Ma 87 UDC 681.327.12:534.75

ACTA
POLYTECHNICA
SCANDINAVICA

MATHEMATICS, COMPUTING AND MANAGEMENT IN ENGINEERING
SERIES No. 87

Using Self-Organizing Maps and Learning Vector Quantization for Mixture Density Hidden Markov Models

MIKKO KURIMO

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

Thesis for the degree of Doctor of Technology to be presented with due permission for public examination and criticism in Auditorium F1 of the Helsinki University of Technology on the 3rd of October, at 12 o'clock noon.

Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory of Computer and Information Science

ESPOO 1997

Kurimo, M., **Using Self-Organizing Maps and Learning Vector Quantization for Mixture Density Hidden Markov Models**. Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series No. 87, Espoo 1997, 57 pp. Published by the Finnish Academy of Technology. ISBN 952-5148-30-0. ISSN 1238-9803. UDC 681.327.12:534.75

Keywords: Self-Organizing Map, SOM, Learning Vector Quantization, LVQ, Gaussians mixtures, density estimation, neural networks, speech recognition

ABSTRACT

This work presents experiments to recognize pattern sequences using hidden Markov models (HMMs). The pattern sequences in the experiments are computed from speech signals and the recognition task is to decode the corresponding phoneme sequences. The training of the HMMs of the phonemes using the collected speech samples is a difficult task because of the natural variation in the speech. Two neural computing paradigms, the Self-Organizing Map (SOM) and the Learning Vector Quantization (LVQ) are used in the experiments to improve the recognition performance of the models.

A HMM consists of sequential states which are trained to model the feature changes in the signal produced during the modeled process. The output densities applied in this work are mixtures of Gaussian density functions. SOMs are applied to initialize and train the mixtures to give a smooth and faithful presentation of the feature vector space defined by the corresponding training samples. The SOM maps similar feature vectors to nearby units, which is here exploited in experiments to improve the recognition speed of the system.

LVQ provides simple but efficient stochastic learning algorithms to improve the classification accuracy in pattern recognition problems. Here, LVQ is applied to develop an iterative training method for mixture density HMMs, which increases both the modeling accuracy of the states and the discrimination between the models of different phonemes. Experiments are also made with LVQ based corrective tuning methods for the mixture density HMMs, which aim at improving the models by learning from the observed recognition errors in the training samples.

The suggested HMM training methods are tested using the Finnish speech database collected in the Neural Networks Research Centre at the Helsinki University of Technology. Statistically significant improvements compared to the best conventional HMM training methods are obtained using the speaker dependent but vocabulary independent phoneme models. The decrease in the average number of phoneme recognition errors for the tested speakers have been around 10 percent in the applied test material.

PREFACE

This work has been carried out in the Neural Networks Research Centre, Helsinki University of Technology, during 1993–1997. I wish to thank Academy Professor Teuvo Kohonen for guidance and being the supervisor of the thesis and providing all the excellent facilities and financial support. His contribution has also been substantial to the whole research field around this work.

I wish to thank all the personnel of the Neural Networks Research Centre and the Laboratory of Computer and Information Science for creating a pleasant and productive working atmosphere. Special thanks for comments on the manuscript are due to Professor Erkki Oja and Dr. Samuel Kaski. I would also like to thank Dr. Jorma Laaksonen and Dr. Samuel Kaski for valuable aid in managing the incomparable document preparation system used for making all the publications and this thesis.

The financial support of the Academy of Finland as well as the Foundation for the Promotion of Technology and Emil Aaltonen Foundation is gratefully acknowledged.

Otaniemi, September 1997

Mikko Kurimo

CONTENTS

LIST OF PUBLICATIONS

The thesis consists of an introduction and the following publications:

1. Mikko Kurimo. Using LVQ to enhance semi-continuous hidden Markov models for phonemes. In *Proceedings of 3rd European Conference on Speech Communication and Technology, EUROSPEECH'93*, pages 1731–1734, Berlin, Germany, 1993.

2. Mikko Kurimo. 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'94*, pages 718–721, Hong Kong, 1994.

3. Mikko Kurimo. Hybrid training method for tied mixture density hidden Markov models using Learning Vector Quantization and Viterbi estimation. In John Vlontzos, Jenq-Neng Hwang, Elizabeth Wilson, editors, *Neural Networks for Signal Processing IV, Proceedings of the IEEE Workshop on Neural Networks for Signal Processing, NNSP'94*, pages 362–371, Ermioni, Greece, 1994.

4. Mikko Kurimo. Segmental LVQ3 training for phoneme-wise tied mixture density HMMs. In Giovanni Ramponi, Giovanni L. Sicuranza, Sergio Carrato, Stefano Marsi, editors, *Signal Processing VIII, Proceedings of the European Signal Processing Conference, EUSIPCO'96*, pages 1599–1602, Trieste, Italy, 1996.

5. Mikko Kurimo and Panu Somervuo. 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'96*, pages, 358–361, Philadelphia, PA, USA, 1996.

6. Mikko Kurimo. Training mixture density HMMs with SOM and LVQ. Helsinki University of Technology, Laboratory of Computer and Information Science, Report A43, January 1997, 26 pages. Also submitted for publication in a journal.

LIST OF ABBREVIATIONS

SOM	Self-Organizing Map
VQ	Vector quantization
LVQ	Learning Vector Quantization
HMM	Hidden Markov model
CDHMM	Continuous density HMM
MDHMM	Mixture density HMM
SCHMM	Semi-continuous HMM
PWMHMM	Phoneme-wise tied MDHMM
ASR	Automatic speech recognition
ML	Maximum likelihood
EM	Expectation maximization
BMU	Best-matching unit
GPD	Generalized Probabilistic Descent
ANN	Artificial neural network
MLP	Multilayer perceptron
TDNN	Time delay neural network
RBF	Radial basis function
MCE	Minimum classification error
RKDE	Reduced kernel density estimator
PDC	Partial distance computation
MMI	Maximum mutual information
MAP	Maximum a posteriori probability
PDF	Probability density function
RMS	Root mean square

LIST OF SYMBOLS

$\boldsymbol{x}$	input vector
$\boldsymbol{m}_i$	ith reference vector
c	index of the closest $\boldsymbol{m}_i$ to $\boldsymbol{x}$
c'	index of the second closest $\boldsymbol{m}_i$ to $\boldsymbol{x}$
N	number of reference vectors
t	discrete time index
h_{ci}	neighborhood function in SOM
α	learning rate
r	neighborhood radius in SOM
D	dimension of the input vectors
M	dimension of the SOM display
$\mathbf{1}_r$	vector of zeros except the rth component is 1
$\boldsymbol{I}_i$	location of the ith SOM unit on the SOM grid
$p(\boldsymbol{x})$	probability density function of $\boldsymbol{x}$
ϵ	stabilizing constant factor for LVQ3
C_i	ith class in LVQ
M	number classes in LVQ
$g_i(\boldsymbol{x})$	discriminant function of the ith class
$d_k(\boldsymbol{x})$	misclassification measure for $\boldsymbol{x} \in C_k$
$\boldsymbol{O}$	sequence observation vectors of a system
q	sequence of states a system undergoes
N	number of states in an HMM system
a_{ij}	transition probability from state i to state j
$b_i(\boldsymbol{O}_t)$	output probability density model of state i
π_i	starting probability of state i
λ	description of an HMM
$\boldsymbol{\mu}_{jm}$	mean vector of mth Gaussian in jth codebook
$\boldsymbol{c}_{im}$	weight of mth Gaussian for state i
K	number of best matching units used in density approximations
d	average quantization error

1 INTRODUCTION

Introduction to the problem. Many important pattern recognition problems arise from situations where a source has a finite number of states, and in each state it emits patterns having particular stationary characteristics. This assumption can also be used as a successful approximation in many practical tasks. One example is automatic speech recognition (ASR), where the pattern sequences are produced when a speaker modifies his articulatory mechanism. Another common example is automatic recognition of cursive handwriting which consists of arcs and lines drawn for the letters.

A straightforward approach to find out the state sequences of the system is to first divide the pattern sequences into short segments corresponding to different states and then to classify them using stationary models. The final recognition result is then obtained by classifying the sequence of these intermediate results. However, sometimes the segmentation and the final classification problems are tied closely together as also in the two given examples. The most likely rival result candidates may require different segmentations so that a separate segmentation phase may destroy some of the essential information for the classification.

In the pattern sequence recognition problems the information about the state of the system is often not transmitted explicitly, but only in the stochastic features of the emitted patterns. The feature vectors computed from the successive observations then form a discrete time stochastic process, where the individual feature vectors provide only local hints of the underlying states.

A method used in this work to join the segmentation and segment classification into a unified probabilistic approach is the hidden Markov model (HMM). The popularity of the model is based on its simple mathematical formulation that allows efficient implementations. The model is particularly widespread in ASR despite the fact that the Markov property, according to which the transition probability to a new state depends only on the current process state, does not accurately hold for speech.

The training of the HMM based recognition models occurs automatically eliminating most of the irritating manual segmentation and labeling work. For ASR, the models learn the characteristics of the states and transitions directly from the collected speech samples requiring only some initial values for the parameters and the phonetic transcription of the collected words. The most difficult and crucial thing to learn is how to model the emitted features of the states, because the densities of the outputs are not simply parameterizable and the models should also tolerate some variation in the data. In practice, there will probably always be states that do not get enough data to allow confident density estimates and yet the models should generalize well to unknown test data.

Artificial neural networks (ANNs) have some properties that suit well to the computation of the likelihoods of the states corresponding to the observed features. ANNs do not make as restricting assumptions on the parameterization of the densities as the conventional statistical methods and they can be flexibly extended and can operate with high speed by using only simple processing elements. Also simple training methods can be developed to allow the learning of the discriminative features between the models.

Introduction to this thesis project. The ASR research in the Laboratory of Information and Computer Science has long traditions since the first phoneme recognition system developed about 20 years ago. The system was able to segment and label speech into phoneme classes using the Learning Subspace Method producing phonemic strings which were correct to 81 % for one speaker (Jalanko, 1980). The corresponding word recognition accuracy of 94 % was obtained by this system in a closed thousand-word vocabulary applying a suitable post-processing. Later, the development of new algorithms and recognition methods in the laboratory and the opportunities to use dramatically more powerful computers have increased the average speaker dependent rate of correct phonemes in phoneme strings for unlimited vocabulary without post-processing now to about 97 %. This rate corresponds the average phoneme error rate of about 5 % that was measured in this work as an average of seven speakers and 350 test words.

The research on LVQ and HMMs for ASR started in the laboratory in 1990 from experiments with LVQ-trained codebooks for discrete observation density HMMs (Torkkola et al., 1991; Kohonen, 1991). The prominent system was used as a basis for several reports (Kurimo, 1992; Mäntysalo, 1992; Utela, 1992). The continuous density HMMs were first applied as a reference method, but after the introduction of the semi-continuous density models and LVQ based training methods the work leading to (Kurimo, 1994) and this thesis began to take shape. The motivation for this work has been to enhance the modeling and training methods to decrease the recognition errors produced by the HMM decoding system and study the effects of the extension of the system to use higher dimensional feature vectors.

In this work two ANN paradigms, the Self-Organizing Map (SOM) and the Learning Vector Quantization (LVQ) are used with HMMs to incorporate some especially useful features into the modeling of the phonemes. SOM is applied to initialize and train the output densities of the states of the HMM (see Section 3.2 for closer explanations). The output density model is a mixture of Gaussian density functions and by SOM the codebooks of Gaussians are organized into a grid such that Gaussians responding well to similar features are expected to be found near each other in the grid. The obtained smooth and ordered representation of the feature space assists in the full exploitation of the modeling capacity and fast search. The LVQ is used as a simple method to increase the discrimination ability of the density codebook to minimize the recognition error rate.

Contents of the thesis are organized as follows: Chapter 2 presents briefly the underlying methodology and the background for the thesis. The main algorithms for SOM, LVQ and HMMs are described and their properties central for this work are stressed. The main results and methods obtained by the author are summarized and motivated in Chapter 3. The exact experimental results are presented in the publications which are part of the thesis. The detailed description and formulation of the segmental LVQ3 training method and tuning method based on LVQ2 is given in the publications, as well. Brief conclusions are given in Chapter 4. At the end of this introductory part there is a short summary of the publications and a highlighted list of the main contributions of the author.

2 BACKGROUND

2.1 Self-Organizing Map

2.1.1 The SOM algorithm

The Self-Organizing Map (Kohonen, 1982; Kohonen, 1995; Kohonen et al., 1996a) is an unsupervised non-parametric regression process to represent nonlinear, high-dimensional data on a low-dimensional illustrative display. The input data points are mapped to SOM units on a usually one or two-dimensional grid. The mapping is learned from the training data samples by a simple stochastic learning process, where the SOM units are adjusted by small steps with respect to the feature vectors that are extracted from the data and presented one after another (in a random order).

The most familiar version of the algorithm (Kohonen, 1990b) uses the minimum Euclidean distance criterion for D-dimensional stochastic sample $\boldsymbol{x} \in \mathcal{R}^D$ to find the winner unit c (*the best-matching unit*, BMU) of the total of N SOM units. The criterion works so that the weight vector $\boldsymbol{m}_c$ of the BMU fulfills the condition

$$||\boldsymbol{x} - \boldsymbol{m}_c|| \leq ||\boldsymbol{x} - \boldsymbol{m}_i|| \, , \; \forall i = 1, \ldots , N \, . \tag{1}$$

The update of the SOM weights corresponding to $\boldsymbol{x}$ and c from the equation (1) is

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) + h_{ci}(t)[\boldsymbol{x}(t) - \boldsymbol{m}_i(t)] \, , \; \forall i = 1, \ldots , N \, , \tag{2}$$

where $t = 0, 1, \ldots$ is a discrete time index. The value of the *neighborhood function* $h_{ci}(t)$ can be, for example, the *learning rate* $\alpha(t) \in]0,1[$, if the array distance between $\boldsymbol{m}_i$ and $\boldsymbol{m}_c$ is smaller than the *neighborhood radius* $r(t)$ and zero otherwise (Kohonen, 1995). The desired form of convergence and characteristics of the result is ensured by the gradual decrease of $\alpha(t)$ and $r(t)$ while t increases.

For faster convergence and simpler computations, the learning rate control can be eliminated by using the iterative batch adaptation (Kohonen, 1992; Kohonen, 1993; Kohonen, 1995; Mulier and Cherkassky, 1995):

$$\boldsymbol{m}_i = \frac{\sum_{t=1}^{T} h_{ci}\boldsymbol{x}(t)}{\sum_{t=1}^{T} h_{ci}} , \tag{3}$$

where each iterative weight modification is based on the averaged effect of a set of samples. Even if the set of samples used in the batch steps is large enough, it is important to note that the modification steps (3) must be iterated several times, because the choice of the BMU (index c in equation 3) depends on the current values of m_is.

2.1.2 About the ordering and convergence

Ordering of the SOM units. The objective of the SOM is to provide a mapping which preserves the local topology of the input space in the low-dimensional display as faithfully as possible. The preservation of the topology means that nearby inputs are mapped to nearby SOM units.

Assuming that D is the dimension of the input samples, the dimension of the SOM display M can be chosen according to the purpose of the application. The larger M, the more faithful can the neighborhood relations of the input samples preserve, but the more difficult it is to observe the SOM ordering. For simple SOMs, where $M = D = 1$, it can be shown, for example in (Cottrell and Fort, 1987; Erwin et al., 1992a), that under certain conditions on the neighborhood function, the one-dimensional SOM will eventually be ordered so that the SOM weights will form either the sequence

$$m_1 < m_2 \ldots < m_N \quad \text{or} \quad m_1 > m_2 \ldots > m_N \,. \tag{4}$$

For multidimensional SOMs the convergence to an ordered state can be shown only for some special cases. For example, for $M = D > 1$ SOMs organized into a rectangular grid and input $\boldsymbol{x} \in [0,1]^D$, it is shown in (Flanagan, 1996) that the SOM will attain a dimension-wise ordered state (5) in a finite time with probability one. However, since no absorbing states for the general $M = D > 1$ SOMs are known, the process will also abandon that state in a finite time (Fort and Pages, 1996). According to (Flanagan, 1996), the $M = D > 1$ SOM is in a dimension-wise ordered state, if for all input dimensions $r = 1, \ldots, M$ holds either

$$m_{ir} < m_{jr} < m_{kr} \quad \text{or} \quad m_{ir} > m_{jr} > m_{kr} \tag{5}$$

for any triple i, j, k of SOM units, if

$$\boldsymbol{I}_k - \boldsymbol{I}_j = \boldsymbol{1}_r \quad \text{and} \quad \boldsymbol{I}_j - \boldsymbol{I}_i = \boldsymbol{1}_r \,. \tag{6}$$

In (6) the vector $\boldsymbol{I}_i \in \mathcal{N}^M$ shows the position of unit i in the SOM grid and the vector $\mathbf{1}_r$ is a M-dimensional vector of zeros except the rth component which equals to one. In (5) the component m_{ir} refers to the rth component of the weight vector of unit i.

Because no commonly accepted ordering definition is known for the normal case, where $D > M >= 1$, no valid proofs have been given for the general global ordering of the SOM. From the practical point of view, the confirmation of the organization in finite time from random initial state and random sample order is, anyhow, not crucial; it is more interesting to develop ways to produce the desired approximative mapping easily and quickly.

Asymptotic density of the SOM units. The convergence of the process can be forced by decreasing the learning rate gradually towards zero following a schedule that allows enough learning and controls the balance between the effect of successive adaptations. Stochastic approximation (Robbins and Monro, 1951) can be used to reach a suitable stable state for the SOM and for a $M = D = 1$ SOM it can be shown (Ritter and Schulten, 1988) that the following conditions are necessary and sufficient to reach the existing stable state from all sufficiently close initial states with a pre-specified neighborhood function using any small positive function $\alpha(t)$:

$$\lim_{T\to\infty} \int_0^T \alpha(t)dt = \infty \,, \tag{7}$$

$$\lim_{T\to\infty} \alpha(T) = 0 \,. \tag{8}$$

As shown in (Ritter, 1989) for $M = D = 1$ SOMs, the width of the neighborhood function h_{ci} affects strongly the asymptotic point density of the SOM units. The asymptotic point density is the number of units in the small unit area of input space and is given proportional to the probability density of the input samples $p(x)$. This relation of the densities is also known as the magnification factor (Kohonen, 1995; Bauer et al., 1996) of the SOM. If no neighbors are adjusted, the magnification factor is proportional to $p(x)^{1/3}$ and as the number of adjusted neighbors is increased it will increase towards $p(x)^{2/3}$ (Ritter and Schulten, 1986). If the number of adjusted neighbors is constant through the learning process the asymptotic value of the magnification factor can be determined exactly (Ritter, 1989; Ritter, 1991).

For multidimensional input the analysis of the magnification factor of the SOM is very difficult. For the classical vector quantization (VQ) with no topology $(r = 0)$ the exponent of the magnification factor proportionality discussed above will be quite close to one, if the dimension is high. It can be shown that the density of the best random quantizer is proportional to $p(\boldsymbol{x})^{\frac{D}{D+2}}$, where D is the dimension of the input vectors (Zador, 1982; Cottrell et al., 1997).

Depending on the shape of the neighborhood function, there may exist stationary states other than the ordered state (4) as can be shown for $M = D = 1$ SOMs (Erwin et al., 1992b). Due to the meta-stable states the ordering time may increase by orders of magnitude, if too narrow pre-specified neighborhood function is used throughout the process. According to empirical experience with different SOMs and different input the organization is most conveniently achieved by applying wide neighborhood at first, then decreasing it gradually and finally using very slow decrease, when the number of affected neighbors approach one (Kohonen, 1990b; Erwin et al., 1992b; Kohonen, 1995).

2.1.3 Some SOM applications and properties

The topology preservation has the property to map slightly altered input vectors to the original BMU or its immediate neighbors. This property is useful in different kinds of error tolerant mappings and smooth function approximations, required, for example, in transmission of signals (Luttrell, 1990; Bradburn, 1989; Kangas, 1995).

The topology preserving mapping from the high-dimensional input space to the low-dimensional display provides an opportunity to illustrate the relationships between different input regions. It is also possible to use SOM to find out the most significant data components that affect the ordering by viewing the component planes of the SOM weight vectors. The ability to visualize multidimensional data is exploited in many application areas, for example, data mining (Kaski, 1996) and process analysis (Kasslin et al., 1992).

The probability density of the input space is projected to the SOM by the point density of units such that the SOM weight vectors are at the densest in the areas where the most accurate vector quantization is needed. In addition to aiding in some VQ applications, this offers a way to approximate the probability density function (PDF) of the input. The PDF approximation can be used directly in maximum likelihood classification of static vectors (Hämäläinen, 1995) or, combined with state transition probabilities, to classify pattern sequences in the HMM framework as explained in this work (Section 3.2.4). The SOM is, however, not intended for optimal classification, but the classification accuracy of the SOM codebook can be improved by LVQ methods (Kohonen, 1995) which are discussed for this work in Sections 2.2 and 3.3.

The smoothness of the obtained mapping allows efficient exploitation of the available modeling capacity, when the amount of codebook vectors is much larger than the number of the distinct input clusters. Insufficiency and defects in the training data produces often some important training problems, for example, missing components in data vectors, over-learning and need for parameter smoothing. However, in SOM training their effect is not necessarily so bad since the use of training neighborhood provides smoothing also for units that are seldom chosen to be the BMU (Kohonen, 1995).

If a set of best matching SOM units, say the top five ranks, are wanted for the application, the ordering of the units can be used to apply some very fast approximative search methods for large codebooks. Such an approximation can be sufficient, if the exact identity of the BMUs is not crucial, but only the quantization errors should be of correct order of magnitude to show how well the input matches with the SOM. This is typical for using SOM for density function approximation purposes as in this work. For example, the search could then begin from the neighborhood of good BMU candidates and gradually proceed towards the direction of improved match values (Kohonen, 1996) (see the Section 3.2.5 for a more exact description).

Comprehensive views on different SOM applications are given in (Kohonen, 1995; Kohonen et al., 1996d).

2.2 Learning Vector Quantization

2.2.1 LVQ algorithms

The Learning Vector Quantization (LVQ) (Kohonen, 1986b; Kohonen et al., 1988) aims at defining the decision surfaces between the competing classes. The decision surfaces obtained by a supervised stochastic learning process of the training data are piecewise-linear hyperplanes that approximate the Bayesian minimum classification error (MCE) probability.

There exist several versions of LVQ described in (Kohonen, 1995; McDermott, 1990; Makino et al., 1992) which have slightly different properties. The ones used here are the LVQ2 (Kohonen et al., 1996b), which is efficient for fine tuning the decision borders between the competing classes and the LVQ3 (Kohonen et al., 1996b) which adds a stabilizing term to LVQ2 for improved long-run behavior.

LVQ2. The version described here as LVQ2 corresponds actually the LVQ2.1 (Kohonen, 1990a). For each stochastic input sample $\boldsymbol{x} \in \mathcal{R}^D$, the adjustments are performed for the *two* best-matching codebook vectors $\boldsymbol{m}_c$ and $\boldsymbol{m}_{c'}$, which are found using the minimum Euclidean distance criterion (1).

If $\boldsymbol{m}_c$ and $\boldsymbol{x}$ belong to the same class, but $\boldsymbol{m}_{c'}$ and $\boldsymbol{x}$ to different classes, respectively, then

$$\begin{aligned} \boldsymbol{m}_c(t+1) &= \boldsymbol{m}_c(t) + \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_c(t)] \\ \boldsymbol{m}_{c'}(t+1) &= \boldsymbol{m}_{c'}(t) - \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_{c'}(t)] \,, \end{aligned} \tag{9}$$

where the teaching gain $\alpha(t) \in]0,1[$ is decreased monotonically. An additional constraint for the update is that the sample $\boldsymbol{x}$ should lie in a narrow window between $\boldsymbol{m}_c$ and $\boldsymbol{m}_{c'}$ (Kohonen, 1995).

LVQ3. The same adjustments as in LVQ2 (9) occur, but there is also an extra weight update for the case where all $\boldsymbol{m}_c$, $\boldsymbol{m}_{c'}$ and $\boldsymbol{x}$ represent the same class:

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) + \epsilon\alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_i(t)] \ , \tag{10}$$

where $\epsilon \in]0,1[$ is a stabilizing constant factor and $i \in \{c, c'\}$. The value of ϵ should reflect the width of the adjustment window around the border between classes c and c' so that with a narrow window the stabilizing learning steps (10) are small (i.e. ϵ is small) (Kohonen, 1990b). In some cases the window constraint is unnecessary and it can be removed and thus $\epsilon = 1$ applied as suggested in (Kohonen, 1995). For example, if the reference vectors are carefully initialized, most of the few samples that still satisfy the other update conditions of the LVQ2, exist already around the boundary area (Katagiri and Lee, 1993). In that case the window constraint may be unnecessary and, may even reject some useful modification and slow down the learning.

2.2.2 Minimization of classification errors

There exist several proofs for the convergence of the traditional discriminant methods in two-class problems with separable pattern distributions (Nilsson, 1965; Amari, 1967). However, equally strong proofs have not been given for the multiclass tasks, very common in practice, where the pattern distributions are non-separable and of unknown functional form. Some probabilistic gradient methods can be shown to approach on the average the optimal classification solutions for an infinite amount of training data, but there may exist difficulties with small training sets. Furthermore, the performance is often measured only in a particular independent test data set so that the theoretical convergence properties do not necessarily coincide well with practical results.

In the LVQ learning laws (Kohonen, 1995) much emphasis has been given to the average performance in different difficult practical classification experiments in order to ensure that the methods will actually work in various contexts. The classification has been determined according to the class of the nearest reference vector as explained in the previous section. The selection of the nearest reference and the parameter adjustment formulas are normally expressed using the Euclidean distance metric, but corresponding expressions can be derived for other metrics that may be more suitable for certain applications (Katagiri and Lee, 1993).

Measuring the misclassification. In LVQ and stochastic gradient descent algorithms the number of expected classification errors is minimized using misclassifications or near-misses observed in stochastic training samples. For the small model adjustment steps corresponding to each such sample, some continuous measure for the degree of misclassification is useful. For two-class problems the measure is based on the difference between the Bayesian class discriminants g_i defined using the distances to the closest references from the competing classes. For example, the discriminant for class i can be simply

$$g_i(\boldsymbol{x}) = ||\boldsymbol{x} - \boldsymbol{m}_c||^{-2} , \tag{11}$$

where the index c of the BMU (1) is separately determined among the set of references of each class. It is also possible to use an appropriately weighted sum of contributions of several closest references or reference sequences as in (Chang and Juang, 1992; McDermott and Katagiri, 1994).

For decisions between multiple classes (M) there exist several possibilities. In (Juang and Katagiri, 1992) the misclassification of $\boldsymbol{x} \in C_k$ is measured by

$$d_k(\boldsymbol{x}) = -g_k(\boldsymbol{x}) + [\frac{1}{M-1} \sum_{i,i \neq k}^{M} g_i(\boldsymbol{x})^{\eta}]^{1/\eta} , \tag{12}$$

which is a continuous extension of the one used in (Amari, 1967), where only the discriminant differences between the correct class and the confusing classes ($g_i > g_k$) are averaged. In (12) η is a positive number to control the relative effect between larger and smaller discriminants.

In (Juang and Katagiri, 1992) definition (12) is used to define a framework of discriminative training methods called GPD (Generalized Probabilistic Descent) to minimize the expected cost of the misclassifications. In the extreme case of (12), where $\eta \to \infty$, only the largest discriminant (class C_i) affects the misclassification measure

$$d_k(\boldsymbol{x}) = -g_k(\boldsymbol{x}) + \max_{i,i \neq k} g_i(\boldsymbol{x}) . \tag{13}$$

This approach leads to the LVQ2 algorithm (Komori and Katagiri, 1992). In (Katagiri et al., 1991) it is shown that the practically efficient simple LVQ2 actually does approximate well the more complicated gradient search GPD implementations.

Why LVQ? The minimization of classification errors is the main objective in most pattern recognition applications. This is often approached by modeling of the probability densities of the competing classes, but since it is, in practice, often not possible to assume any proper parametric density model, the lowest error rate is obtained by concentrating on the actual discrimination between the classes.

The methods based on neural networks may outperform other methods in tough problems, where the prior knowledge cannot help much in the classification and the system characteristics must be learned automatically from the data. For those complicated situations it is advantageous that the algorithm consists of a large number of very simple units capable of learning locally. This kind of structure can be efficiently parallelized to exploit all the available computational power. Simple algorithms often have the tendency to be widely applicable and provide easy integration with other methods for efficient hybrid solutions.

2.3 Hidden Markov Models

2.3.1 Description of the model

The main idea behind HMMs is to integrate a simple and efficient temporal model and the available statistical modeling tools for stationary signals into a sound mathematically tractable framework. The majority of applications have been in speech recognition (Rabiner, 1989; Juang and Rabiner, 1991), but successive results are also reported in other fields, for example, in the handwritten script recognition (Cho and Kim, 1995) and in the modeling of the protein chains (Baldi and Chauvin, 1996).

Assumptions and definitions. The basic assumption in the Markov models is that the system has a finite number of possible states, but it can occupy only one of them at a time. In each state characteristic signals are emitted and the signal features are observed independently for time intervals $t = 1, \ldots, T$. For a sequence of observed feature vectors of the system, $\boldsymbol{O} = (\boldsymbol{O}_1, \boldsymbol{O}_2, \ldots, \boldsymbol{O}_T)$, there exists then the corresponding hidden sequence of states, $q = (q_0, q_1, \ldots, q_T)$, that the system has visited starting from the initial state q_0. Another basic assumption is the *Markov property*, i.e. the transition probability of the system to the next state depends only on the previous state regardless of the earlier transition history.

The HMM consists of the state transition probability matrix $A = [a_{ij}]$, where

$$a_{ij} = \Pr(q_t = j | q_{t-1} = i) \, , \; i, j = 1, \ldots, N \, , \tag{14}$$

and the set of state probability density models $B = \{b_i(\boldsymbol{O}_t)\}_{i=1}^{N}$. The density model can be either a set of discrete probabilities for quantized observations

$$b_i(\boldsymbol{O}_t) = \Pr(\boldsymbol{O}_t | q_t = i) \tag{15}$$

or a continuous probability density function

$$b_i(\boldsymbol{O}_t) = P(\boldsymbol{O}_t | q_t = i) \, . \tag{16}$$

Combining the definitions above and the starting probabilities of the states $\pi_i = \Pr(q_0 = i)$, an HMM can then be described fully by the triple $\lambda = (\pi, A, B)$ (Rabiner, 1989; Juang and Rabiner, 1991). Different HMMs can be conveniently connected by including the transitions between models to the matrix A. For example, in ASR one HMM is usually assigned to each phoneme, but it is also possible to use indeclinable words or other parts of words as the basic units.

Probability of the sequences. The joint probability of the state sequence q being generated by the HMM and the observation sequence $\boldsymbol{O}$ being generated by that state sequence is

$$\Pr(\boldsymbol{O}, q|\lambda) = \pi_{q0} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(\boldsymbol{O}_t) \, . \tag{17}$$

Since the generating state sequence is unknown the actual probability of the observation sequence for the model λ is

$$\Pr(\boldsymbol{O}|\lambda) = \sum_{q} \pi_{q0} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(\boldsymbol{O}_t) \, . \tag{18}$$

The probability (18) can be computed using the forward-backward procedure (Baum, 1972).

The dynamic programming by Viterbi algorithm (Forney, 1973) is commonly used to decode the most likely state sequence behind the observations by maximizing recursively the probability $\Pr(\boldsymbol{O}, q|\lambda)$ (17). The estimation of the HMM parameters using the maximum likelihood (ML) criterion, i.e. maximization of $\Pr(\boldsymbol{O}|\lambda)$ over λ, is done using the Baum-Welch algorithm (Baum and Petrie, 1966). Anyhow, a simpler ML training can be obtained by replacing the maximization of $\Pr(\boldsymbol{O}|\lambda)$ by the maximization of the likelihood of the most probable state sequence obtained by the Viterbi search. The optimal model is then

$$\bar{\lambda} = \arg\max_{\lambda} \max_{q} \Pr(\boldsymbol{O}, q|\lambda) \, . \tag{19}$$

This latter method is called the segmental K-means or the Viterbi training and it can be shown (Rabiner et al., 1986) to have the same asymptotic behavior as the Baum-Welch training, but with less numerical difficulties.

Output density models. The output probability density models aim at providing the conditional probabilities for the output features of the system associated with each state. The different ways to generate the output probabilities have been used to classify the HMMs.

A traditional way is to estimate a parametric density function (often a Gaussian) for each state (Baker, 1975; Jelinek, 1976). Because a simple function is clearly insufficient to represent the density of, for example, a phoneme, the model has been extended to a linear combination of PDFs for (16) (Liporace, 1982; Juang, 1985)

$$b_i(\boldsymbol{O}_t) = \sum_{m=1}^{M} c_{im} b_{im}(\boldsymbol{O}_t) \, , \tag{20}$$

where the mixture weights satisfy the conditions $c_{im} \geq 0$ and $\sum_{m=1}^{M} c_{im} = 1$ for all states i. The mixture components can be, for example, Gaussian PDFs $b_{im}(\boldsymbol{O}_t) \sim N(\mu_{im}, \Sigma_{im})$, where $\mu_{im} \in \mathcal{R}^D$ and $\Sigma_{im} \in \mathcal{R}^{D \times D}$ are the mean vector and the covariance matrix, respectively. HMMs which have parametric output density functions are generally called continuous density HMMs (CDHMMs) and those having mixture density functions mixture density HMMs (MDHMMs).

Another traditional way is to use vector quantization (VQ) (Makhoul et al., 1985) to transform the features into a set of output symbols and then estimate their discrete probability distribution in each state (Rabiner et al., 1983).

By combining the gains of the continuous and discrete models, good results have been obtained by a class of methods called semi-continuous HMMs (SCHMMs) (Huang and Jack, 1989; Paul, 1989; Bellegarda and Nahamoo, 1990) and multi-codebook vector quantization HMMs (Segura et al., 1994; Peinado et al., 1994). In SCHMMs a single set of Gaussians is shared between all states of all HMMs so that it is not necessary to train individual sets of Gaussians for each state. By forming a large set of Gaussians, which can be regarded actually as a codebook of Gaussian densities, the number of the mixtures used for each state can be much larger without increasing the total number of parameters to be estimated. Also the covariance matrices are often shared between the Gaussians as a trade-off to be able to increase the size of the codebook. In practice, SCHMMs are easier to train than CDHMMs, because the training of the codebook can be separated from the segmentation and even be trained mostly with unlabeled data (e.g., see Publication 1).

The SCHMMs offer an extension to the discrete observation HMMs, since it enhances the VQ performance by smoothing the quantization effects with weighted average of the probabilities given by several nearest units instead of only the BMU. The weights are inversely proportional to the distance of the units. This weighting leads actually to density model family known as the kernel density estimators or Parzen estimators (Parzen, 1962) with suitably chosen kernel functions. In the broad sense, as applied in this work, all the SCHMMs and CDHMMs, where the density model consists of a finite set of weighted multivariate unimodal kernel functions, belong to the mixture density HMMs (MDHMMs).

An illustration of a MDHMM and its parameters is shown in Figure 1. The temporal structure for this phoneme model is a typical simple uni-directional chain without skips.

Publication 3 as several other recent studies, e.g. (Young and Woodland, 1994; Digalakis et al., 1996), suggest to improve the performance of SCHMMs by tying some parameters for several similar states. The problem in practice is that the finite training data cannot provide reliable weights for the whole Gaussian codebook for every HMM state. For large vocabulary recognition tasks the tying of parameters is often crucial for developing a required amount of context dependent phoneme models. A common choice for tying mixtures is to have a separate Gaussian codebook for each HMM, which can then be called as a phoneme-wise

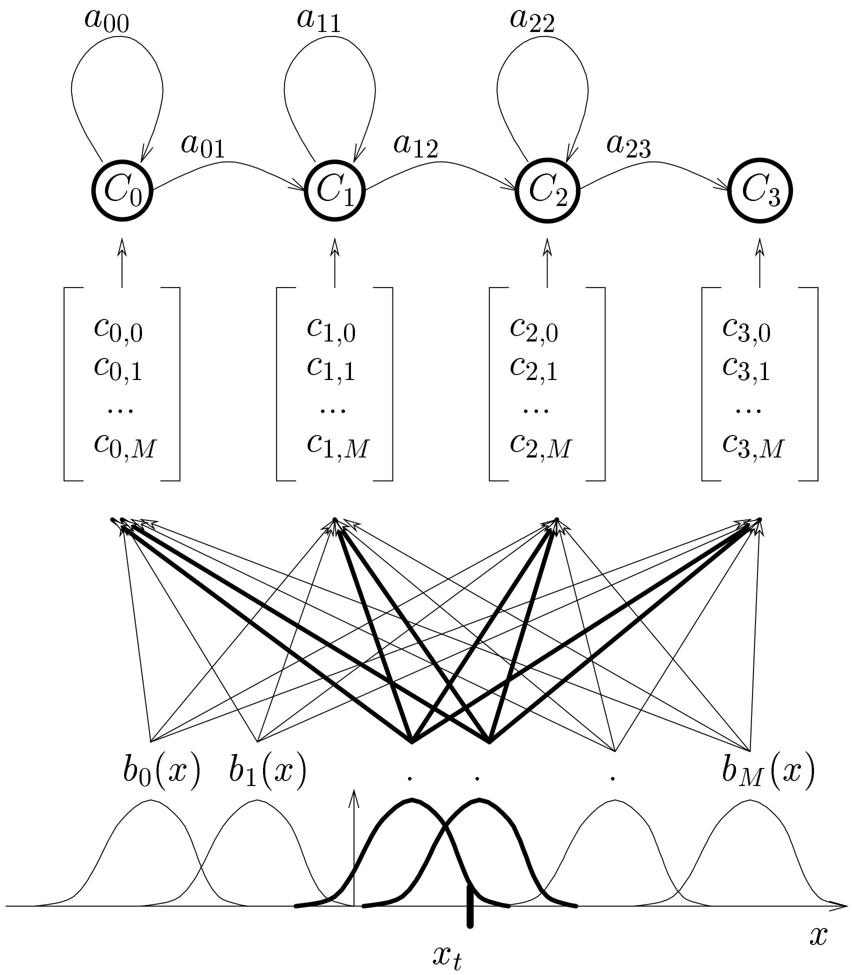

Figure 1: In phoneme-wise tied MDHMMs the same mixtures are used for the states representing the same phoneme. The model is defined by the set of transition probabilities a_{ij}, mixture weights c_{ik} and mixture densities $b_k(x)$. The output probability of state C_i at time t is computed as weighted average of the values provided by the mixture Gaussians for the current observation vector x_t. The observed feature vector is here a scalar only for illustrative purposes.

tied MDHMMs. In Publication 3 there is a brief comparison of different HMM classes.

HMM/ANN hybrids. The rapid development both in the computer hardware and in the neural computation methods has made it possible to apply many attractive density models for the HMMs to constitute hybrid HMM/ANN recognition systems (Lippmann, 1989; Morgan and Bourlard, 1995). The densities can be approximated by, for example, multilayer perceptrons (MLPs) (Bourlard and Wellekens, 1990; Franzini et al., 1990; Renals et al., 1994), time delay neural networks (TDNNs) (Dugast et al., 1994) or radial basis function networks (RBFs) (Huang and Lippmann, 1991; Singer and Lippmann, 1992; Renals et al., 1994).

The Gaussian RBFs are actually very close to Gaussian kernel density estimators; the difference is only in the way they are trained (Renals et al., 1991; Renals et al., 1992). The ANNs can also be used for subtasks in the density modeling, as for generating the VQ codebook by LVQ (Iwamida et al., 1990; Torkkola et al., 1991; Makino et al., 1992) or smoothing the VQ weights parameters by SOM (Zhao and Rowden, 1991; Monte, 1992; Kim et al., 1994).

In this thesis the ANNs (SOM and LVQ) are applied to assist in the training of the MDHMMs to develop methods that provide low error rates with a tractable amount of computations. The applications to the density initialization is discussed in Publications 1 and 3, to the actual training of the HMMs in Publications 4 and 5 as an enhancement of the conventional Viterbi training, and to the corrective tuning in Publication 2. Publication 6 compares then the results of some options for the total training process. Some of the main alternative HMM training methods have been presented, for example in (Kurimo, 1994) and (Morgan and Bourlard, 1995).

2.3.2 Limitations and gains

In theoretical considerations it seem to exist several limitations of the applicability of HMMs for the phoneme modeling. First, the assumption of phonemes as the invariable speech units is insufficient. Comparison of the same phonemes pronounced in different utterances reveals the well-known coarticulation effect, i.e. the articulatory apparatus adapts the current phoneme to the neighboring phonemes so that the realization is slightly variable. Some additional variation is due to the changes in the data collection conditions and the speaker. In order to take all variations into account, several different HMMs should actually be trained for each phoneme. Another possibility would be to apply a more complicated state structure that allows skips and branches instead of the simple uni-directional chain of Figure 1. However, the trainability and the scarcity of the training data restrict the use of very sophisticated models.

The basic assumptions of Markov models that only the present state affects the transition probabilities and that the successive observations are independent, which allow the simple probabilistic inference, do not hold accurately for speech. Both the production of the signal and the feature extraction are affected by the neighboring events so that the models can only be treated as approximations. The static density models are only approximations, as well, since all phonemes do not generate quasistationary signals. For using only a few Gaussian mixtures per phoneme there is yet a possibility for mismatch, because the variation around the centroids of the clusters in the input space is not inherently Gaussian. Such non-Gaussian clusters can be accurately modeled using only a sufficiently large number of mixtures.

Numerical difficulties arise from the wide dynamical range of the Gaussian densities. To prevent too low sequence probabilities special tricks need to be used. For example, the probabilities can be normalized at each time step or instead of multiplication in (17) the summation of logarithmized probabilities can be used (Rabiner, 1989). The joining of the discrete weights and transition probabilities to the density function values having much wider dynamical range may sometimes cause problems, as well.

Anyhow, despite their inaccuracy and theoretical incompleteness the mixture density HMMs seem to work very well in practice, and they are applied in almost all current ASR systems. The reason is perhaps that since their structure is so simple and mathematically feasible, the models can be extended easily. It can be shown (Niles and Silverman, 1990) that the HMM training is actually very close to some ANN methods. In practice, the complexity of the data and the theoretical limitations of the HMMs can be then partly overcome by brute force, i.e by increasing codebooks, input dimensions, and the number of models. As known for neural computation in general, very complex systems can be modeled in appropriate accuracy by connecting a sufficient amount of simple units (Gorin and Mammone, 1994). Special care must, however, be taken to control that there is enough training data and the data represents the task well. Otherwise the huge models will be over-fitted and will generalize poorly to independent test data. The heavily extended models are of the black-box type and do not necessarily offer any intelligible insights into the system.

2.4 Overview of speech recognition applications

Considerable progress has been made in ASR technology in the recent years (Young, 1996) and many new products have been launched, but still completely new approaches need to be developed before robust, general-purpose speech recognizers will be available (Bourlard, 1995). Current applications work only in relatively controlled environments and well-specified domains. For example, almost all applications are still restricted to the recognition of distinct words from a predefined vocabulary. Anyhow, such applications can be rather helpful in certain tasks.

By the vocabulary independent phoneme modeling as experimented in this work it is possible to recognize any spoken words, even those that have no meaning as long as they contain phonemes that can be trained for the models. The commercial production of the recognizers for unlimited vocabulary tasks have still been prevented by the excessive error rate and the need for careful speaker dependent training. However, some of the recognizer errors can be corrected by suitable post-processing of the phoneme sequences.

ASR performance. Some of the main external factors influencing the performance of the vocabulary dependent recognizers are gathered here to categorize the applications and understand the limits of the state-of-art ASR products.

1. The size and complexity of the vocabulary naturally affect the difficulty of the task.

2. Task based constraints that reduce the set of word candidates for each utterance are commonly used extensively to provide opportunities for large vocabulary speech recognizers. The average word branching factor is called *perplexity* that gives an idea of the expected number of words between which the recognizer must actually make its decision.

3. The data collecting conditions may require special signal processing tricks, if there is much noise in the environment or if the conditions are instable.

4. The speaking rate and the silence between successive words is one fundamental criterion to classify the ASR tasks. Speech used in natural conversations between humans is very difficult to recognize.

5. The match between the training and testing material is important. The more difference there is between the data, the more re-training or adaptation is required for good performance.

6. Speaker dependence divides the tasks into different categories. At one end good performance for one speaker can be obtained with a relatively few training data and simple models. At the other end the training of models which should perform adequately for anyone needs serious efforts.

State-of-art in ASR error rates. The following brief list exhibits the word error rates of representatives of some important categories (Rabiner, 1994) for vocabulary dependent recognizers. It is noteworthy that the evaluations (Rabiner, 1994) were applied in laboratory conditions and the performance in real-world conditions is usually much worse. The sentence error rate for the continuous speech tasks is much higher as well which makes these systems unsuitable for real-world applications. For large vocabulary tasks, it is customary to place rather strict restrictions to the grammar and syntax to keep the word matching procedure computationally feasible.

1. Isolated words, speaker independent, 129 airline words, word error rate 2.9%

2. Connected words, speaker dependent, airline reservations, finite state grammar with perplexity 4, 129 airline words, word error rate 0.1%

3. Continuous speech, speaker independent, airline travel information system (ATIS), finite state grammar with perplexity 12, 1800 words, word error rate 4.0%

4. Continuous speech, speaker independent, speech read from the *Wall Street Journal*, finite state grammar with perplexity 200, 20 000 words, word error rate 13.0%

Examples of existing applications. Previously the emphasis has been on the manufacturing sector, but in the last years the markets in telecommunications have shown exponential growth in sales figures, e.g. see (Rabiner, 1994). Telecommunications applications, as the following examples, usually reduce costs by replacing human attendants or provide new services previously impossible to implement.

1. Automation of operator assisted calls which need only to distinguish between the charging options
2. Automation of directory assistance based on spelled or spoken names
3. Voice dialing services either by name or number
4. Voice prompter to replace the touch tone
5. Information access services like sports scores, weather reports, and ticket booking

3 RESULTS

3.1 About the phoneme recognition experiments

Recognition error rate. A testing task used throughout the thesis deals with automatic speech recognition for unlimited vocabulary. Finnish speech data collected between 1991–1996 in the Laboratory of Computer and Information Science was used in the recognition tests. For each of over 20 speakers there are at least four recording sessions collected during a time span of normally a couple of months. In each session the speaker has uttered a word list consisting of over 300 different Finnish words. For the tests to compare the different algorithms and methods in this thesis, only a subset of speakers have been chosen, however, to keep the computational load tolerable. The speaker dependent recognition models are trained using three word sets and tested by the remaining one. To get a reliable view of the error rate, the leave-one-out principle is used in the testing and the average error rate of all the sets is reported. The error rate consists of the number of all phoneme errors (inserted, deleted, and changed phonemes) divided by the total number of phonemes. To gain statistical significance for the model comparisons, the error rates of all tests for all tested speakers are averaged. The rate of correct phonemes is sometimes given as well, but because it does not react to the phoneme insertion errors, it is not as illustrative as the phoneme error rate.

No post-processing of the results is applied in order to extract all the differences produced by the compared models. The post-processing can exploit language dependent syntax and point out uncommon phoneme combinations from the raw phoneme sequences. An optional post-processing module in the applied ASR system is based on the Dynamically Expanding Context algorithm (DEC)

(Kohonen, 1986a).The recognition of long phoneme versions like /AA/ from their short counterparts is a source of some frequent errors, as well. Here, the distinction is made using phoneme dependent duration limits learned iteratively during the model training. This simple separation does not take any context information into account. In Finnish the mismatches between the written and spoken format of words are quite exceptional, but these errors as well as some unmodeled rare phonemes increase the lowest obtainable value for the error rate.

Although the recognition test settings look similar in all publications included in the thesis, the error rate comparisons between the publications need to be done with special care. The speech database was revised in 1995 and the data collected after that is mainly used in the experiments, because there is now a broader variety of speakers available ranging from ASR researchers to novices. In some of the experiments the leave-one-out principle to average the results was abandoned in order to be able to test the methods with a larger amount of speakers. The set of training words was extended, as well, from the previous 311 to 350 by including some more uncommon words for better balance among the phoneme combinations. Since other more independent high quality Finnish databases suitable for similar experiments have unfortunately not been available, the data from 1991, which apply slightly different sampling rate than the new data, is still used to verify some of the main results.

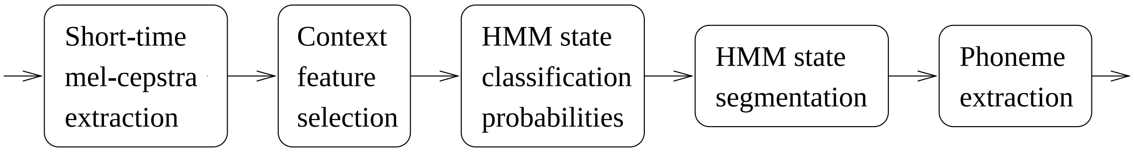

Figure 2: The main phases of the ASR by HMMs.

ASR system. For a brief review of the ASR system used in the experiments see the Figure 2. The recognition occurs in five successive phases. The contribution of this work restricts mainly to the middle phase, where the output density models of the HMM states are used to provide a likelihood for each state that the state would produce the current observed feature vector of the signal. This part is the one that consumes over 50 % of the total computational load of the recognition process. Some experiments have been made for Publications 4 and 6 with different extensions for the feature vectors, which fall to the second phase in Figure 2. The other three main phases of the recognition system are currently rather straightforward implementations of conventional algorithms with some minor practical tricks only to make the processing more effective.

The preprocessing of the acoustic signal is basically the same for all the experiments except that the sampling rate was increased for the new data from 12.8 kHz to 16 kHz, commonly available in the workstations. The acoustic features used throughout this work consists of the mel-cepstrum coefficients and the RMS-value of the signal. The basic feature vectors for the experiments are 20 component cepstra, but also extended feature vectors like averaged, concatenated, and delta cepstra were tested (Publication 4) and also a version where the concatenated vectors use only the 10–15 first coefficients (Publication 6).

The HMM structure has been subject to continuous development in this work. Some basic assumptions of the system remaining unchanged are the simple temporal structure of uni-directional chains without skips (see Figure 1), the principle of using one HMM for each of phoneme and the exclusion of some rare phonemes leaving totally 22 HMMs, which include only common Finnish phonemes and the silences directly before and after the word. The building blocks for the output density of the states have been Gaussians with a shared diagonal covariance matrix.

The most common types of mixture density HMMs are compared, for example, in Publication 3 in terms of the number of parameters, the recognition speed, and the achieved error rate. The structure called phoneme-wise tied mixture density HMM (PWMHMM) (Figure 1) performed best and was chosen to be the baseline method for both the further developments and the online prototype system. The current average phoneme error rate for the seven speakers used in Publication 6 is 5.3% measured on the 350-word test sets. By applying several successive training methods, a lower error rate (4.8%) can be achieved.

3.2 The applications of SOM for MDHMMs

3.2.1 Motivation

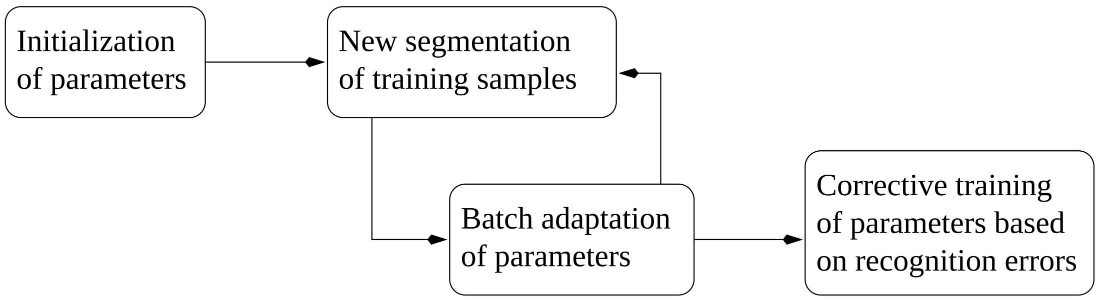

Figure 3: The main phases of HMM training based on the Viterbi segmentation.

The main phases of HMM training are shown in Figure 3. Conventionally the HMMs are initialized by using either manual segmentation of the training samples or a segmentation provided by another previously trained HMM. Equal duration of the states is normally assumed for the initial segmentation into states inside each phoneme. The parameters of each state are then obtained by finding a model that maximizes the likelihood of the associated data. The actual training loop (Figure 3) consists of alternating segmentation and likelihood maximization phases (Rabiner, 1989). When the HMMs are ready their performance can be tuned by corrective training (Bahl et al., 1988).

The investigations to apply SOM for the training of MDHMMs were motivated mainly by the following three ideas and needs:

1. A suitable algorithm is needed to initialize the centroids of the Gaussian mixtures used for the modeling of the output density functions of the HMM states. If the number of mixtures is large, the traditional approaches seem sometimes to fail in providing a good basis for convergence to high quality solutions, e.g. see Publication 6.
2. A smooth probability density representation is desired for dealing with the finite amount of training data to simultaneously maintain a high level of generalization and modeling accuracy for the independent test data.
3. For mixture densities with a large number of mixture components it is not reasonable to compute and sum the contributions of all components, to get an approximation for the density function. If the set of components, where the majority of the best matches are expected to occur, can be quickly separated, significant savings in the recognition speed are to be expected without a notable increase in the error rate.

3.2.2 Mixture density initialization by SOM

In Publication 6 some initialization methods of the MDHMMs are compared. Since the direct characteristics of the competing vector sets reveal only hints about the suitability for the recognition task (see Publication 3), the approach chosen in Publication 6 relies on the obtained recognition results after the codebooks are fed to the HMMs, and a typical HMM training is completed.

It can be shown (Juang, 1985) that the maximum likelihood HMM training will eventually lead to at least locally optimal parameter values, but it is the duty of the initialization to make sure that the obtained result will be as close as possible to the global optimum. In practice, the error function is so complicated due to the large training databases that it is not feasible to iterate the HMM training until the final convergence, since the progress will usually be very slow after the first couple of iterations. The ML criterion is actually not particularly effective to minimize the number of misclassifications, but it was used in the experiments to compare initializations, anyhow, because the advanced training algorithms like the segmental GPD (Juang and Katagiri, 1992) or the maximum mutual information (MMI) (Bahl et al., 1986) are even more vulnerable to bad initializations. Actually, the normal procedure for HMM training is to apply GPD or MMI only for models first trained by ML methods (in Figure 3 GPD or MMI would substitute the corrective training).

In addition to comparing the average test set error rates after a certain number of ML iterations, the performance of the differently initialized models is analyzed by taking into account the speed of convergence. In this case the speed of convergence refers to the number of training epochs it takes before the recognition result is close to some benchmark result. The benchmark result is used as a substitute of the optimal result simply because the determination, and even the definition, of the optimal result is somewhat troublesome. The benchmark result is determined here by finding out how low error rates can actually be achieved with the experimented ML method. This is done by selecting the model which gives the lowest average error rate after a long training session. The tested model is assumed to be near enough to the benchmark when the average recognition result is not significantly different (tested by the matched-pairs and the McNemar's statistical tests (Gillick and Cox, 1989)) from the benchmark.

The results in Publication 6 show that there are remarkable differences depending on the applied initialization methods. By the criteria described above, the SOM and LVQ initialization methods suit best to the MDHMMs. SOM was selected for the initialization to be used with further training method comparisons, because its fast implementation. For phoneme-wise tied MDHMMs (see Figure 1), the SOM initialization can be implemented very efficiently, since each mixture density codebook can be independently initialized.

3.2.3 Segmental SOM training

The MDHMM initialization experiments (Publication 6) mentioned in the previous section pointed out that the initial ordering of the codebooks obtained by the SOM affect positively in the HMM training. In SOM the ordering of units occurs due to the adaptation of the neighbors of the BMU. During the further HMM training, the ordering will be, however, destroyed quite severely because only the best matching density components are allowed to adapt in the likelihood maximization phases (see Figure 3). If the number of mixtures is small, this closer adaptation to the training data is inevitable in order to reach the maximal modeling accuracy, since the density approximation accuracy of the Gaussian kernels decays rapidly with growing distance from the centroid of the Gaussian. Anyhow, in a large codebook several kernels can often specialize to the same input cluster and then several nearby kernels could still be adapted simultaneously to maintain the ordering. A method called the *segmental SOM training* is presented in Publication 5 to incorporate the use of the neighborhoods into the MDHMM training.

The MDHMM training procedure for producing ordered density codebooks and maintaining the ordering throughout the training by the segmental SOM training occurs in the following steps: (This can be compared to Figure 3.)

1. Use pre-segmented training data to initialize the density codebooks by training one SOM for each phoneme
2. Initialize the state dependent mixture weights according to the partitioning of the data obtained by the SOM units. If necessary the unit covariances (kernel widths) can be initialized likewise.
3. Segment all the training data (no pre-segmentation required) by Viterbi decoding of the phoneme sequences using the obtained HMMs.
4. Adapt all the parameters by a batch SOM iteration (21).
5. Repeat from step 3. until the upper limit for the number of iterations or the lower limit for the average change of parameters is reached.

A batch iteration by the segmental SOM produces new mean vectors $\boldsymbol{\mu}_{jm}$ for Gaussian kernels $N(\mu_{jm}, \Sigma_{jm})$, $\forall m = 1, \ldots, M_j$ of mixture density codebook j, by computing the averages of the associated sample vectors $\boldsymbol{O}_t$:

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\sum_{t=1}^{T} \delta(q_t, j) h_{o,m} \boldsymbol{O}_t}{\sum_{t=1}^{T} \delta(q_t, j) h_{o,m}} , \tag{21}$$

where the indicator function $\delta(q_t, j) = 1$, if q_t (the decoded state for time t) is connected to the codebook j (otherwise $\delta(q_t, j) = 0$). In general, each state could use several codebooks and the same codebooks can be connected to several states. Here, the states in the same HMM use the same codebook (see Figure 1). The most probable correct state sequences q corresponding to the observation sequences $\boldsymbol{O}$ are determined by the current segmentation (see Figure 3). The equation (21) is essentially the same as (3). If individual covariance matrices would be needed for each Gaussian kernel, the adaptation formula for $\hat{\boldsymbol{\Sigma}}_{jm}$ corresponding to (21) could be obtained by substituting $\boldsymbol{O}_t$ in (21) by the matrix of the deviations from the mean vector $(\boldsymbol{O}_t - \boldsymbol{\mu}_{jm})(\boldsymbol{O}_t - \boldsymbol{\mu}_{jm})^T$.

The mixture weights that connect individual Gaussian kernels to the output density function of an HMM state (20) are set in the batch iteration to reflect the contribution of the kernels for the output density function of that state:

$$\hat{c}_{im} = \frac{\sum_{t=1}^{T} \delta(q_t, i) h_{o,m}}{\sum_{t=1}^{T} [\delta(q_t, i) \sum_{m=1}^{M_i} h_{o,m}]} , \tag{22}$$

where the indicator function $\delta(q_t, i) = 1$, here, if the state $q_t = i$ (otherwise $\delta(q_t, i) = 0$). The neighborhood function $h_{o,m} > 0$, if the unit m belongs to the neighborhood of the best-matching unit o of the current codebook (thus $h_{o,m}$ depends on t via the index o). From the different kinds of neighborhood functions $h_{o,m}$ (Kohonen, 1995), the simple bubble type is used here for simplicity. After each new adaptation of $\boldsymbol{\mu}_{jm}$s and c_{im}s the size of the adaptation neighborhood is decreased gradually until it is empty, and then the process continues by adapting only the parameters of the best-matching mixture for each training sample.

Convergence. The batch mode is used in the adaptation for faster convergence to acceptable results and easier control of the decrease of the learning rate and the neighborhood. For the training depending on the gradually improving Viterbi segmentation of the data, the batch training is a natural choice to ensure equal weighting of the training words. When the neighborhood is reduced to zero the segmental SOM equals the segmental K-means (Rabiner et al., 1986), which is the conventional ML training algorithm for HMMs.

For the analysis of the convergence of the suggested MDHMM training method mostly the same guidelines are valid as for the segmental K-means (Juang and Rabiner, 1990). The difference between the segmental K-means and segmental SOM is the same as between the normal K-means (MacQueen, 1967) and the normal batch SOM (Kohonen, 1995) as analyzed in (Luttrell, 1990), for example. If the SOM neighborhood size is small enough to ensure the increase of the likelihood of the model in the parameter adaptation steps the direction of the convergence can be expected to be close to that in the segmental K-means. However, in this HMM training experiments the neighborhood radius of the segmental SOM is gradually decreased to zero and after that only the parameters of the best matching mixture is adapted with steps identical to the segmental K-means.

The models trained by SOM are not optimized to discriminate between different models. LVQ is used for that purpose to optimize the classification boundaries for the areas in the observation sequences, in which the models behave inappropriately, by tuning the density functions (see Section 3.3).

Like the SOM training with zero neighborhood, also the LVQ training forces the codebook to fold and loose its smoothness. The large-scale structure of the codebook is not, however, entirely broken preserving still some potential for smoothing and density approximations. Figure 4 illustrates the breaking of the codebook structure by the zero neighborhood training. The figure shows the values of the mixture Gaussians of a codebook organized into a 14x10 SOM grid for one randomly selected input vector.

3.2.4 Characteristics of the mixture density approximation by SOM

Any bounded continuous PDF can be approximated with an arbitrary accuracy using a weighted sum of Gaussian densities, if the number of the Gaussians is large enough (Feller, 1966). If no prior knowledge of the functional form of the PDF is assumed, the mixture density estimation can be reduced to weighted kernel density estimation with, for example, Gaussian kernel functions. The idea of the self-organizing reduced kernel density estimator (RKDE) (Holmström and Hämäläinen, 1993; Hämäläinen, 1995) is to reduce the number of kernels by

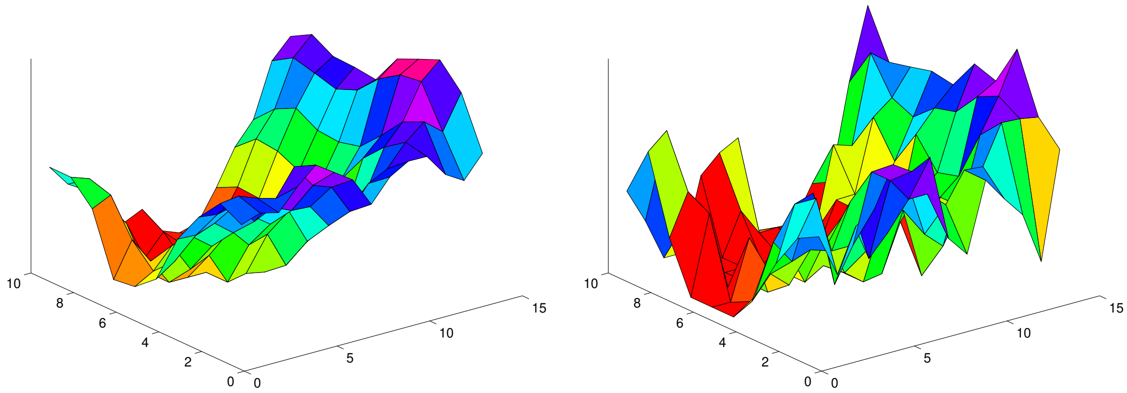

Figure 4: The responses of the individual mixture density components in the codebook for phoneme /A/ organized into 10x14 grid are plotted for one randomly selected input vector. The first plot (left) is the situation when the radius is decreased to one and the second is after training with zero neighborhood.

using the centroids of the weighted Gaussian kernel functions trained by SOM. The covariances of the kernels in kernel density estimators are normally reduced to the so-called smoothing parameters that assume the feature vector components to be independent and already optimally scaled.

Smoothing. Smoothing by SOMs has been commonly used for treating the unit weights of discrete observation density HMMs (Monte, 1992; Zhao and Rowden, 1991; Kim et al., 1994). Classical algorithms for the smoothing are the floor smoothing and the deleted interpolation (Rabiner, 1989; Rabiner et al., 1983; Jelinek and Mercer, 1980).

The adaptation of the neighborhood of the BMU in the self-organizing process offers a very natural way of smoothing the parameters by using additional samples falling nearby. So the group of units where a certain input will be mapped, if it is subject to some slight variation, are estimated by partly the same set of training samples ensuring the continuity of the PDF. The size of the neighborhood controls the amount of smoothing by affecting the extent of the overlap between the training samples falling for nearby units. The normal training concept of SOMs uses a large neighborhood in the beginning and gradually shrinks it as the training proceeds. This is well motivated by the smoothing point of view as well. Large neighborhood provides a high level of smoothing for every unit and then closer adaptation to the training data occurs in the areas well represented by the training data.

Tolerance for data defects. The smoothness of an output density of an HMM state is important for several reasons. The training data is finite and even with large and carefully balanced databases, there will almost always be input space areas with insufficient presentation. Training the codebook first using a large neighborhood exploits in a way some additional input samples for each unit. In addition to providing less random parameter values outside the main input data clusters, this may also allow to train larger codebooks by the same available training material, because the same samples could be used in training of larger amount of units.

In K-means based training it may sometimes happen that part of the available modeling capability is left unreachable, because some of the Gaussian kernels will be practically unused. In SOM, however, the first training epochs with large neighborhoods draw automatically most units near the important areas and enable thus their later use. The advantage of the maximal exploitation of the modeling capacity is revealed, when the training reaches the stage of closer adaptation. As the training proceeds the quality of the result does not improve like it should, if the initialization has been insufficient (e.g. see Publication 6).

Selective accuracy. High accuracy for the density modeling for MDHMMs is required only for the most likely states for good discrimination between the closest candidates. For the other states a quick approximation providing only the correct order of magnitude suffices.

For the mixture Gaussian probability estimates it is characteristic that each Gaussian provides a good accuracy only for samples near the centroid and that the total density estimate is strongly dominated by the few Gaussians close to the sample. To get more Gaussians to the areas where many input vectors fall, it is important that the point density of the Gaussian units reflect as well as possible the density of the input space. This ensures then the maximal resource utilization where it is most beneficial. The more units there are in the important areas the more robust and accurate PDF approximation can be expected. If there were only one unit, the small random effects due to its distance from the observation and its state-dependent weight may influence excessively to the accuracy of the PDF estimate. The exact form of the kernel functions will have less significance, as well, when the amount of participating kernels increases. This can be of importance for strongly non-Gaussian densities.

Compared to the codebooks trained without neighborhoods, e.g. trained by K-means, the accuracy obtained by the best-matching Gaussian can be worse, but the next $(K-1)$ best matches will be better. Figure 5 shows the relative difference of the distances of the nearest Gaussians for differently trained density codebooks. From all the curves having the SOM initialization, the later segmental K-means training will provide the BMU most accurately.

For following sections there are a couple of further notes from Figure 5. If the segmental LVQ3 training (Section 3.3.4) is applied after the SOM initialization the corresponding curve almost equals the 'som+skm' curve. This is not surprising, because both training methods rely on the smoothing obtained only by the initialization and because the discriminative adaptations are not expected to

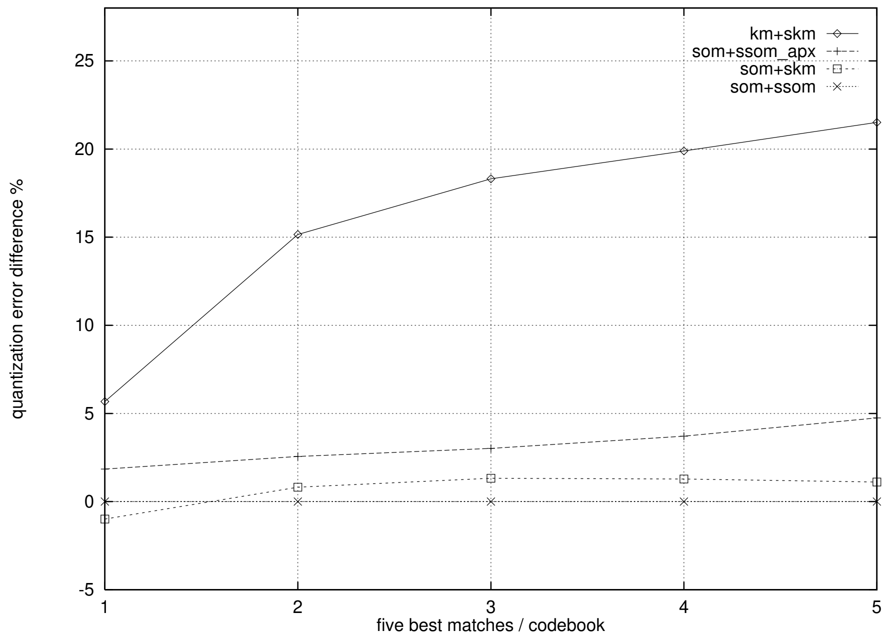

Figure 5: The average distances (here called quantization errors) of the five best matching units in the Gaussian codebook of the correct phoneme. The correct phoneme means the phoneme for which the feature vector is segmented at the end of the HMM training. The labels of the curves indicate the training methods that have been used. The initialization options K-means and SOM are indicated by 'km' and 'som', respectively. The segmental training options are correspondingly abbreviated as 'skm' and 'ssom'. The 'ssom_apx' means that the approximative topological K-best search (see Section 3.2.5) is used in the testing. To see the development of the relative differences of the compared training methods the average quantization error d of each of the top 5 units is plotted using the relative difference of errors to the default method $(d - d_0)/d_0$, here d_0 is from 'som+ssom'. The purpose of this figure is to show that adapting the neighborhoods will bring more units near the input vectors, in average.

affect the average distances. For Section 3.2.5 it is noteworthy that the quality of the approximative search on the SOM codebook is rather close to the original, and decreases slowly when going further from the best match. This indicates that the PDF approximation will not dramatically mess the total values of the output densities for the expected closest states.

Fast computation is necessary for the output densities of HMM states, because new density values need to be computed for every state when a new feature vector is fed to the system. Since the total density estimate is strongly dominated by the few Gaussians close to the sample, a good approximation can be obtained by using only the K best matching Gaussians of the total M in equation (20). Experiments to determine the relation between K and M for different HMMs are made in Publication 2. On situations of several candidates close to the sample, some of those K units can be replaced by other close-by candidates without a significant change in the total density value. These properties enable ways to generate some approximative K-best search methods (see the next section) without degrading much the modeling accuracy for codebooks smooth and dense around the sample vector as offered by the SOM training.

3.2.5 Examples of fast search methods in ordered codebooks

When the dimension of the feature vectors and the number of mixtures in the Gaussian codebooks are increased for better recognition accuracy, the density approximation made for each HMM state with each feature vector in the observation sequence will grow to a bottle-neck for online operation. As mentioned earlier the computations necessary for the density approximation can be much reduced, if a small subset where most of the K best matching density components are expected to be found, can be extracted from the large codebook. This can be done by clustering the Gaussians after the HMM training (Bocchieri, 1993), but by the SOM this is conveniently incorporated in the codebook training. Suggestions here for search that exploit the SOM structure include the topological and the tree K-best search. These methods also utilize the correlation between the successive feature vectors. There are, naturally, also other ways to speed up the density computation, for example, by using a simple coarse approximation for the states that do not potentially affect much the Viterbi search result. Methods to determine such codebooks and their approximations are experimented in Publication 5 and in (Lopez-Gonzalo and Hernandez-Gomez, 1993; Komori et al., 1995).

The topological K-best search for MDHMMs was tested and described in Publication 5 as an example of a way for utilizing the topology of an organized codebook of mixture densities for a fast approximative search algorithm for large codebooks. Briefly, the search begins by re-ranking the set of K best matches obtained from search regarding to the previous observation vector. Then the neighbors of the new best match are checked and if a new best match is found, the neighbors of it are checked as well. The local search process for the new set of K best matches similar to the fast winner search algorithms for SOM (Kohonen, 1996; Kohonen et al., 1996c; Lampinen and Oja, 1989) continues then until no more new best matches are found (see Figure 6). A complete search through the codebook is performed periodically to react for abrupt feature changes (Lopez-Gonzalo and Hernandez-Gomez, 1993).

In (Lampinen and Oja, 1989) the local search process starts from an initial guess based on the best match approximation obtained by the projections to the first few vector components (Friedman et al., 1975) and in (Zhao and Rowden, 1991) the search begins always from the center of the codebook. In (Kohonen, 1996; Kohonen et al., 1996c) a pointer to the previous best match is assigned for each training vector, which are used several times in the training. The use of similar pointer tables to bypass layers in a tree-search SOM was suggested by

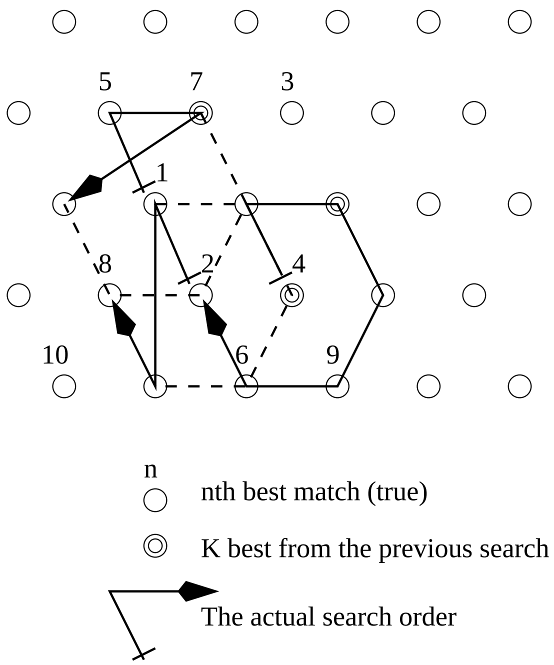

Figure 6: The topological search order for K-best search in SOM codebooks. In this example $K = 3$ and the local search will start from the immediate neighbors of the unit that will actually be the 4th best in the current rank. In this example the search will finally cover only 13 units and find the first and second best matches but miss the third.

(Koikkalainen, 1995). In Publication 5, only the set of K best matches for the previous observation vector is stored and it is assumed that because the successive observations in the sequence resemble each other, the new set of best matches can be found starting from the previous set. An important note is that the real K best matches are not necessarily found at every time instant, but rather a set of them that is related to the assumed best match. The presented method is expected to work well for the particular type of density estimation discussed in this work, but not for non-SOM codebooks or SOM applications in general.

Some important topics for fast search. The scanning of the density codebooks is done by using the partial distance computation (PDC). This is an elementary but efficient method for finding the BMU from a codebook. In the PDC the computation of the distance between two vectors aborts when the sum of the added component-wise differences exceeds a reference distance. Therefore the fastest execution time can be expected, if the rejections of the non-K-best matches can be determined as soon as possible. This can be assured on the average, if the candidates are ordered so that the most likely winners are checked first. For the same purpose the components of the feature vectors should be processed in the order of decreasing significance. With no special knowledge about the rank of the candidates except the continuous character of the signal, a good average performance can be expected, for example, if the candidates are scanned according to the distance in SOM topology from the expected winner. The better the topology of the input space is preserved in the SOM the better results this search order will give. Similarly, with no special knowledge about the rank of the components, it is best to organize them according to the decreasing variance, in general.

The frequency of the complete search affects the ability to react for fast changes in the signal characteristics and is, along with the number of the K best matches and the size of the basic search neighborhood, a controllable variable which can be used to increase either the accuracy or the speed of the search. It might become necessary to increase the search area and the frequency of the complete search, when the neighborhood used in the codebook training has decreased to zero for better accuracy and the ordering starts to disappear. However, the success of this approximate search is clearly dependent on the way the codebook is organized so that poor performance may occur, for example, if the best matches lie on the different edges of the grid. One way to reduce the risk of bad approximations might be to monitor the spread of the K best matches after the complete scan and start other local searches around the next best clusters, if necessary.

Some experimental comparisons. According to the ASR experiments reported in Publication 5 and recent checks with 80-dimensional context vectors (Kurimo, 1997), the obtained speed-up compared to the unordered complete search (optimized by PDC as well) is about 26–42 % depending on the codebook size and the feature vectors. The corresponding increase of the average number of errors is about 4–10 %. For codebooks trained with the segmental LVQ3 the same relative speed-up is obtained and the amount of errors increases by about 10 %. It seems that here the further training without neighborhood updates has not excessively damaged the topology necessary for speeding up the search.

To reflect the real significance of the speed-up, the relative improvements above were measured for the whole ASR system (Figure 2) except the mel-cepstrum computation (less than 10 % of the total time). The output density estimation which is the approximated part here, consumes originally about 50 % of the total time. The largest codebooks tested here have, however, contained only 140 units per phoneme for 80-dimensional inputs, so gains might be more significant, if really large SOMs would be required.

The tree-search SOM (Koikkalainen and Oja, 1990) suits well to the fast approximative search for large codebooks, because the tree structure offers $O(\log N)$ search complexity instead of the normal $O(N)$. The idea is to restrict the necessary search area of the SOM by the BMU selection in a smaller SOM. The structure can be extended into a tree with the smallest SOM at the top. This top layer is then the only SOM that needs to be thoroughly scanned in the process to find the best match from the largest SOM at the bottom (see Figure 7).

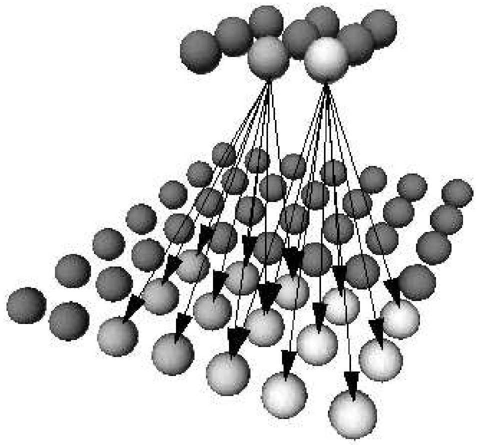

Figure 7: Two layers of the tree-search SOM. Here, each upper layer unit has a grid of 9 child units.

The idea of tree-search for mixture densities has been used without the SOM structure, e.g. in (Seide, 1995). The loss of accuracy in tree-search follows from the sequential branching decisions by which most of the units are eliminated from the individual inspection. The neighboring search areas for tree-search SOM can be made slightly overlapping for better cover and search accuracy. For K-best search the effect of the branching decisions is softened as well by expanding the search into the lower layer search areas associated with the rival best-matches from the upper layer.

The MDHMM training with the tree-search SOM can be made much like the training with the normal SOM. The codebook is first trained with pre-segmented samples proceeding from the smallest SOM layer on the top towards the largest layer at the bottom. The main exception from the normal SOM algorithm is that the neighborhoods are pre-specified for each layer. When the upper layers are ready they are fixed and used to determine the search areas and the adjusted set of units for the next layer. The Gaussian kernels are finally trained using the lowest layer units. In MDHMM density estimation the upper layers are just search tools that limit the search area and their match values do not (directly) affect the approximated probability value. The adjustments of the mixture densities necessary to optimize the MDHMM performance also affect only the bottom layer of the codebook.

The results of the experiments with MDHMMs reported in Publication 5 indicate that the tree-search SOM can be used as a slightly more inaccurate but faster substitute to the normal SOM codebook. In the comparison by ASR tests against the corresponding MDHMM with normal SOM codebooks, the tree-search SOM, in which the total recognition time decreased by 20 %, increased the average number of recognition errors by 14 %.

For multiprocessor implementations the utility of these search methods is probably small, because the parallelization of the complete search may shift the bottle-neck of the recognition time elsewhere in the process. Compared to the rapid capacity improvements in the general-purpose workstations, the significance of the described speed-ups concerns only situations, where some system must be made to work online on a certain computer. The idea was mainly to show some direct ways to exploit the ordered mapping generated automatically by the SOM, which does not depend as much on the data dimensionality or memory size as some traditional search methods do. Both the training by using the neighborhood smoothing and the topology based fast search methods are expected to scale well for larger dimensions and codebook sizes.

3.3 LVQ for MDHMMs

3.3.1 Motivation

When carefully trained by the maximum likelihood methods or using the SOMs, the resulting HMMs can be quite accurate in modeling the characteristics of the phonemes. However, the models are not optimized to discriminate between the states of different models. Some phonemes might resemble each other very much in certain contexts and thus the corresponding models may respond equally well. The selection of the highest likelihood model can then be based on some random details giving arbitrary decisions.

The LVQ was studied in order to develop a training method which would learn the essential information needed to discriminate the phonemes from the training samples in a simple adaptive manner. The method should tune the HMM parameters so that the correct models would stand out with as wide marginal as possible to avoid the risk of sequence misclassifications. Anyhow, any side effects like instable learning behavior or loss of generalization should be avoided. A possible loss of accuracy in the probability values for all competing sequences is not of concern, because the objective of this ASR subtask is only to find out the correct phonetic transcriptions regardless of the correctness of the probabilities.

3.3.2 LVQ for training vector quantization codebooks

The first idea concerning the integration of the LVQ into the MDHMM training scheme was to provide the HMM states with VQ codebooks which would already efficiently discriminate between the phoneme labels of individual feature vectors extracted from the pre-segmented speech samples. This means that LVQ was used in the initialization phase (see Figure 3). The aim was that the maximum likelihood training of the HMMs would then converge rapidly to a solution of high discrimination ability.

In (Kurimo and Torkkola, 1992b; Kurimo and Torkkola, 1992a) experiments were presented for the LVQ initialization of the CDHMMs. The output densities of the states were mixtures of 25 Gaussians that were unique for each state, but shared the same common diagonal covariance matrix. The results revealed that the LVQ codebooks tend to lead to models of lower error rates than the conventional K-means codebooks both in the Baum-Welch and Viterbi training (Rabiner, 1989). For the Viterbi training the LVQ initialization seemed also to be slightly better than the SOM initialization.

In Publication 1 the LVQ codebooks were generated for the SCHMMs where all states for every phoneme share the same large codebook of Gaussians. The difficulty in such an approach is the definition of which units represent the correct and incorrect phoneme classes since every unit may participate the density estimation for every state. The solution experimented in Publication 1 borrowed the idea of the data based majority voting used commonly in labeling of SOM units, e.g. see (Kohonen, 1995). Despite some possible problems caused by the variation of the number of labels per phoneme classes, the LVQ initialization experiments showed lower error rates compared to using only SOMs or K-means.

For phoneme-wise tied MDHMMs (Publication 3) there exist no problems with the mixture labeling since same mixtures are not used for different phonemes. The small phoneme-wise codebooks can be very easily trained first by SOM, and then for LVQ the codebooks can be concatenated into one. Anyhow, as concluded from the experiments in Publication 6, the SOM initialization alone seems to be sufficient. The discriminative training has more effect when embedded into the actual HMM training phase (see Figure 3) to constitute the new segmental LVQ3 method (see Section 3.3.4).

3.3.3 Corrective tuning based on LVQ2

For the fine tuning of the HMMs several methods have been suggested based on heuristics resembling the LVQ2 (Bahl et al., 1988; Mizuta and Nakajima, 1990; Juang and Katagiri, 1992; Galindo, 1995). The idea is first to train the HMMs normally and then improve their performance by some additional corrective iterations (last phase in Figure 3). The formulation of the training method is here more difficult than in the initialization phase where the labels of the training samples are predetermined. The segmentation of the training samples that determines to which state each data vector belongs, which is the basis of the LVQ training, is assumed to vary while the models gradually improve. The correct state segmentation of the data depends on the HMMs and vice versa. Thus the tuning of the model and the segmentation of the data must be developed together so that the tuning and the segmentation operations take turns and the process advances iteratively.

Since the segmentation of the data is not predefined but improves gradually while the training proceeds, it is difficult to determine the optimal size for the

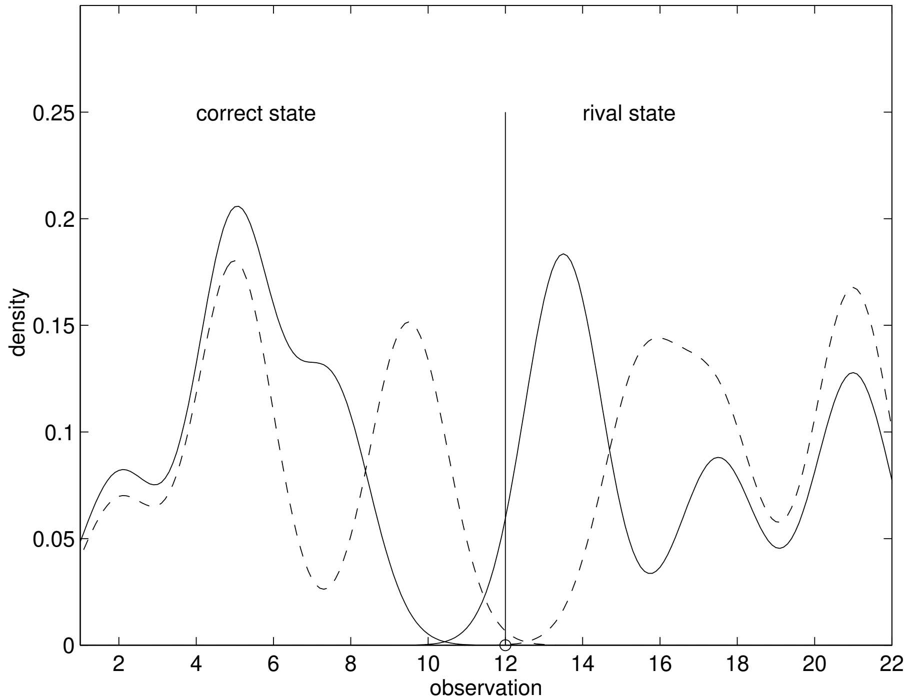

Figure 8: Adjusting the mixture densities of the competing states regarding to one input observation (here, a scalar value 12). The parameters to be modified are the centroids of the nearest Gaussian for the correct state and, if a rival HMM state causes a misrecognition (as in this illustration), also its corresponding centroid. The mixture weights of the modified mixtures are tuned respectively, but taking care of the normalization. In this simple three-mixture example for scalar observations the solid curves present old PDFs and the dashed curves the new PDFs resulting from the tuning operation.

stochastic learning steps. There are two conflicting aims, to weight more the gradient directions given by the later better segmentations, but still to force the process to converge at a reasonable speed. The learning from recognition errors by corrective tuning relates the models rather closely to the training data so that for finite data the long-run behavior of the learning process might not give the desired result. The solution tested in Publication 2 was to use an exponential decay for the learning rate and to check the change of the error rate on a separate data set after each epoch through the training set. Using then all the test runs for all the available speakers the suitable average amount of epochs was determined to be the limit after which no average improvement was detected. The corrective tuning method based on the LVQ2 learning law (9) is described in Publication 2. Figure 8 illustrates a simple example of the tuning process.

Contrary to most of the other corrective training methods, the LVQ2 based method used here does not relate the individual parameter adjustments to the exact extent of the difference of the resulting probabilities of the rival HMM state sequences. These whole word path probability differences follow from the straight derivation of the loss function, e.g. see (Rainton and Sagayama, 1992). Anyhow, here only the given learning rate and the local differences of the likelihood values provided by the rival mixtures determine the extent of the parameter adjustments. This is to avoid the risk of improper step sizes, which might follow from the occurrence of both severe and slight errors in the same word. Another difference is that the method applies the LVQ2-type learning law (Kohonen, 1990b) only to the detected actual misrecognition cases. Thus no tuning occurs, if the decoded phonemes match to the correct phoneme segmentation obtained by constraining the Viterbi search to the known correct transcription from the word list. This restriction is to avoid the risk of forcing some parameters out of the useful range by too frequent penalizations.

In Publications 1 and 2 brief experimental results are given on applying the corrective tuning for CDHMMs and SCHMMs after the normal maximum likelihood training by Baum-Welch or Viterbi training. In Publications 4 and 6 more thorough experiments are presented for MDHMMs by trying different input features and speech data. The conclusions are that, on the average, the recognition results have improved significantly by the corrective tuning. However, the new segmental LVQ3 method (see the next section) can give the same improvement faster and more conveniently by using only one training algorithm.

3.3.4 Segmental LVQ3 training

As apparent from the previous section, despite the good average results obtained by careful supervision, the corrective tuning may sometimes behave in undesired ways, either if incorrect adjustment step size, insufficient initialization or excessive discrimination is applied. To overcome the possible instability of the corrective LVQ2 tuning and to simplify the HMM training by making the additional tuning unnecessary, the segmental LVQ3 is presented as a substitute for separate successive maximum likelihood and corrective training phases. The idea is to apply a simple discriminative training method that would be robust enough to accept directly a quick HMM initialization, for example, based on SOM (see Section 3.2.2) so that the conventional ML training can be completely eliminated from the training scheme (see Figure 3). The new simple discriminative training should simultaneously be able to produce acceptable error rates in just a few training epochs without the need of a separate fine tuning phase.

Briefly, the segmental LVQ3 is a rather straightforward combination of the segmental K-means and the corrective training. Depending on the input the algorithm operates in either of two alternative options. For correctly recognized phonemes, the conventional likelihood maximization mode is used, but if the recognition fails, the likelihood of the correct phoneme is improved and the likelihood of the incorrect rival is lowered to increase the discrimination ability of the model. The conventional mode improves both the stability of the learning and the robustness against the initial values, because it prevents any excessive parameter penalization and directs the discrimination to the cases where it is really necessary.

The adjustments are made in batches consisting of the whole training set, similarly as in the segmental K-means. This removes the problem of the determination of the suitable learning rate schedule to properly take into account the changing segmentation and also improves the convergence speed. It is to be expected, however, that the result obtained in batch training may be somewhat more dependent on the initialization as that in stochastic training, because it may more easily get stuck to local minima. To overcome such possible convergence problems, the so-called Wegstein modification of the parameter adjustments could be applied in the same way as suggested for batch version of SOM (Kohonen, 1995). Since the segmental LVQ3 training method has provided good experimental results (see Publications 4 and 6), it seems to suit well for the segmental training as such, anyhow.

The exact adjustment laws of the segmental LVQ3 are presented in Publication 4. Publication 6 describes the same algorithm, but with a slightly revised notation.

3.3.5 Relations to other corrective training algorithms

The relations and mutual differences of the segmental LVQ3 and other corrective training algorithms for HMMs are discussed in Publication 6 with emphasis on the direct error corrective training algorithms (Bahl et al., 1988; Mizuta and Nakajima, 1990), and the application of GPD (Chou et al., 1992).

Because of some observed weaknesses in the traditional HMM training methods, which, however, have to a large extent been eliminated from the newer models, there has been suggested a wide range of other training algorithms, more or less related to the discriminative training. Some of them have also gained wide acceptance, such as the maximum mutual information (MMI) training (Bahl et al., 1986; Kapadia et al., 1993) and the maximum a posteriori probability (MAP) training (Lee et al., 1990; Gauvain and Lee, 1994; Huo et al., 1995).

Both of MMI and MAP attempt to increase the posteriori probability of the model sequence for the training data. The update rules can be presented as modifications of the present parameter values by an additional term obtained from the current iteration. Thus corrective training versions of both of the algorithms can also be derived. In (Ljolje et al., 1990) the MMI criterion is shown to emerge from a direct approximation of the empirical error rate. Compared to the simple discriminative training experimented with in this work, MMI and MAP involve considerably more effort in computing the parameter adjustments, but to demonstrate the practical gain no rigorous experimental study for characteristics of all the major methods at different conditions has yet been given, unfortunately.

4 CONCLUSIONS

Applications of two neural network paradigms, SOM and LVQ, for training the mixture density HMMs in ASR have been described and experimented. The main objective for the developments was to obtain models that are easy to train, cause less recognition errors, and are conveniently scalable to exploit developments in the computational resources.

The performed experiments indicate that SOM and LVQ principles are useful when the parameters of the rather complicated ASR system must be learned from the collected speech samples. The decrease in the average number of phoneme recognition errors for the tested speakers have been around 10 % in the applied test material: For example in Publication 6, the error rate decreased from 5.4 % to 4.8 %. The average error rate improvements are not very dramatic compared to the conventional training methods, but since the application has so much potential and importance as the ASR and the performance of the system is the main restrictive factor, even small improvements can have significant consequences.

For the future development of the HMMs the modeling of the duration of the states should perhaps be more explicit and the variation in the state output could perhaps be better treated by using application dependent knowledge rather than just increasing the number of units in the density models. For speech input beneficial extensions might follow from the use of diphone modeling or from allowing branches and skips in the HMM structure. Important future aspects for improvements of the ASR system include speaker independence or speaker adaptation, tolerance for unexpected disturbances and changes, and robustness to faster and more careless pronunciation. Also the applicability to other languages and the integration of context knowledge into the classification decisions would be interesting topics.

REFERENCES

Amari, S. (1967). A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, 16(3):299–307.

Bahl, L., Brown, P., de Souza, P., and 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)*, pages 49–52.

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

Baker, J. M. (1975). The DRAGON system – an overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):24–29.

Baldi, P. and Chauvin, Y. (1996). Hybrid modeling, HMM/NN architectures, and protein applications. *Neural Computation*, 8:1541–1561.

Bauer, H.-U., Der, R., and Herrmann, M. (1996). Controlling the magnification factor of self-organizing feature maps. *Neural Computation*, 8(4):757–771.

Baum, L. (1972). An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov processes. *Inequalities*, 3:1–8.

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

Bellegarda, J. and Nahamoo, D. (1990). Tied mixture continuous parameter modeling for speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(12):2033–2045.

Bocchieri, E. (1993). Vector quantization for the efficient computation of continuous density likelihoods. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 692–695.

Bourlard, H. (1995). Towards increasing speech recognition error rates. In *Proceedings of 4th European Conference on Speech Communication and Technology*, pages 883–894, Madrid, Spain.

Bourlard, H. and Wellekens, C. J. (1990). Links between Markov models and multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1167–1178.

Bradburn, D. (1989). Reducing transmission error effects using a self-organizing network. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 531–537, Piscataway, NJ. IEEE Service Center.

Chang, P.-C. and Juang, B.-H. (1992). Discriminative template training for dynamic programming speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 493–496, San Francisco,USA.

Cho, S.-B. and Kim, J. H. (1995). An HMM/MLP architecture for sequence recognition. *Neural Computation*, 7:358–369.

Chou, W., Juang, B., and 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)*, volume 1, pages 473–476, San Francisco,USA.

Cottrell, M. and Fort, J.-C. (1987). Étude d'un processus d'auto-organisation. *Annales de l'Institut Henri Poincaré*, 23(1):1–20. (in French).

Cottrell, M., Fort, J.-C., and Pages, G. (1997). Theoretical aspects of the SOM algorithm. In *Workshop on Self-Organizing Maps*, pages 246–267, Espoo, Finland.

Digalakis, V., Monaco, P., and Murveit, H. (1996). Genones: Generalized mixture tying in continuous hidden Markov model-based speech recognizers. *IEEE Transactions on Speech and Audio Processing*, 4(4):281–289.

Dugast, C., Devillers, L., and Aubert, X. (1994). Combining TDNN and HMM in a hybrid system for improved continuous-speech recognition. *IEEE Transactions on Speech and Audio Processing*, 2(1):217–223.

Erwin, E., Obermayer, K., and Schulten, K. (1992a). Self-organizing maps: Ordering, convergence properties and energy functions. *Biological Cybernetics*, 67(1):47–55.

Erwin, E., Obermayer, K., and Schulten, K. (1992b). Self-organizing maps: Stationary states, metastability and convergence rate. *Biological Cybernetics*, 67(1):35–45.

Feller, W. (1966). *An Introduction to Probability Theory and its Applications*, volume II. John Wiley & Sons Inc., New York.

Flanagan, J. A. (1996). Self-organisation in Kohonen's SOM. *Neural Networks*, 9(7):1185–1197.

Forney, G. D. (1973). The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Fort, J.-C. and Pages, G. (1996). About the Kohonen algorithm: Strong or weak self-organization? *Neural Networks*, 9(5):773–785.

Franzini, M., Lee, K.-F., and Waibel, A. (1990). Connectionist Viterbi training: a new hybrid method for continuous speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 425–428, Albuquerque, NM.

Friedman, J., Baskett, F., and Shustek, L. (1975). An algorithm for finding nearest neighbors. *IEEE Transactions on Computers*, 24:1000–1006.

Galindo, P. L. (1995). A competitive algorithm for training HMM for speech recognition. In *Proceedings of 4th European Conference on Speech Communication and Technology*, pages 2187–2190, Madrid, Spain.

Gauvain, J.-L. and Lee, C.-H. (1994). Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing*, 2(2):291–298.

Gillick, L. and 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)*, pages 532–535, Glasgow, Scotland.

Gorin, A. and Mammone, R. J. (1994). Introduction to the special issue on neural networks for speech processing. *IEEE Transactions on Speech and Audio Processing*, 2(1):113–114.

Hämäläinen, A. (1995). *Self-organizing Map and Reduced Kernel Density Estimation*. PhD thesis, University of Jyväskylä, Jyväskylä, Finland.

Holmström, L. and Hämäläinen, A. (1993). The self-organizing reduced kernel density estimator. In *Proceedings of the International Conference on Neural Networks (ICNN)*, pages 417–421, Piscataway, NJ. IEEE Service Center.

Huang, W. Y. and Lippmann, R. P. (1991). HMM speech recognition with neural net discrimination. In Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3*, pages 194–202. Morgan Kaufmann, San Mateo, CA.

Huang, X. and Jack, M. (1989). Semi-continuous hidden Markov models for speech signals. *Computer Speech and Language*, 3(3):239–252.

Huo, Q., Chan, C., and Lee, C.-H. (1995). Bayesian adaptive learning of the parameters of hidden Markov model for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 3(5):334–345.

Iwamida, H., Katagiri, S., McDermott, E., and 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)*, volume 1, pages 489–492.

Jalanko, M. (1980). *Studies of Learning Projective Methods in Automatic Speech Recognition*. PhD thesis, Helsinki University of Technology, Espoo, Finland.

Jelinek, F. (1976). Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–536.

Jelinek, F. and Mercer, R. (1980). Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of an International Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands. North-Holland.

Juang, B.-H. (1985). Maximum likelihood estimation for mixture multivariate stochastic observation of Markov chains. *AT&T Technical Journal*, 64(6):1235–1249.

Juang, B.-H. and Katagiri, S. (1992). Discriminative learning for minimum error classification. *IEEE Transactions on Signal Processing*, 40(12):3043–3054.

Juang, B.-H. and 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(9):1639–1641.

Juang, B.-H. and Rabiner, L. R. (1991). Hidden Markov models for speech recognition. *Technometrics*, 33(3):251–272.

Kangas, J. (1995). Increasing the error tolerance in transmission of vector quantized images by self-organizing maps. In Fogelman-Soulié, F. and Gallinari, P., editors, *Proceedings of ICANN'95, International Conference on Artificial Neural Networks*, volume 1, pages 287–291. EC2 et Cie.

Kapadia, S., Valtchev, V., and Young, S. (1993). MMI training for continuous phoneme recognition on the TIMIT database. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 491–494.

Kaski, S. (1996). *Data Exploration Using Self-Organizing Maps*. PhD thesis, Helsinki University of Technology, Espoo, Finland.

Kasslin, M., Kangas, J., and Simula, O. (1992). Process state monitoring using self-organizing maps. In Aleksander, I. and Taylor, J., editors, *Artificial Neural Networks, 2*, volume I, pages 1532–1534, Amsterdam, Netherlands. North-Holland.

Katagiri, S. and Lee, C.-H. (1993). A new hybrid algorithm for speech recognition based on HMM segmentation and learning vector quantization. *IEEE Transactions on Speech and Audio Processing*, 1(4):421–430.

Katagiri, S., Lee, C.-H., and 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*, pages 298–308, Princeton, New Jersey, USA.

Kim, D.-S., Lee, S.-Y., Han, M.-S., Lee, C.-H., Park, J.-G., and Suh, S.-W. (1994). Multi-dimensional 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*, pages 541–542, Iizuka, Japan. Fuzzy Logic Systems Institute.

Kohonen, T. (1982). Clustering, taxonomy, and topological maps of patterns. In *Proc. 6ICPR, Int. Conf. on Pattern Recognition*, pages 114–128, Washington, DC. IEEE Computer Soc. Press.

Kohonen, T. (1986a). Dynamically expanding context, with application to the correction of symbol strings in recognition of continuous speech. In *Proceedings of the 8th International Conference on Pattern Recognition*, pages 1148–1151, Paris, France.

Kohonen, T. (1986b). Learning vector quantization for pattern recognition. Report TKK-F-A601, Helsinki University of Technology, Espoo, Finland.

Kohonen, T. (1990a). Improved versions of learning vector quantization. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 1, pages 545–550, San Diego, California.

Kohonen, T. (1990b). The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480.

Kohonen, T. (1991). Workstation-based phonetic typewriter. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 279–288, Princeton, New Jersey, USA.

Kohonen, T. (1992). New developments of learning vector quantization and the self-organizing map. In *Symposium on Neural Networks; Alliances and Perspectives in Senri (SYNAPSE'92)*, Osaka, Japan. Senri Int. Information Institute.

Kohonen, T. (1993). Things you haven't heard about the self-organizing map. In *Proceedings of the International Conference on Neural Networks (ICNN)*, pages 1147–1156, Piscataway, NJ. IEEE Service Center.

Kohonen, T. (1995). *Self-Organizing Maps*. Springer, Berlin.

Kohonen, T. (1996). The speedy SOM. Technical Report A33, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland.

Kohonen, T., Barna, G., and Chrisley, R. (1988). Statistical pattern recognition with neural networks: Benchmarking studies. In *Proceedings of the International Conference on Neural Networks (ICNN)*, volume I, pages 61–68, Los Alamitos, CA. IEEE Computer Soc. Press.

Kohonen, T., Hynninen, J., Kangas, J., and Laaksonen, J. (1996a). SOM_PAK: the self-organizing map programming package. Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland. Program package available via WWW at URL http://nucleus.hut.fi/nnrc/som_pak.

Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J., and Torkkola, K. (1996b). LVQ_PAK: the learning vector quantization programming package. Report A30, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland. Program package available via WWW at URL http://nucleus.hut.fi/nnrc/lvq_pak.

Kohonen, T., Kaski, S., Lagus, K., and Honkela, T. (1996c). Very large two-level SOM for the browsing of newsgroups. In von der Malsburg, C., von Seelen, W., Vorbrüggen, J. C., and Sendhoff, B., editors, *Proceedings of ICANN96, International Conference on Artificial Neural Networks, Bochum, Germany, July 16-19, 1996*, Lecture Notes in Computer Science, vol. 1112, pages 269–274. Springer, Berlin.

Kohonen, T., Oja, E., Simula, O., Visa, A., and Kangas, J. (1996d). Engineering application of the self-organizing map. *Proceedings of the IEEE*, 84(10):1358–1384.

Koikkalainen, P. (1995). Fast deterministic self-organizing maps. In Fogelman-Soulié, F. and Gallinari, P., editors, *Proceedings of ICANN'95, International Conference on Artificial Neural Networks Paris, France, 9-13 October 1995*, volume 2, pages 63–68. EC2 et Cie.

Koikkalainen, P. and Oja, E. (1990). Self-organizing hierarchical feature maps. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume II, pages 279–284, Piscataway, NJ. IEEE Service Center.

Komori, T. and Katagiri, S. (1992). GPD training of dynamic programming-based speech recognizers. *J. Acoustical Society of Japan*, 13(6):341–349.

Komori, Y., Yamada, M., Yamamoto, H., and Ohora, Y. (1995). An efficient output probability computation for continuous HMM using rough and detailed models. In *Proceedings of 4th European Conference on Speech Communication and Technology*, pages 1087–1090, Madrid, Spain.

Kurimo, M. (1992). Combinations of adaptive vector quantization methods and hidden Markov models in speech recognition. Master's thesis, Helsinki University of Technology, Espoo, Finland. (in Finnish).

Kurimo, M. (1994). Application of learning vector quantization and self-organizing maps for training continuous density and semi-continuous Markov models. Licentiate's Thesis, Helsinki University of Technology, Espoo, Finland.

Kurimo, M. (1997). SOM based density function approximation for mixture density HMMs. In *Workshop on Self-Organizing Maps*, pages 8–13, Espoo, Finland.

Kurimo, M. and Torkkola, K. (1992a). Application of SOMs and LVQ in training continuous density hidden Markov models. In *Proceedings of the International Conference on Spoken Language Processing*, volume 1, pages 543–546, Banff, Canada.

Kurimo, M. and Torkkola, K. (1992b). Combining LVQ with continuous density hidden Markov models in speech recognition. In *Proceedings of the SPIE's Conference on Neural and Stochastic Methods in Image and Signal Processing*, volume 1766, pages 726–734, San Diego, USA.

Lampinen, J. and Oja, E. (1989). Fast self-organization by the probing algorithm. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume II, pages 503–507, Piscataway, NJ. IEEE Service Center.

Lee, C.-H., Lin, C.-H., and Juang, B.-H. (1990). A study on speaker adaptation of continuous density HMM parameters. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 145–148.

Liporace, L. A. (1982). Maximum likelihood estimation for multivariate observations of Markov sources. *IEEE Transactions on Information Theory*, 28(5):729–734.

Lippmann, R. (1989). Review of neural networks for speech recognition. *Neural Computation*, 1(1):1–38.

Ljolje, A., Ephraim, Y., and Rabiner, L. (1990). Estimation of hidden Markov model parameters by minimizing empirical error rate. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 709–712.

Lopez-Gonzalo, E. and Hernandez-Gomez, L. A. (1993). Fast vector quantization using neural maps for CELP at 2400 bps. In *Proceedings of 3rd European Conference on Speech Communication and Technology*, volume 1, pages 55–58, Berlin, Germany.

Luttrell, S. P. (1990). Derivation of a class of training algorithms. *IEEE Trans. on Neural Networks*, 1(2):229–232.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of Fifth Berkeley Symposium on Math. Statist. and Prob.*, pages 281–297.

Makhoul, J., Roucos, S., and Gish, H. (1985). Vector quantization in speech coding. *Proceedings of IEEE*, 73(11):1551–1588.

Makino, S., Endo, M., Sone, T., and Kido, K. (1992). Recognition of phonemes in continuous speech using a modified LVQ2 method. *J. Acoustical Society of Japan*, 13(6):351–360.

Mäntysalo, J. (1992). Some experiments on LVQ-based speech recognition with high-dimensional context vectors. Master's thesis, Helsinki University of Technology. (in Finnish).

McDermott, E. (1990). LVQ3 for phoneme recognition. In *Proc. Spring Meet. Acoust. Soc. Jpn.*, pages 151–152.

McDermott, E. and Katagiri, S. (1994). Prototype-based minimum classification error/ generalized probabilistic descent training for various speech units. *Computer Speech and Language*, 8(4):351–368.

Mizuta, S. and Nakajima, K. (1990). An optimal discriminative training method for continuous mixture density HMMs. In *Proceedings of the International Conference on Spoken Language Processing*, volume 1, pages 245–248, Kobe,Japan.

Monte, E. (1992). Smoothing HMMs by means of a SOM. In *Proceedings of 1992 International Conference on Spoken Language Processing*, volume 1, pages 535–537, Banff, Canada.

Morgan, N. and Bourlard, H. (1995). Neural networks for statistical recognition of continuous speech. *Proceedings of the IEEE*, 83(5):742–770.

Mulier, F. and Cherkassky, V. (1995). Self-organization as an iterative kernel smoothing process. *Neural Computation*, 7(6):1165–1177.

Niles, L. T. and 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)*, volume 1, pages 417–420.

Nilsson, N. (1965). *Learning Machines*. McGraw-Hill, New York.

Parzen, E. (1962). On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076.

Paul, D. B. (1989). The Lincoln robust continuous speech recognizer. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 449–452, Glasgow, Scotland.

Peinado, A. M., Segura, J. C., Rubio, A. J., and 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)*, volume 1, pages 61–64.

Rabiner, L., Levinson, S., and Sondhi, M. (1983). On the application of vector quantization and hidden Markov models to speaker-independent isolated word recognition. *Bell System Technical Journal*, 62:1075–1105.

Rabiner, L., Wilpon, J., and Juang, B. (1986). A segmental K-means training procedure for connected word recognition. *AT&T Technical Journal*, 64:21–40.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rabiner, L. R. (1994). Applications of voice processing to telecommunications. *Proceedings of the IEEE*, 82(2):199–228.

Rainton, D. and Sagayama, S. (1992). Appropriate error criterion selection for continuous speech HMM minimum error training. In *Proceedings of the International Conference on Spoken Language Processing*, volume 1, pages 233–236, Banff,Canada.

Renals, S., Morgan, N., and Bourlard, H. (1991). Probability estimation by feed-forward networks in continuous speech recognition. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 309–318, Princeton, New Jersey, USA.

Renals, S., Morgan, N., Bourlard, H., Cohen, M., and Franco, H. (1994). Connectionist probability estimators in HMM speech recognition. *IEEE Transactions on Speech and Audio Processing*, 2(1):161–174.

Renals, S., Morgan, N., Bourlard, H., Franco, H., and Cohen, M. (1992). Connectionist optimization of tied mixture hidden Markov models. In Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3*, pages 167–174. Morgan Kaufmann, San Mateo, CA.

Ritter, H. (1989). Asymptotic level density for a class of vector quantization processes. Report A9, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland.

Ritter, H. (1991). Asymptotic level density for a class of vector quantization processes. *IEEE Transactions on Neural Networks*, 2(1):173–175.

Ritter, H. and Schulten, K. (1986). On the stationary state of Kohonen's self-organizing sensory mapping. *Biological Cybernetics*, 54(1):99–106.

Ritter, H. and Schulten, K. (1988). Convergence properties of Kohonen's topology preserving maps: fluctuations, stability, and dimension selection. *Biological Cybernetics*, 60(1):59–71.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407.

Segura, J., Rubio, A., Peinado, A., Garcia, P., and Roman, R. (1994). Multiple VQ hidden Markov modelling for speech recognition. *Speech Communication*, 14:163–170.

Seide, F. (1995). Fast likelihood computation for continuous mixture densities using a tree-based nearest neighbor search. In *Proceedings of 4th European Conference on Speech Communication and Technology*, pages 1079–1082, Madrid, Spain.

Singer, E. and 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)*, volume 1, pages 629–632, San Francisco,USA.

Torkkola, K., Kangas, J., Utela, P., Kaski, S., Kokkonen, M., Kurimo, M., and Kohonen, T. (1991). Status report of the Finnish phonetic typewriter project. In Kohonen, T., Mäkisara, K., Simula, O., and Kangas, J., editors, *Artificial Neural Networks*, volume I, pages 771–776, Amsterdam, Netherlands. North-Holland.

Utela, P. (1992). Phoneme recognition with discrete density Markov models. Master's thesis, Helsinki University of Technology. (in Finnish).

Young, S. (1996). A review of large-vocabulary continuous-speech recognition. *IEEE Signal Processing Magazine*, pages 45–57.

Young, S. and Woodland, P. (1994). State clustering in HMM based speech recognition. *Computer Speech and Language*, 8(4):369–384.

Zador, P. L. (1982). Asymptotic quantization error of continuous signals and the quantization dimension. *IEEE Transactions on Information Theory*, IT-28(2):139–148.

Zhao, Z. and Rowden, C. (1991). Application of Kohonen self-organising feature maps to smoothing parameters of hidden Markov models for speech recognition. In *Second Int. Conf. on Artificial Neural Networks (Conf. Publ. No.349)*, pages 175–179, London, UK. IEE.

SUMMARY OF THE PUBLICATIONS

In **Publication 1** a complete HMM training method is described, where the codebooks formed with SOM and LVQ are used to initialize the state output densities of semi-continuous HMMs (SCHMMs). A single large SOM is first trained using unsegmented speech data. Then the codebook is labeled using phoneme labels selected by majority voting in segmented speech samples. The segmented speech data is used for LVQ training to improve the phoneme classification accuracy. The trained codebook is then used as an initialization of the mixture densities of the SCHMMs and the final parameters are trained by the Baum-Welch re-estimation. The experimental results show that using the LVQ for the initialization provides better final error rates on the average. The experimented error corrective tuning method decreases the average error rates for some HMM configurations as well.

Publication 2 presents descriptions and comparisons of the versions of the LVQ-based corrective tuning method for both the CDHMMs and SCHMMs. According to the experiments the best version was to use LVQ2 learning law with respect to the segmentation based on the most probable path visiting the correct phonemes in the correct order. Experimental results are also given for varying the number of the best-matching mixture Gaussian components selected to approximate the state output probabilities. The optimal number as a fraction of the total number of Gaussians is proposed for both the CDHMMs and SCHMMs using the average error rate.

Publication 3 describes a version of the Gaussian MDHMMs, where the density codebooks are tied phoneme-wise, and motivates the construction by both training and performance aspects. The application of LVQ to the codebook training is presented and some analysis of the codebooks are provided based on different classification statistics to see the effects of SOM and LVQ training. The proposed models are compared to both CDHMMs and SCHMMs with respect to the number of parameters, the recognition time and the average error rate.

In **Publication 4** the segmental LVQ3 training method is presented for the MDHMMs. The equations are given to compute the batch adjustments for the mixture weights and mean vectors. It is shown experimentally that the algorithm gives at least as good results as the previously used combination of the maximum likelihood training and the corrective tuning, but it requires considerably fewer training epochs. Experiments are also reported on how the recognition accuracy of the current system with optimized configuration for online operation could be improved by increasing the number of Gaussians in the mixtures and the dimensionality of the features by applying context vectors.

The objective of **Publication 5** is to evaluate ways to improve the recognition speed without losing much in the accuracy. The methods exploit the topological structure in the density codebooks obtained by using training methods based on the SOM. In addition to the advanced and approximative search methods based on the normal SOM, also incorporation of the tree-search SOM to the

MDHMMs is studied. To maintain the topology for the MDHMMs a segmental SOM algorithm is described. The effects of the search boosting methods are compared using the average error rate on the testing database and the average recognition speed statistics.

Publication 6 includes a short introduction and motivation to the development of the training methods for MDHMMs. The segmental LVQ3 is described and compared to the segmental K-means, the corrective LVQ2 tuning and the segmental GPD. The use of two statistical tests to compare the recognition results and measure the average training length requirements is presented. The baseline recognition system is described and experiments covering training modifications, speed enhancements and feature vector extensions are reported on two different databases.

The **contribution of the author** in Publication 5 was the development of the ideas, doing the experiments and writing the paper. The coauthor was responsible for the implementation of the tree-search SOM for the SOM_PAK environment and preliminary experiments to find out a suitable tree structure. The adaptation of the tree-search SOM to the MDHMM framework was done by the author. The HMM based ASR system used in all of the publications and the online prototype system was prepared by the author following the guidelines of the long ASR traditions in the Laboratory of Information and Computer Science.

Finally, the following simplified list of the novelties of this thesis is given to distinguish the author's own contributions from the ideas derived elsewhere:

- Using SOM and LVQ to initialize the mixture densities for various HMM structures.

- Development of the segmental MDHMM training algorithms based on SOM and LVQ3 to benefit from the preserved local topology and phoneme differentiation in the Viterbi training, respectively.

- Development of a simple corrective tuning method based on LVQ2 for the mixture densities.

- Using the SOM codebook structures to enable fast density approximations for the states of the MDHMMs.

- Development of the phoneme-wise tied mixture density codebooks with a large pool of identically shaped Gaussian mixtures.

- Evaluation of the proposed ASR system with MDHMM training assisted by ANN methods for the Finnish speech data.