6·0	5·8	6	5
LVQ3	7·0	6·3	>10	9
SOM + LVQ3	6·1	5·7	4	4
Test II				
KM	6·3	6·2	>10	>10
SOM	5·8	5·5	5	4

discriminative segmental GPD are shown in Fig. 4, where the proposed method seems to beat the rival methods both with respect to the fast convergence and low error rate. The average error rates for these combinations (Table II, five training epochs, Test I) are: SOM + LVQ3 5·3, KM + SKM 6·2 and KM + SGPD 5·8.

If the convergence speed is not important, the best final error rate is achieved by concatenating different training methods. In Table II the lowest average error rate is obtained by training the MDHMMs again by the segmental GPD after they have first been trained by another method (segmental K-means or LVQ3). However, in the verification test for the slightly easier database (but simpler models) the segmental GPD actually increased the error rate and the LVQ2 based tuning method gave the lowest error rate. It should be noted that this type of long training requires the availability of enough training data in order to avoid overlearning effects.

According to the statistical tests for Table II, there is, in general, no need for more than the five training epochs with one training algorithm. In addition to the segmental GPD, corrective tuning using LVQ2 also introduces a significant drop in the error rate for models trained first by segmental K-means. For models trained first by segmental LVQ3 that difference is smaller, probably indicating that the discriminative learning already occurs in the first part of the training.

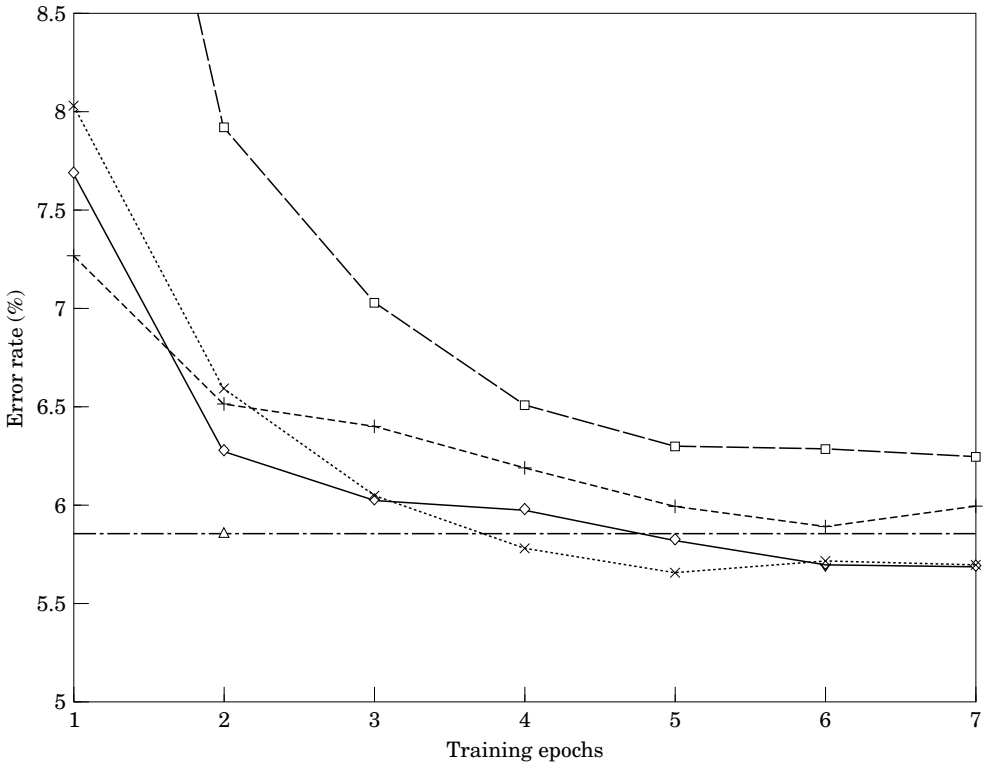

Figure 3. The development of the average test set error rate in the traditional (segmental K-means type) Viterbi training for different initializations for the density codebooks (see also Test I in Table I). The approximate limit rate is here defined to be reached, when the result is on average not statistically different from the best result obtained by full Viterbi training (5·6%): som (◇), km (+), knn (□), som+lvq (×), limit rate (△).

4.5. Boosting the recognition phase

When training a large number of mixtures to model the observation density of the state, the result is that different mixtures become sensitive to different sorts of realizations of the phoneme. Thus a single observation usually gets a large response of only a few mixtures and the rest provide much smaller responses. Further, the mixture coefficients will obtain many small and few large values, which strengthens even more the peaks in the observation probabilities produced by different mixtures (Kurimo, 1994b). This phenomenon can be effectively used in reducing the computation time for the observation probabilities of the states by an approximative solution of only K nearest mixtures instead of total M (see also Bellegarda and Nahamoo (1990)). Figure 5 illustrates the average error rates and recognition times using this approximation for different mixture sizes. It is remarkable that instead of decreasing as more mixtures are applied, the error rate actually increases after a certain limit, which is surprisingly small, for example K values 2,2,5 for M values 24,70,140, respectively. The average recognition error rates differ significantly for K values 4,5,10 or larger for M values 24,70,140, respectively. The explanation might be that the usage of only a few Gaussians emphasizes more

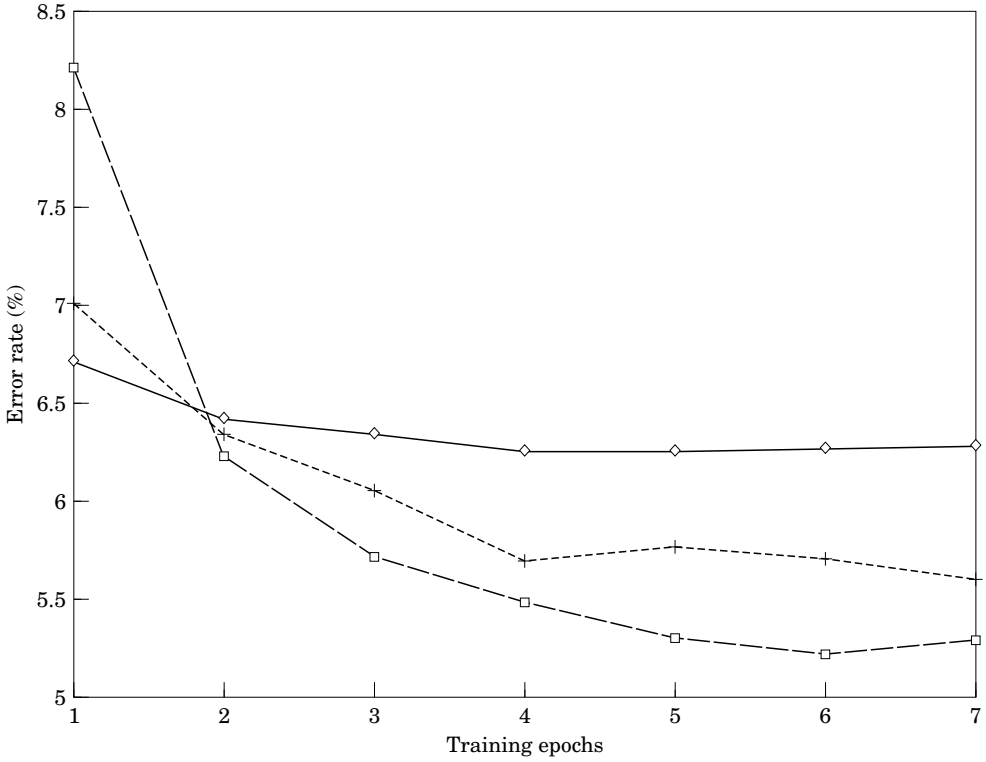

Figure 4. The development of the average test set error rate in the Viterbi training by the segmental LVQ3 initialized by SOMs and the reference methods segmental K-means and segmental GPD with K-means initialization (see also Test I in Table I): km + skm (◇), km + sgpd (+), som + slvq (□).

clearly the distinctive features of the states. Using all the Gaussians reflects perhaps more about how well the average features of states are represented. As shown in Kurimo (1994a) this probability approximation can also be successfully applied in the training phase.

When the centroids of the mixtures of a phoneme are initially organized by SOM, the mixtures providing high responses are often close to each other in the SOM topology, although the fine structure of the SOM is destroyed due to the training with zero neighbourhood. This can be exploited in the search of the K nearest mixtures by sorting only a small neighbourhood around the best matching mixture and tracking the trajectory towards the gradient of the match (Kurimo and Somervuo, 1996). The savings in the recognition time are important considering that the observation probability computations take quite a significant part of total computation time (over 50% including the signal preprocessing). More efficient ways to speed up the recognition by using the SOM topology are presented in Kurimo and Somervuo (1996).

4.6. Scaling up the system

One important aspect in developing the experiments for the previous sections was the ability to maintain an online demonstration system for speech recognition. The re-

TABLE II. Average test set error rates for alternative training methods after the initialization by K-means or SOM. The main test (I) uses "context80" features (see Section 4.6) on Data 95 and the verification test (II) "normal" features on Data 90. The training methods are segmental K-means (SKM), segmental GPD (SGPD), segmental LVQ3 (SLVQ3) and the corrective tuning based on LVQ2. In the two-method combinations the first method is used for the five first epochs. The statistical significance tests are Matched-Pairs (MP) and McNemar (MN). "=" means that the result has not changed significantly after five epochs (">" (or "<") otherwise)

Initialization	HMM training	Error rate %		Significance	
Test I		5 ep.	10 ep.	MP	MN
KM	SKM	6·2	6·1	=	=
SOM	SKM	5·5	5·4	=	=
KM	SGPD	5·8	5·6	=	>
SOM	SLVQ3	5·3	5·3	=	=
KM	SKM + SGPD	6·2	5·4	>	>
SOM	SKM + SGPD	5·5	4·8	>	>
SOM	SLVQ3 + SGPD	5·3	4·8	>	>
KM	SKM + LVQ2	6·2	5·6	>	>
SOM	SKM + LVQ2	5·5	5·2	>	>
SOM	SLVQ3 + LVQ2	5·3	5·2	=	>
Test II					
KM	SKM	6·0	5·9	=	=
SOM	SKM	5·8	5·7	=	=
KM	SGPD	7·1	6·7	>	=
SOM	SLVQ3	5·6	5·5	=	=
KM	SKM + SGPD	6·0	7·3	>	>
SOM	SKM + SGPD	5·8	7·3	<	<
SOM	SLVQ3 + SGPD	5·6	7·3	<	<
KM	SKM + LVQ2	6·0	5·5	>	>
SOM	SKM + LVQ2	5·8	5·5	>	>
SOM	SLVQ3 + LVQ2	5·6	5·4	>	>

cognition speed requirements restricted the previous experiments both in the sense of the feature space dimensions and the density codebook sizes for mixture density functions. However, since the computational capacity of the available workstations continuously increases, it is vital to study the next step: *what will happen to the developed modelling and training method, when the dimensions will be doubled or trebled.*

Most of the current speech recognition systems transform the speech into 10–15 dimensional cepstra and Δ-cepstra vectors. It is probably not worth adding higher cepstral coefficients, but codebooks built of concatenated and averaged combinations of features consisting of information over the local context seem to increase recognition accuracy (e.g. Mäntysalo *et al*., 1992). The scaling up experiments in this work include the variations of the feature vectors summarized in Table III.

The delta features were computed as a difference of the short-time features five frames apart. The context vectors are formed using several successive frames (see Fig. 6). The basic idea in context vectors is the same as suggested in Mäntysalo *et al*. (1992) for DHMMs and LVQ codebooks, but the dimension is slightly reduced by leaving out the outermost windows. The objective of using these extended feature vectors is to provide the HMMs more freedom to create component-wise sequential dependencies

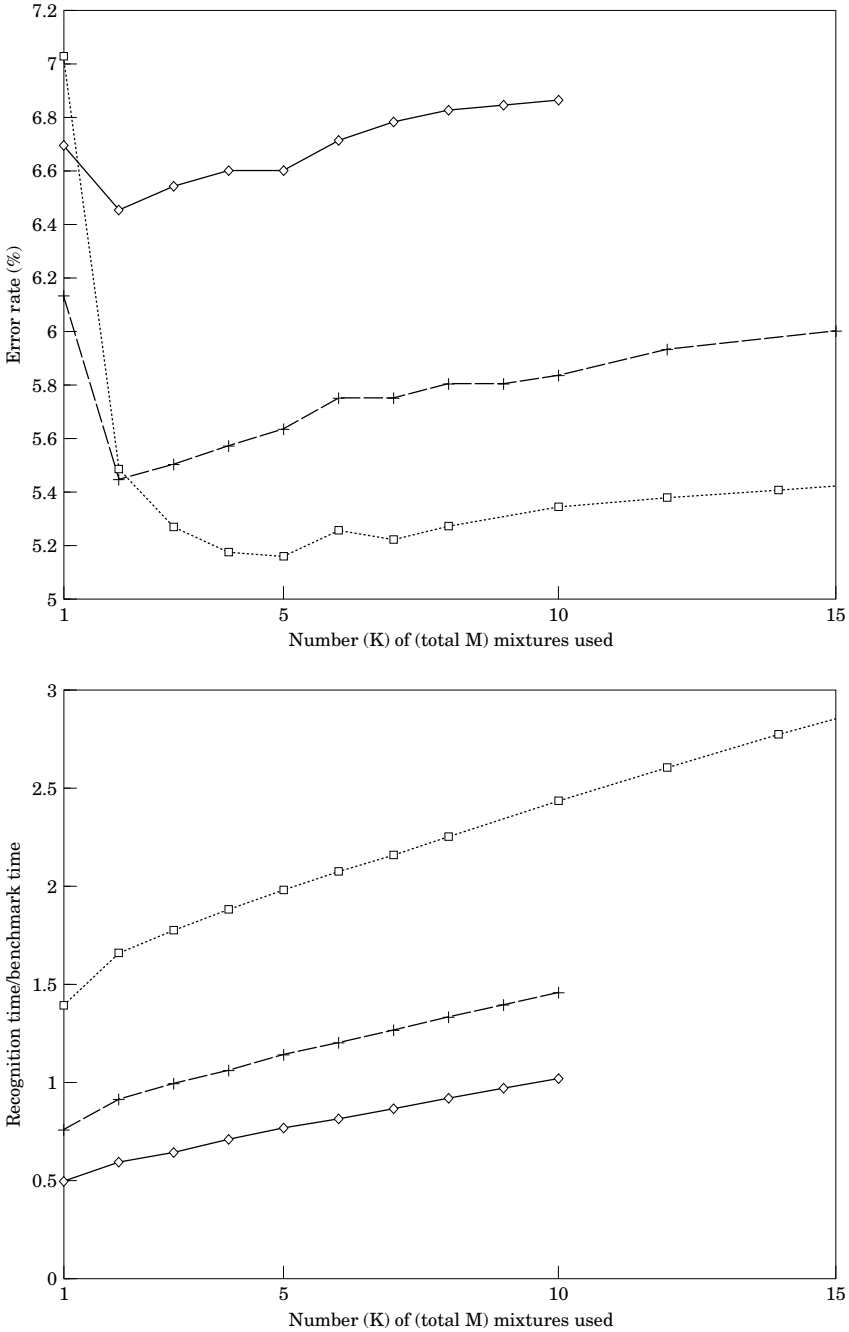

Figure 5. The average test set error rates and computation times when only K nearest mixtures instead of the total M are used to approximate the observation probabilities of HMM states. The M values 24, 70 and 140 were tested. The recognition time is the average recognition time per word divided by that of the baseline system ($K=3$ and $M=70$). The recognition times were recorded with a Silicon Graphics Power Challenge server using precomputed features: $M=24$ ($\diamond$), $M=70$ ($+$), $M=140$ ($\square$).

TABLE III. The contents of the alternative feature vectors in the tests

Feature vector	Cepstra normal	Δ	RMS normal	Δ	Context concatenation	Dimension in total
Basic	20		1		no	21
Delta21	10	10	1		no	21
Delta42	20	20	1	1	no	42
Context80	15		1		5 successive	80
Context105	20		1		5 successive	105

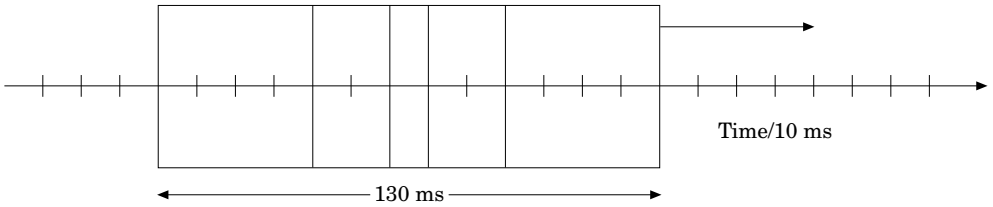

Figure 6. The context vector includes concatenated averages of one, two and four successive cepstral vectors. Successive context vectors are formed by sliding the window through the computed short-time features.

TABLE IV. The average test set error rates for alternative feature vectors described in the Table III. The MDHMMs are trained by SOMs and the segmental LVQ3. The recognition time factor is the average recognition time per word divided by that of the baseline system (70 mixtures and "basic" features)

Feature vectors	Error rate % Data 90	Data 95	Recognition time factor
Basic	5·5	7·7	1·0
Delta21	4·7	6·8	1·3
Delta42	4·0	6·4	1·8
Context80	3·6	5·3	1·7
Context105	3·4	5·3	1·8

by giving the observation densities of the states information on variable length features.

To compare the increase of the recognition performance, the error rates and the approximate recognition time factors are given in Table IV. The optimization of the recognition speed includes the partial distance computation mentioned in Kurimo and Somervuo (1996) that speeds up more the processing of the high than low dimensional input and also more the cepstrum than delta-cepstrum input. The corresponding comparison is also shown for changing the size of the mixture pool from the default

TABLE V. The average test set error rates for various mixture sizes and the default ("basic") feature vectors

Mixtures/ phoneme	Error rate % Data 90	Data 95	Recognition time factor
24	6·9	9·8	0·5
70	5·5	7·7	1·0
140	5·2	7·3	1·8

70 mixtures per phoneme (see Table V). The workstation based demonstration system currently operates with the "basic" features (Table IV), but it will soon[1] be able to process also the "context80" features online, like the other demonstration version equipped with special neural hardware. Although the best result is obtained by the "context105" features the difference to the "context80" is not significant.

4.7. Discussion

By studying the results more closely it was observed that although the difference of results between the methods were quite consistent for all speakers compared to the presented average values, the variation between the level of the results for speakers vary remarkably depending, perhaps, on the clarity, the speed and the dictation experience. For example, while the worst speaker could not give an error rate lower than 5·9%, the best speaker achieved 1·7%. The best obtained error rate for one speaker (1·7%) is quite high, since almost 80% of the remaining phoneme errors (actually 25 out of 32) were then caused by confusions between long and short versions of the same phoneme, for example /II/ and /I/, and some rare phonemes not modelled at all in these experiments. If the long phonemes could be correctly identified (e.g. using some special methods) and their separation errors were excluded from the average error rates, the improved recognition error rate for the "context80" features (see Table IV) and the two databases, respectively, would be as low as 1·9% and 3·5% instead of the given 3·4% and 5·3%.

It is obvious that the results obtained from different initialization methods vary depending on the random starting values. Because the experiments here show that the success in the initialization considerably affects the obtained final error rates, the effect of the random number seed was tested for otherwise equal initialization and training conditions. In this test the baseline training method (21 dimensional features, SOM initialization and segmental LVQ3 training) was performed for 10 different seeds. Each of the 10 speaker dependent training sessions included all the seven test speakers from the database Data 95. Each recognition test result was then, in turn, compared to the other nine to check by Matched-Pairs test, if the differences were statistically significant. Because a significant difference was observed using the 95% confidence level in only one of the 10 tests, it seems that the average error rates used through this section cover fairly well the variation caused by the selection of the random number seed. Anyhow,

[1] The extension to the "context80" features and 140 mixture components per phoneme was made in April 1997.

it would also be difficult, in practice, to always run every test with a large variety of different seed values.

5. Conclusions

In this paper it is observed that the convergence of MDHMMs to low recognition error rates is significantly accelerated by initializing the Gaussian mean vectors of the mixture densities using SOMs compared to the traditional initialization methods. The advantage offered by the SOMs is related to the way in which the codebook vectors of the SOMs are smoothed using nearby vectors in the topologically specified neighbourhood. After the initialization phase, to fully train the MDHMMs, the integration of LVQ3 in the segmental learning framework of the Viterbi training is formulated and tested. This training enhancement is found valuable, since it adds the beneficial discriminative effects of corrective tuning to the actual training so that separate additive tuning is no longer necessary to obtain low error rates. In the experiments segmental LVQ3 provides, on average, lower error rate and faster convergence compared to the conventional Viterbi training or the segmental GPD. However, if the training conditions and data quality allow the data to be recycled many times for long training, the successive usage of several training algorithms seems to provide the lowest final error rates.

According to the experiments to boost up the recognition phase, the approximation of the Gaussian mixture output density by using only a few nearest mixtures produces slightly lower error rates than by using many mixtures. This is assumed to be due to the increased discrimination between the different states, because then only the relations to the most representative mixture densities affects the resulting observation probabilities. The capabilities of scaling up the baseline system were also investigated by expanding the feature vectors to cover the local context of individual short-time features and by augmenting them with differential features. Reductions of about 40% in the error rate were obtained in these experiments.

The practical achievements of this project are utilized by the online ASR demonstration system for Finnish. Although the system is not intended to be a practical product, it shows that the proposed modelling and training methods can, in fact, be used in a pilot application operating in real time on current commercial workstation technology without any special hardware.

References

Amari, S. (1967). A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers* **16**, 299–307.

Bahl, L., Brown, P., de Souza, P. & Mercer, R. (1986). Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Tokyo, pp. 49–52.

Bahl, L., Brown, P., de Souza, P. & Mercer, R. (1988). A new algorithm for the estimation of hidden Markov model parameters. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New York, pp. 493–496.

Baker, J. M. (1975). The DRAGON system—an overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing* **23**, 24–29.

Baldi, P. & Chauvin, Y. (1996). Hybrid modeling, HMM/MLP architectures, and protein applications. *Neural Computations* **8**, 1541–1561.

Baum, L. & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics* **37**, 1554–1563.

Bellegarda, J. & Nahamoo, D. (1990). Tied mixture continuous parameter modeling for speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Process* **38**, 2033–2045.
Bourlard, H. & Wellekens, C. J. (1988). Links between Markov models and multi-layer perceptrons. In *Advances in Neural Information Processing Systems 1* (Touretzky, D. S., ed.), pp. 502–510. Morgan Kaufmann Publishers, San Mateo, CA.
Broomhead, D. & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems* **2**, 321–355.
Cho, S.-B. & Kim, J. H. (1995). An HMM/MLP architecture for sequence recognition. *Neural Computation* **7**, 358–369.
Chou, W., Juang, B. & Lee, C. (1992). Segmental GPD training of HMM based speech recognizer. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, San Francisco, pp. 473–476.
Digalakis, V. & Murveit, H. (1994). Genones: Optimizing the degree of mixture tying in a large vocabulary hidden Markov model based speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Adelaide, pp. 537–540.
Forney, G. D. (1973). The Viterbi algorithm. In *Proceedings of the IEEE* **61**, 268–278.
Galindo, P. L. (1995). A competitive algorithm for training HMM for speech recognition. In *Proceedings of 4th European Conference on Speech Communication and Technology (EUROSPEECH)*, Madrid, pp. 2187–2190.
Gillick, L. & Cox, S. (1989). Some statistical issues in the comparison of speech recognition algorithms. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Glasgow, pp. 532–535.
Huang, X. & Jack, M. (1989). Semi-continuous hidden Markov models for speech signals. *Computer Speech and Language* **3**, 239–252.
Hwang, M., Rosenfeld, R., Thayer, E., Mosur, R., Chase, L., Weide, R., Huang, X. & Alleva, F. (1994). Improving speech recognition performance via phone-dependent VQ codebooks and adaptive language models in SPHINX-II. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Adelaide, pp. 549–552.
Iwamida, H., Katagiri, S., McDermott, E. & Tohkura, Y. (1990). A hybrid speech recognition system using HMMs with an LVQ-trained codebook. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Albuquerque, pp. 489–492.
Jelinek, F. (1976). Continuous speech recognition by statistical methods. In *Proceedings of the IEEE* **64**, pp. 532–435.
Juang, B.-H. (1985). Maximum likelihood estimation for mixture multivariate stochastic observation of Markov chains. *AT&T Technical Journal* **64**, 1235–1249.
Juang, B.-H. & Katagiri, S. (1992). Discriminative learning for minimum error classification. *IEEE Transactions on Signal Processing* **40**, 3043–3054.
Juang, B.-H. & Rabiner, L. R. (1990). The segmental K-means algorithm for estimating parameters of hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing* **38**, 1639–1641.
Juang, B.-H. & Rabiner, L. R. (1991). Hidden Markov models for speech recognition. *Technometrics* **33**, 251–272.
Katagiri, S. & Lee, C.-H. (1993). A new hybrid algorithm for speech recognition based on HMM segmentation and learning vector quantization. *IEEE Transactions on Acoustics, Speech, and Signal Processing* **1**, 421–430.
Katagiri, S., Lee, C.-H. & Juang, B.-H. (1991). New discriminative training algorithms based on the generalized probabilistic descent method. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, Princeton, pp. 298–308.
Kim, D.-S., Lee, S.-Y., Han, M.-S., Lee, C.-H., Park, J.-G. & Suh, S.-W. (1994). Multidimensional HMM parameter estimation using self-organizing feature map for speech recognition. In *Proceedings of the 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing*, pp. 541–542. Fuzzy Logic Systems Institute, Iizuka, Japan.
Kohonen, T. (1986). Learning Vector Quantization for Pattern Recognition. Technical Report TKK-F-A601, Helsinki University of Technology.
Kohonen, T. (1990). The Self-Organizing Map. In *Proceedings of the IEEE* **78**, 1464–1480.
Kohonen, T. (1991). Workstation-based Phonetic Typewriter. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, Princeton, pp. 279–287.
Kohonen, T. (1995). *Self-Organizing Maps.* Springer, Berlin.
Kohonen, T., Kangas, J., Laaksonen, J. & Torkkola, K. (1992). LVQ_PAK: A program package for the correct application of learning vector quantization algorithms. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Baltimore, pp. 725–730.
Komori, T. & Katagiri, S. (1992). GPD training of dynamic programming-based speech recognizers. *Journal of Acoustical Society of Japan* **13**, 341–349.
Kurimo, M. (1993). Using LVQ to enhance semi-continuous hidden Markov models for phonemes. In

Proceedings of 3rd European Conference on Speech Communication and Technology (EUROSPEECH), Berlin, pp. 1731–1734.
Kurimo, M. (1994a). Corrective tuning by applying LVQ for continuous density and semi-continuous Markov models. In *Proceedings of International Symposium on Speech, Image Processing and Neural Networks (ISSIPNN)*, Hong Kong, pp. 718–721.
Kurimo, M. (1994b). Hybrid training method for tied mixture density hidden Markov models using Learning Vector Quantization and Viterbi estimation. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, Ermioni, pp. 362–371.
Kurimo, M. (1996). Segmental LVQ3 training for phoneme-wise tied mixture density HMMs. In *Signal Processing VIII* (Ramponi, G., Sicuranza, G. L., Carrato, S. & Marsi, S., eds), Proc. EUSIPCO, pp. 1599–1602. Edizioni Lint Triests, Italy.
Kurimo, M. & Somervuo, P. (1996). Using the Self-Organizing Map to speed up the probability density estimation for speech recognition with mixture density HMMs. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Philadelphia, pp. 358–361.
Kurimo, M. & Torkkola, K. (1992a). Combining LVQ with continuous density hidden Markov models in speech recognition. In *Neural and Stochastic Methods in Image and Signal Processing* (Su-Sing Chen, ed.), Proc. SPIE 1766, pp. 726–734.
Kurimo, M. & Torkkola, K. (1992b). Training continuous density hidden Markov models in association with Self-Organizing Maps and LVQ. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, Copenhagen, pp. 174–183.
Liporace, L. A. (1982). Maximum likelihood estimation for multivariate observations of Markov sources. *IEEE Transactions on Information Theory* **5**, 729–734.
Lippmann, R. (1989). Review of neural networks for speech recognition. *Neural Computation* **1**, 1–38.
Mäntysalo, J., Torkkola, K. & Kohonen, T. (1992). LVQ-based speech recognition with high-dimensional context vectors. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Banff, pp. 539–542.
Mizuta, S. & Nakajima, K. (1990). An optimal discriminative training method for continuous mixture density HMMs. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Kobe, pp. 245–248.
Monte, E. (1992). Smoothing HMMs by means of an SOM. In *Proceedings of 1992 International Conference on Spoken Language Processing (ICSLP)*, Banff, pp. 551–554.
Nadas, A., Nahamoo, D. & Picheny, M. (1988). On a model-robust training method for speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* **36**, 1432–1436.
Niles, L. T. & Silverman, H. F. (1990). Combining hidden Markov model and neural network classifiers. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Albuquerque, pp. 417–420.
Paul, D. P. (1989). The lincoln robust continuous speech organizer. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Glasgow, pp. 449–452.
Peinado, A. M., Segura, J. C., Rubio, A. J. & Benitez, M. C. (1994). Using multiple vector quantization and semicontinuous hidden Markov models for speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Adelaide, pp. 61–64.
Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* **77**, 257–286.
Rabiner, L., Wilpon, J. & Juang, B. (1986). A segmental *K*-means training procedure for connected word recognition. *AT&T Technical Journal* **64**, 21–40.
Rainton, D. & Sagayama, S. (1992). Minimum error classification training of HMMs—implementation details and experimental results. *Journal of Acoustical Society of Japan* **13**, 379–387.
Singer, E. & Lippmann, R. P. (1992). A speech recognizer using radial basis function neural networks in an HMM framework. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, San Francisco, pp. 629–632.
Torkkola, K., Kangas, J., Utela, P., Kaski, S., Kokkonen, M., Kurimo, M. & Kohonen, T. (1991). Status report of the Finnish phonetic typewriter project. In *Artificial Neural Networks* (Kohonen, T., Mäkisara, K., Simula, O. and Kangas, J., eds), Proc. ICANN, pp. 771–776. North-Holland, Amsterdam, Netherlands.
Uela, P. (1992). *Phoneme recognition with discrete density Markov models.* Master's Thesis, Helsinki University of Technology (in Finnish).
Zavaliagkos, G., Schwarz, R., McDonald, J. & Makhoul, J. (1995). Adaptation algorithms for large scale HMM recognizers. In *Proceedings of 4th European Conference on Speech Communication and Technology (EUROSPEECH)*, Madrid, pp. 1131–1134.
Zhao, Z. & Rowden, C. (1991). Application of Kohonen self-organising feature maps to smoothing parameters of hidden Markov models for speech recognition. In *Second International Conference on Artificial Neural Networks (Conf. Publ. No. 349)*, pp. 175–179. IEE, London, U.K.

(*Received 2 January 1996 and accepted for publication 7 November 1997*)