HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Science

Application of Learning Vector Quantization and Self-Organizing Maps for training continuous density and semi-continuous Markov models

Licentiate's Thesis

Mikko Kurimo

Espoo 1994

HELSINKI UNIVERSITY OF TECHNOLOGY **ABSTRACT OF THE LICENTIATE'S THESIS**

Author:	Mikko Kurimo		
Name of the thesis:	Application of Learning Vector Quantization and Self-Organizing Maps for training continuous density and semi-continuous Markov models		
Date:	February 1, 1994	**Number of pages:**	86
Department:	Faculty of Information Technology Department of Computer Science	**Professorship:** Tik-61 Computer and Information Sciences	
Supervisor:	Professor Teuvo Kohonen		

Experiments on variations of continuous density and semi-continuous hidden Markov models (CDHMM and SCHMM) as a part of a speech recognition system are performed. The work is oriented to the testing of different combinations of neural network methods developed in the Laboratory of Information and Computer Science at Helsinki University of Technology and more conventional statistical parameter estimation methods based on the maximum likelihood (ML) principle.

Self-Organizing Maps (SOM) and Learning Vector Quantization (LVQ) are applied into the initialization of the mean vectors of the mixture Gaussian densities for CDHMMs and SCHMMs to reduce the amount of required ML estimation and achieve more discriminative phoneme models. Experiments are also made with a LVQ-based corrective tuning method by which the HMMs can be further enhanced for lower recognition error rates.

Some approximations for the computationally complex determination of the continuous output densities for the CDHMMs and SCHMMs are also tested. The suggested approximations reduce the number of covariance parameters in the multivariate mixture Gaussian densities and the number of mixtures actually used in the observation probability computations. The lowest average phoneme recognition error rates achieved by the novel combinations of training methods were about 5.6 %.

PREFACE

The work presented in this thesis has been carried out at the Laboratory of Information and Computer Science in the Faculty of Information Technology of the Helsinki University of Technology.

I am greatly indebted to Professor Teuvo Kohonen for valuable contributions to my studies and providing excellent facilities and financial support for this research. I wish to thank Dr. Kari Torkkola for useful discussions and guidance as well as all of the personnel of the laboratory for creating a comfortable and inspiring atmosphere.

The financial support of the Foundation for the Promotion of Technology is gratefully acknowledged.

Espoo February 1, 1994

Mikko Kurimo

LIST OF ABBREVIATIONS

SOM	Self-organizing map
VQ	Vector quantization
LVQ	Learning vector quantization
KM	K-means algorithm
HMM	Hidden Markov models
CDHMM	Continuous density HMM
DHMM	Discrete HMM
SCHMM	Semi-continuous HMM
BW	Baum-Welch algorithm
MLE	Maximum likelihood estimation
MEC	Minimum error classification
GPD	Generalized probabilistic descent
ODT	Optimal discriminative training
BMU	Best-matching unit

Contents

1 INTRODUCTION **9**

2 HIDDEN MARKOV MODELS **11**

2.1 General 11

2.2 HMM formulation 12

2.2.1 Markov chains 12

2.2.2 Hidden Markov model 12

2.2.3 The decoding of the state sequence 13

2.3 Continuous density HMMs 14

2.4 Discrete HMMs 15

2.5 Semi-continuous HMMs 16

2.5.1 The tying of mixture densities 16

2.5.2 Extension of the DHMMs 17

2.5.3 The general use of tying 18

2.6 Parameter estimation 18

2.6.1 Iterative methods 18

2.6.2 Selection of the objective 19

3 DENSITY INITIALIZATION BY SOM **20**

3.1 Initialization of HMMs 20

3.1.1 Importance 20

3.1.2 Objective 20

3.2 SOM algorithm 21

3.2.1 VQ 21

3.2.2 Learning algorithm 21

3.2.3 Quantization error vs. Topology preserving 22

3.2.4 Using different topologies and neighborhoods 22

3.2.5 Effects of the neighborhood to the density of the quantization levels 23

3.3 Some general properties of SOMs 24

3.3.1 Smoothing by using the neighborhoods 24

3.3.2 Applicability of unlabeled training data 26

3.3.3 Using the topology of SOM 28

3.3.4 Probability approximation by SOM 28

3.4 Using SOM for observation density initialization of HMMs 30

3.4.1 Motivation 30

3.4.2 Implementation issues 31

3.4.3 Analysis of mixture weights 32

4 DENSITY INITIALIZATION BY LVQ 35

4.1 Increasing discrimination between the models 35

4.2 LVQ 36

4.2.1 Minimization of the classification error 36

4.2.2 LVQ algorithms 36

4.2.3 Codebook initialization 38

4.2.4 Other variations 38

4.3 Using LVQ for density initialization of HMMs 39

4.3.1 Motivation 39

4.3.2 Implementation issues 39

5 PARAMETER ESTIMATION **41**

5.1 Maximum likelihood training 41

5.1.1 Baum-Welch reestimation 41

5.1.2 Viterbi training 43

5.1.3 Problems 45

5.2 Maximum mutual information training 46

5.3 Corrective training 47

5.3.1 Discrete output density 47

5.3.2 Continuous output density 48

5.4 Generalized probabilistic descent method 48

5.4.1 Definitions 48

5.4.2 Adjustment step 50

5.5 Minimum error classification training 50

5.5.1 Differentiating the cost function 51

6 CORRECTIVE TUNING BY LVQ **53**

6.1 Intuitive motivation 53

6.2 Description of the algorithm 54

6.2.1 Paths and segmentation 54

6.2.2 Density tuning 54

6.3 Connections to GPD and ODT 55

6.4 Implementation 57

7 PHONEME RECOGNITION EXPERIMENTS **60**

7.1 Recognition system 62

7.2 Experiments 64

7.2.1 Material 64

7.2.2 Matched-Pairs statistical test 64
7.2.3 Model configuration 65
7.2.4 Training combinations 69
7.2.5 Initialization tests 70
7.2.6 Recognition tests 72
7.2.7 Corrective tuning tests 74

8 CONCLUSIONS **76**

8.1 Final interpretations 76
8.2 Future considerations 77

Chapter 1

INTRODUCTION

The motivation for this work arose from the speech recognition experiments performed in the Laboratory of Information and Computer Science of Helsinki University of Technology with Hidden Markov Models and Learning Vector Quantization. The theoretical foundations of LVQ and Self-Organizing Maps, which have mainly been developed in this laboratory, were an important guideline in directing the experiments towards those of this work. The major experiments were conducted with LVQ-trained codebooks for discrete observation density HMMs (Torkkola, 1991; Utela, 1992; Mäntysalo, 1992) but the continuous density HMMs were implemented first as a reference model (Kurimo, 1991) and later as a smaller subproject after the success in using LVQ as a part of training scheme (Kurimo, 1992). The implementation of the semi-continuous HMMs with LVQ-training was a natural successor of the CDHMM project because being a kind of intermediate between DHMMs and CDHMMs it provoked a certain interest.

The first applications of hidden Markov modeling for speech recognition were developed in Carnegie Mellon University (Baker, 1975) and in IBM (Jelinek, 1976). Since the early 1980's the HMMs have been commonly studied, e. g. (Levinson, 1983; Juang, 1985; Liporace, 1982; Bahl et al., 1983), but only recently it has become widely accepted that to achieve the best recognition results application of some kind of discriminative training is necessary in addition to the popular maximum likelihood training.

Due to the lack of efficient analytical formulation of the discriminative training for HMMs, neural networks are used to provide hybrid models combining their classification power with the time-domain modeling capability of HMMs, e. g. (Bourlard & Wellekens, 1988; Franzini et al., 1990; Bengio et al., 1991). LVQ is readily applicable to the DHMMs and the reports of hybrid systems of DHMMs with LVQ-trained codebook (Iwamida et al., 1990a; Iwamida et al., 1990b; Torkkola et al., 1991a; Kohonen, 1991) reveal the evident success of such methods.

For CDHMMs there have been several training methods that share the idea of discriminativity but also the increased amount of computations like the maximum mutual information (Bahl et al., 1986) and maximum discrimination information (Epraim et al., 1989). Some very successful experiments have been performed with corrective training based on the principle that after first recognizing some samples with an initial model the model parameters can be further tuned depending on the success in the recognition task. These methods have been developed for both DHMMs (Bahl et al., 1988) and CDHMMs (Mizuta & Nakajima, 1990) but perhaps the lack of rigorous theory behind the estimation formulas inhibit some of their popularity.

The Generalized Probabilistic Descent method (Katagiri et al., 1991) is a significant attempt to define a theoretically justified and practically applicable discriminative training method for CDHMMs. From the GPD it is possible to derive simplified and efficient iteration formulas that behave as the LVQ2 (Komori & Katagiri, 1992; Kurimo, 1992; Rainton & Sagayama, 1992b).

The contents of this report is clustered in order to present the material loosely in the same order as the items appear in the proposed new training scheme as illustrated in figure 7.4 on page 69.

The first task is the selection of the model type and chapter 2 give basic definitions of the most common HMMs divided according to the characteristics of the observation density model used in the states of the HMMs.

The following two chapters (3 and 4) are devoted to the declaration of the two different vector quantization methods and their application to the density initialization of the CDHMMs and SCHMMs.

The chapter 5 presents several methods that have been used in the training of HMM parameters and have some contribution to the development of the LVQ based tuning method described detailed in chapter 6.

The results of the experiments as well as the various setups and proposed HMM configurations are presented in chapter 7 followed by the conclusions in chapter 8 and the list of references.

As this report is intended to be a description of the methods necessary to compose a training scheme for CDHMMs and SCHMMs it is perhaps necessary to somehow indicate here which ideas originate from the author and which come from other sources listed at the end of this report. The group of these original ideas consists of the use of SOM and LVQ in the initialization, the special version of LVQ based corrective tuning method characterized in chapter 6, the adjoint version of large number of Gaussian mixtures and tied variances for CDHMMs and the proposal for HMM training scheme specified in chapter 7.

Chapter 2

HIDDEN MARKOV MODELS

2.1 General

The strengths of the hidden Markov models are based on its mathematical framework and its implementational structure. Being built on the consistent statistical methodology and probabilistic theory the mathematical framework provides tractable tools for examination and analyzation of the models.

The HMMs are applied in this work to model sequential observations, where the sources of successive short-time features are in some way interrelated. The strength of HMM is that it offers a simple way to combine the modeling of stationary stochastic processes producing the observable short-time features and the temporal relationship among the processes together in a well-defined probability space. The short-time processes and the long-time characteristics can conveniently be separated for independent analysis because this probability measure can be easily decomposed into corresponding parts.

The temporal model in HMMs relies on a relatively simple structure of states and a probabilistic model of their mutual transitions, but it, anyhow, offers considerable flexibility to configure the models for the associated signal sources. To achieve an optimally efficient configuration without any a priori knowledge of the signal source requires, however, an exhaustive series of experiments and significant computational efforts. Fortunately, the HMMs are quite robust in that sense assuming that the number of states is within reasonable limits.

The modeling of the stationary observation processes associated with the states of HMMs involves also considerable flexibility since the range of observation distributions that can be accommodated by well-developed training algorithms is rather large. However, as observed in speech recognition applications the choice of the state output distribution affects strongly to the model performance, both with respect to the recognition accuracy and the computation time.

2.2 HMM formulation

2.2.1 Markov chains

A system of multiple states is a Markov chain, if for any discrete time instant t the state of the system can be described as being in one of the N distinct states expressed by the state variable q_t.[1] An important property of Markov chains is also that the state of the system at time t depends only on the q_{t-1} regardless of the previous state transition history and the value of t. A Markov chain can be described by a collection of states $\{q_i\}_{i=1}^{N}$ and a state transition probability matrix $A = [a_{ij}]$, where

$$a_{ij} = \Pr(q_t = j | q_{t-1} = i) \ , \ i, j = 1, \cdots, N \ . \tag{2.1}$$

The following axiomatic constraints for a_{ij} hold:

$$a_{ij} \geq 0 \tag{2.2}$$

and

$$\sum_{j=1}^{N} a_{ij} = 1 \ \ i = 1, \cdots, N \ . \tag{2.3}$$

The probability of any state sequence $\boldsymbol{q} = (q_0, q_1, \cdots, q_T)$ being generated by the Markov chain is

$$\Pr(\boldsymbol{q}|A, \pi) = \pi_{q_0} a_{q_0 q_1} \cdots a_{q_{T-1} q_T} \ , \tag{2.4}$$

where $\pi_i = \Pr(q_0 = i)$ is the probability of the initial state.

2.2.2 Hidden Markov model

A Markov model is called “hidden”, if the state sequence $\boldsymbol{q}$ cannot be directly observed. The observation at time t called $\boldsymbol{O}_t$ provides instead only some evidence about the state of the system. That evidence can be exploited indirectly to compute the probabilities for each state that they could have generated $\boldsymbol{O}_t$.

The production of observations is assumed to be stochastic and it is characterized by a set of observation probability measures $B = \{b_i(\boldsymbol{O}_t)\}_{i=1}^{N}$, where

$$b_i(\boldsymbol{O}_t) = \Pr(\boldsymbol{O}_t | q_t = i) \ . \tag{2.5}$$

Because the observations produced by states are assumed to be independent of each other and the time t, the probability of observation sequence $\boldsymbol{O} = (\boldsymbol{O}_1, \boldsymbol{O}_2, \cdots, \boldsymbol{O}_T)$ being generated by a certain state sequence $\boldsymbol{q}$ can be calculated by a product

$$\Pr(\boldsymbol{O}|\boldsymbol{q}, B) = b_{q_1}(\boldsymbol{O}_1) b_{q_2}(\boldsymbol{O}_2) \cdots b_{q_T}(\boldsymbol{O}_T) \tag{2.6}$$

[1]The notation used here tries to follow that of (Juang & Rabiner, 1991) whenever possible.

and the joint probability of $\boldsymbol{O}$ and $\boldsymbol{q}$ by a product of (2.4) and (2.6)

$$\Pr(\boldsymbol{O}, \boldsymbol{q}|\pi, A, B) = \pi_{q_0} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(\boldsymbol{O}_t) \quad . \tag{2.7}$$

The triple

$$\lambda = (\pi, A, B) \tag{2.8}$$

defines completely a HMM and the probability of observation sequence $\boldsymbol{O}$ being produced by that HMM without an explicit knowledge of the generating state sequence is characterized by the sum over all state sequences

$$\Pr(\boldsymbol{O}|\pi, A, B) = \sum_{\boldsymbol{q}} \pi_{q_0} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(\boldsymbol{O}_t) \quad . \tag{2.9}$$

Since the direct evaluation of (2.9) is infeasible due to the exponential growth of computations as a function of sequence length T it is necessary to consider other methods to calculate $\Pr(\boldsymbol{O}|\lambda)$. An excellent tool which cuts the computational requirements to linear relative to T is the well-known forward-backward algorithm (Baum, 1972).

The evaluation of the probability $\Pr(\boldsymbol{O}|\lambda)$ is carried out by a forward induction procedure by defining a forward variable $\alpha_t(j) = \Pr(\boldsymbol{O}_1, \boldsymbol{O}_2, \cdots, \boldsymbol{O}_t, q_t = j|\lambda)$ recursively by

$$\alpha_t(j) = b_j(\boldsymbol{O}_t) \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \ , \quad t = 1, \cdots, T \quad j = 1, \cdots, N \tag{2.10}$$

and $\alpha_0(j) = \pi_j$. So $\alpha_t(j)$ is the probability of being in state j at time t after observing sequence $\boldsymbol{O}_1, \boldsymbol{O}_2, \cdots, \boldsymbol{O}_t$. The probability of the whole sequence is then simply the sum

$$\Pr(\boldsymbol{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \quad . \tag{2.11}$$

2.2.3 The decoding of the state sequence

The usual application of HMMs is to deduce the most likely state sequence $\boldsymbol{q}^*$ from the given sequence of observations $\boldsymbol{O}$. Instead of maximizing $\Pr(\boldsymbol{q}|\boldsymbol{O}, \lambda)$ the problem can be stated equivalently as the maximization of (2.7) because $\Pr(\boldsymbol{q}, \boldsymbol{O}|\lambda) = \Pr(\boldsymbol{q}|\boldsymbol{O}, \lambda) \Pr(\boldsymbol{O}|\lambda)$ and $\Pr(\boldsymbol{O}|\lambda)$ is not involved in the optimization process. The maximization can be computed sequentially using dynamic programming by the Viterbi algorithm (Forney, 1973).

The Viterbi algorithm is closely related to the forward-backward algorithm (Baum, 1972) but uses maximization instead of summation. The forward variable here is $\delta_t(j) = \Pr(\boldsymbol{O}_1, \boldsymbol{O}_2, \cdots, \boldsymbol{O}_t, q_t = j|\lambda)$ recursively defined by

$$\delta_t(j) = b_j(\boldsymbol{O}_t) \max_i \delta_{t-1}(i) a_{ij} \ , \quad t = 1, \cdots, T \quad i, j = 1, \cdots, N \tag{2.12}$$

and $\delta_0(j) = \pi_j$. So $\delta_t(j)$ is the probability using the most likely state sequence to be in state j at time t after observing sequence $\boldsymbol{O}_1, \boldsymbol{O}_2, \cdots, \boldsymbol{O}_t$. The optimal state sequence $\boldsymbol{q}^*$

can be obtained backwards following the state sequence that leads to $q_T^* = \arg\max_i \delta_T(i)$ by, e. g. constructing the table

$$\psi_t(j) = \arg\max_i \delta_{t-1}(i) a_{ij} \ , \quad t = 1, \cdots, T \quad i,j = 1, \cdots, N \tag{2.13}$$

and collecting the states backwards

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \ , \quad t = T-1, T-2, \cdots, 0 \ . \tag{2.14}$$

2.3 Continuous density HMMs

In the HMM defined by $\lambda = (\pi, A, B)$ in the previous section the observation probability measures $B = \{b_i(\boldsymbol{O}_t)\}_{i=1}^N$, where $b_i(\boldsymbol{O}_t) = \Pr(\boldsymbol{O}_t | q_t = i)$ were left without any detailed definition. The set of HMMs that is usually called continuous (observation) density HMMs (CDHMMs) includes models in which the $b_i(\boldsymbol{O}_t)$:s are some parametric probability density functions or mixtures of them.

The most common parametric pdf used in CDHMMs is the mixture Gaussian density

$$b_i(\boldsymbol{O}_t) = \sum_{m=1}^{M} c_{im} b_{im}(\boldsymbol{O}_t) \ \ , i = 1, \cdots, N \ \ , \tag{2.15}$$

where the following constraints for c_{im} hold: $c_{im} \geq 0$ and $\sum_{m=1}^M c_{im} = 1 \ \ i = 1, \cdots, N$. $b_{im}(\boldsymbol{O}_t)$ is here a K-dimensional multivariate Gaussian density with covariance matrix $\boldsymbol{\Sigma}_{im}$ and mean vector $\boldsymbol{\mu}_{im}$

$$b_{im}(\boldsymbol{O}_t) = \frac{1}{(2\pi)^{K/2} |\boldsymbol{\Sigma}_{im}|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{O}_t - \boldsymbol{\mu}_{im})^T \boldsymbol{\Sigma}_{im}^{-1} (\boldsymbol{O}_t - \boldsymbol{\mu}_{im})} \tag{2.16}$$

The common Baum-Welch reestimation algorithm described in section 5.1.1 requires only $b_{im}(\boldsymbol{O}_t)$ to be a strictly log-concave or elliptically symmetric (Juang, 1985) so there are many other applicable parametric pdfs in addition to the multivariate Gaussians considered in this work. However, the use of mixture Gaussians is clearly the dominant trend in this research field.

To approximate simple observation sources the mixture Gaussians provide an easy way to gain a considerable accuracy due to the flexibility and convenient estimation of the pdfs. If the observation source generates a complicated high-dimensional pdf, the mixture Gaussians become computationally difficult to treat due to the excessive number of parameters and large covariance matrices.

When there are huge number of parameters to be estimated, the practical problem is the availability of the necessary amount of well-representative training data. With insufficient training data some parameters will get more or less arbitrary values and, for especially the covariance matrices, this may have drastical consequences.

To get more distinctive and representative models and in the same time to avoid excessive parameters is one of the most important trade-offs in the configuration of CDHMMs. There are plenty of different approximations which aim to concentrate to the essential parameters and omit the unimportant ones. Common strategy is to tie some similar parameters in order to get more reliable estimates, e. g. (Young, 1992; Huang & Jack, 1989a). The concept of tying and its consequences will be dealt in the section 2.5.

As the length of the feature vectors is increased the size of the covariance matrices increases in square proportional to the vector dimension. If feature vectors are designed to avoid redundant components the off-diagonal elements of the covariance matrices are usually small. This suggests to the covariance approximation by diagonal matrices. The diagonality also provides a simpler and faster implementation for the probability computation since the equation 2.5 reduces to

$$\begin{aligned} b_{im}(\boldsymbol{O}_t) &= \frac{1}{(2\pi)^{K/2}|\boldsymbol{\Sigma}_{im}|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{O}_t-\boldsymbol{\mu}_{im})^T \boldsymbol{\Sigma}_{im}^{-1}(\boldsymbol{O}_t-\boldsymbol{\mu}_{im})} && (2.17) \\ &= \frac{1}{(2\pi)^{K/2}(\prod_{k=1}^{K}\sigma_{imk})^{1/2}} e^{-\sum_{k=1}^{K}\frac{(o_{tk}-\mu_{imk})^2}{2\sigma_{imk}^2}} \quad , && (2.18) \end{aligned}$$

where the variance terms $\sigma_{im1}, \sigma_{im2}, \cdots, \sigma_{imK}$ are the diagonal elements of the covariance matrix $\boldsymbol{\Sigma}_{im}$.

2.4 Discrete HMMs

The discrete observation density HMMs are based on partitioning of the pdf of observations into a discrete set of small cells and symbols $v_1, v_2, \cdots, v_M$ one symbol representing each cell. This partitioning is a well studied subject called **vector quantization** and there exists several analysis (e. g. (Makhoul et al., 1985)) for the design of codebook, i. e. the set of vectors $\{m_k\}_{k=1}^{M}$ determining the cells.

The corresponding symbol for the observation $\boldsymbol{O}_t$ is determined by the Nearest Neighbor rule, i. e. select the symbol of the cell with the nearest codebook vector. In definition of the HMM $\lambda = (\pi, A, B)$ of the section 2.2 the observation probability measures $B = \{b_i(\boldsymbol{O}_t)\}_{i=1}^{N}$ are now defined simply as sets of fixed factors for each class i so that the probability of observing $\boldsymbol{O}_t$ in state i is

$$b_i(\boldsymbol{O}_t) = b_i(k) = \Pr(v_k|q_t = i) \quad , \tag{2.19}$$

where v_k is the symbol of the nearest codebook vector $\boldsymbol{m}_k$ for $\boldsymbol{O}_t$. The following constraints for $b_i(k)$ naturally hold: $b_i(k) \geq 0$ and $\sum_{k=1}^{M} b_i(k) = 1 \;\; i = 1, \cdots, N$.

The estimation of the probabilities $b_i(k)$ is normally accomplished in two phases: first the determination of the codebook and then the estimation of the sets of observation probabilities for each codebook vector in each state. The traditional approach (Rabiner et al., 1983) is to minimize quantization error in the first phase and maximize observation likelihood in the second. The corresponding convenient and widely applied algorithms are the K-means clustering (MacQueen, 1967) and the Baum-Welch reestimation (Baum et al., 1970).

The minimization of quantization errors is, however, not necessarily the best objective in the codebook optimization, because the ultimate goal here is not to represent the observation density as closely as possible but to create HMMs which produce as few recognition errors as possible. As observed in e. g. (Iwamida et al., 1990a) the codebook is not a good index of HMM classification performance. An advanced scheme to the codebook design is to minimize the classification errors by LVQ (Kohonen, 1986b) (discussed here in chapter 4). By using a LVQ-HMM hybrid system the high discriminant power of LVQ and the temporal modeling capabilities of HMM can be successfully integrated (Iwamida et al., 1990a; Torkkola et al., 1991a)

The use of vector quantized HMMs for phoneme modeling is well justified in the sense of the complexity and overlapping associated with the output pdfs. Since the output densities of the states of HMMs do not resemble any parametric pdfs the loss of accuracy obtained by VQ compared to continuous densities is neglible. Another convenient fact is that a powerful discriminative training algorithm (LVQ) is readily available for the codebook training. It is noteworthy that as the computation of the discrete output probability is actually a table-lookup operation a considerable recognition speedup can be gained compared to CDHMMs.

The major problem with discrete output probability is that the VQ operation partitions the continuous acoustic space into separate regions destroying the original signal structure. As the output distributions of different states are typically highly overlapping the partitioning introduces small errors. The unreliability of parameter estimates due to the finiteness of the training data prevents the presentation of very large codebooks. This introduces quantization errors which may cause degradation in the DHMM performance when the observed feature vectors are intermediate between two VQ symbols.

2.5 Semi-continuous HMMs

The SCHMM (Huang & Jack, 1989b) is actually a special form of the continuous density hidden Markov model where the set of Gaussian density functions used in the output mixture densities is shared between all the states in all Markov models. The SCHMM can be also regarded as a kind of intermediate between the discrete vector quantization based methods and the normal CDHMM. The concept of semi-continuous density functions is perhaps best understood by observing the models separately from both DHMM and CDHMM point of view.

2.5.1 The tying of mixture densities

As already pointed out in section 2.3 the tying of parameters can be successfully used to reduce redundant information from the HMMs. In a SCHMM the number of mixture Gaussian pdfs used to model the output density of each state can be radically increased while maintaining the same level of computational complexity and same requirements for training data as the corresponding CDHMM.

Because the continuity of the acoustic features and the overlapping of the output densities of different states, it seems reasonable not to estimate a totally different set of mixture Gaussians for each state, but to use the same large Gaussian codebook for all states and only estimate different mixture weights for each state.

With tied mixture densities the reestimation formulas for CDHMMs represented in chapter 5 remain essentially the same. In the case of ML estimation (section 5.1) the Gaussian codebook can be estimated together with the other HMM parameters like the state-dependent Gaussians in CDHMMs. It is also possible to divide the parameter estimation into two distinct phases like in DHMMs, i. e. first estimate the codebook and then use that fixed codebook to estimate the other parameters. The SCHMM parameter estimation scheme proposed in (Kurimo, 1993) combines both these methods in order to first separate the estimation problem to be able to efficiently estimate the Gaussian codebook and then use the unified approach to achieve an optimal combination of the HMM and the VQ parameters. By the terminology used in this work the first codebook estimation phase is called the density initialization (dealt in chapters 3 and 4) and the second reestimation phase is actually called parameter estimation (chapter 5).

2.5.2 Extension of the DHMMs

The SCHMM as proposed in (Huang & Jack, 1989b) is an extension of the DHMM where the discrete output probabilities are combined with continuous pdfs of mixture Gaussian codebook. By this combination the errors involved in VQ (discussed in section 2.4) can be effectively eliminated. What is obtained by SCHMM is actually a codebook of Gaussian density functions, where the densities are smoothly overlapped rather than partitioned as in the VQ. So an acoustic event is not described by a single discrete probability but a continuously weighted combination of several probabilities.

The probability of observing $\boldsymbol{O}_t$ is the equation 2.19 extended by a summation over all M codebook symbols weighted by conditional Gaussian pdfs

$$b_i(\boldsymbol{O}_t) = \sum_{k=1}^{M} b_i(k) f_k(\boldsymbol{O}_t) \quad , i = 1, \cdots, N \quad , \tag{2.20}$$

where $f_k(\boldsymbol{O}_t)$ is a multivariate Gaussian pdf with covariance matrix $\boldsymbol{\Sigma}_k$ and mean vector $\boldsymbol{\mu}_k$.

Another interpretation of equation 2.20 is that the discrete probabilities $b_i(k)$ are state-dependent weights for the continuous probabilities of the mixture Gaussian codebook. From that CDHMM point of view this actually is a M-mixture CDHMM, where all the mixture Gaussian shared by all states of all HMMs. Thus by substituting the notation of mixture weights c_{im} by $b_i(k)$, where the indexes m and k are equivalent, and the notation of Gaussian pdfs $b_{im}(\boldsymbol{O}_t)$ by $f_k(\boldsymbol{O}_t)$ using the tying over all states $i = 1, \cdots, N$, the equation 2.20 follows directly from equation 2.16.

2.5.3 The general use of tying

The concept of tying can be accustomed more generally than only for Gaussian mixtures as explained in (Young, 1992). In phonemes similar to each other it may occur that some optional or obligatory states depending on the model configuration are so similar that it is advantageous to consider these states as only one state. Another possibility is to tie only the output densities if the source of the observations seem to be the same for some states.

By tying the parameters where the loss of the modeling information is small the robustness of the model can be increased and model size reduced. Perhaps the most suitable sets of parameters in which the tying might be accomplished are some similar states (as mentioned above), the Gaussians used in mixture densities (SCHMMs) and the covariance matrices.

In (Paul, 1989) it is reported that the tying of variances does not degrade performance which was also observed in (Kurimo, 1992). The tying of covariance matrices reduces significantly the number of parameters and probability computation time for the diagonal covariance matrices, because the variances of the vector components can be joined to the feature computation since the following holds if $\sigma_{imk} = \sigma_k \;\; \forall i, m$

$$\begin{aligned} b_{im}(\boldsymbol{O}_t) &= \frac{1}{(2\pi)^{K/2}(\prod_{k=1}^{K}\sigma_k)^{1/2}} e^{-\sum_{k=1}^{K}\frac{(o_{tk}-\mu_{imk})^2}{2\sigma_k^2}} && (2.21) \\ &= \frac{1}{\beta} e^{-\frac{1}{2}\sum_{k=1}^{K}(\frac{o_{tk}}{\sigma_k}-\frac{\mu_{imk}}{\sigma_k})^2} && (2.22) \\ &= \frac{1}{\beta} e^{-\frac{1}{2}\sum_{k=1}^{K}(\tilde{o}_{tk}-\tilde{\mu}_{imk})^2} \quad , && (2.23) \end{aligned}$$

where $\beta = (2\pi)^{K/2}(\prod_{k=1}^{K}\sigma_k)^{1/2}$ is a constant and the feature vectors are already scaled so that $\tilde{\boldsymbol{O}}_t = [o_{t1}/\sigma_1, \cdots, o_{tK}/\sigma_K]$. So instead of $\boldsymbol{\mu}_{im}$ the pre-scaled mean vector $\tilde{\boldsymbol{\mu}}_{im} = [\mu_{im1}/\sigma_1, \cdots, \mu_{imK}/\sigma_K$ can be estimated directly and division operations are reduced from equation 2.23.

2.6 Parameter estimation

2.6.1 Iterative methods

An important reason why the HMMs have become so popular in pattern recognition is the presence of effective algorithms for the estimation of the model parameters. Due to the character of the estimation problem no closed form solution can be found, and, in fact, for a given set of parameters it is impossible to prove explicitly that the global optimum is reached. Since the objective of the estimation is to have a model with optimal recognition performance it is also impossible to define unique gradients for the parameters to implement an accurate gradient search algorithm. However, in practical problems like pattern recognition it is not necessary to reach the global optimum but to get a model close to the optimum and giving an enough good performance. Therefore the solutions presented here and in the literature are typically iterative procedures, which enhance the existing parameter values towards some better values in order to maximize some objectives.

2.6.2 Selection of the objective

The traditional approach is to formulate the inverse estimation problem, i. e. the selection of the model parameters so that the likelihood of the training data being produced by the model is maximized. This method is called the Maximum likelihood estimation (MLE), which, in the case of HMMs, can be conveniently accomplished by an iterative two-step procedure called Baum-Welch reestimation (Baum & Petrie, 1966).

The MLE, however, need not be the optimal method in terms of minimizing the classification error rate in recognition tasks. The MLE also supposes that the optimal model can be represented as a suitable parameter combination belonging to the a priori known model family. If the model family of the true model is unknown and incorrectly approximated, the MLE may operate very ineffectively. If the training data is completely labeled, i. e. the association between the data and the classes is precisely known a priori during the training as in the supervised learning, the conditional maximum likelihood estimator (Nadas et al., 1988) can be used to gain robustness of the a priori model. This estimator can be shown to lead to the maximization of the mutual information (MMI) between the observations and class labels (Bahl et al., 1986). However, the MMI suffers from the lack of convenient and robust training algorithms.

A completely different approach to the parameter estimation is to make an attempt to modify the parameters to directly minimize the original training objective, i. e. the misclassification rate. There are several different strategies to accomplish the modifications, for example, the error corrective training proposed by Bahl et al. (Bahl et al., 1988), which tunes the discrete observation probabilities for every training sample for which the likelihood of the observations using the correct model is not better than that with each incorrect model by a certain threshold value.

Chapter 3

DENSITY INITIALIZATION BY SOM

3.1 Initialization of HMMs

3.1.1 Importance

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

The importance of initialization for complicated optimization surfaces can be imagined like a huge topologically uneven plain with extremely many locally optimal points like little hills or wells. The explorer having a limited number of time or strength must climb every hill on his way to find out the highest one. Thus it is easy to realize how much a carelessly chosen initial point can affect to the efficiency of the iteration although the method could always point the locally correct direction.

When striving for better models by increasing modeling complexity (number of HMMs, states, components in output distribution and in feature vectors) also the roughness of the objective surface and the number of local optimums usually increases as the size of the training database is increased in order to reliably estimate the model parameters. So the careful initialization not only saves time but also leads to better models because, in practice, the search must be finished long before the global optimum is reached.

3.1.2 Objective

As stated in the previous section the objective of the initialization is to select suitable parameter values for the iterative training algorithm. It is, however, not so clear that the most suitable initial parameters are those who immediately give the best performance for the recognition objective. Sometimes it may happen that the best performing initial values do not give as good final model as another initialization with other qualifications would give. Those qualifications which determine the suitability of the initialization depend naturally on the actual estimation algorithm. For example, in this work it is shown experimentally that for the MLE of CDHMMs the best initialization for the output distributions is to select them so that the states are well discriminated.

3.2 SOM algorithm

3.2.1 VQ

The motivation for vector quantization (VQ) is to present a complicated continuous multivariate probability function $p(\boldsymbol{x})$ by a finite set of vectors. This constitutes a mapping in which each individual vector $\boldsymbol{x}$ chosen according to $p(\boldsymbol{x})$ will be mapped to the nearest vector of the finite set. The vector set $\{\boldsymbol{m}_i\}_{i=1}^{N}$ [1] is called a **codebook** and its vectors m_i are **codebook vectors**. The difference of a sample vector $\boldsymbol{x}$ and the nearest codebook vector m_c is called the **quantization error**

$$e = ||\boldsymbol{x} - \boldsymbol{m}_c|| \ , \tag{3.1}$$

where m_c is determined by

$$c = \arg\min_i \{||\boldsymbol{x} - \boldsymbol{m}_i||\}. \tag{3.2}$$

3.2.2 Learning algorithm

The SOM algorithm (Kohonen, 1982) for learning the suitable values of the VQ codebook vectors can be described as follows:

Pick up samples $\boldsymbol{x}$ in a random order and update the SOM by adjusting the best-matching unit (the nearest codebook vector m_c) and its neighbors closer to $\boldsymbol{x}$.

$$\boldsymbol{m}_i(t+1) = \begin{cases} \boldsymbol{m}_i(t) + \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_i(t)] & \text{for } i \in N_c(t), \\ \boldsymbol{m}_i(t) & \text{for } i \notin N_c(t). \end{cases} \tag{3.3}$$

The neighborhood $N_c(t)$ around the best matching unit is wide in the beginning of the training and shrinks monotonically with time. The teaching gain $\alpha(t) \in (0, 1)$ is also monotonically decreased during teaching.

[1]The notation here follows mainly that of (Kohonen, 1990b).

3.2.3 Quantization error vs. Topology preserving

It is often desired that the quantization error of a VQ mapping , i. e.

$$E = \int ||\boldsymbol{x} - \boldsymbol{m}_c||^2 p(\boldsymbol{x}) d\boldsymbol{x} \ . \tag{3.4}$$

is minimized. The minimization of 3.4 is, however, difficult since the choice of index c depends on the corresponding $\boldsymbol{x}$ and all the codebook vectors $\{\boldsymbol{m}_i\}_{i=1}^N$. The minimum can be approached iteratively using the SOM with $N_c(t) = \{c\}$, which means that only the best-matching unit (bmu) for each sample $\boldsymbol{x}(t)$ participates to the learning process. The SOM, where no neighbors of the bmu are taught is actually very similar to the iterative K-means (MacQueen, 1967).

Another objective which is sometimes also desired, is to maintain the topology of the sample vector space as closely as possible. This means that similar samples will be mapped to SOM units near each other. To achieve an optimal VQ solution in respect of these two goals SOM with a suitable neighborhood definition $N_c(t)$ should be applied.

3.2.4 Using different topologies and neighborhoods

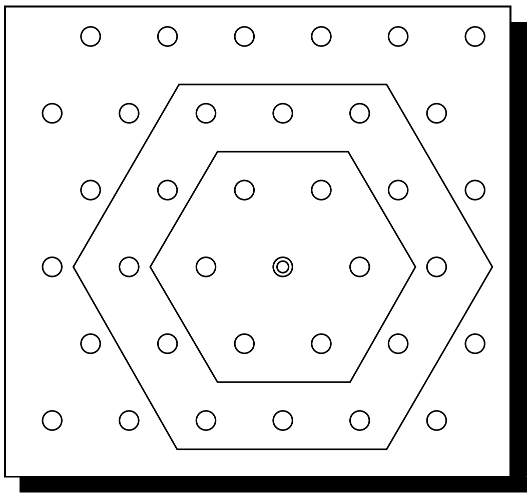


Figure 3.1: *In SOMs the reference vectors are organized to a two-dimensional array. The neighborhood can be defined using, for example, a hexagonal grid. Two neighborhoods which have a different radius are illustrated for the unit in the middle.*

The most common architecture is perhaps a two-dimensional square or hexagonal grid, where the units are organized to the nodes (fig. 3.1). The neighborhood around the best-matching unit $N_c(t)$ is normally defined using the topological distance between the map units. In general, the SOM update rule 3.3 can be expressed as

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) + \alpha(t) h_{ci}(t)[\boldsymbol{x}(t) - \boldsymbol{m}_i(t)] \ , \tag{3.5}$$

where $h_{ci}(t)$ is called the *neighborhood function.* Originally a step function is used as the neighborhood function giving $h_{ci}(t) = 1$, if the unit i belongs to the $N_c(t)$ and 0 otherwise.

It is also possible to apply a neighborhood function that decays, for example, linearly or like the Gaussian density function when the topological distance from the best-matching unit increases.

Without presenting any rigorous analysis it can be noted that when the values of $h_{ci}(t)$ are close to one for many units i around the bmu the topology is tight and the units will stay close together. However, when the learning continues with smaller values of $h_{ci}(t)$ the topology will became more flexible and units will travel further from each other causing folding of the SOM to minimize the overall quantization error 3.4.

3.2.5 Effects of the neighborhood to the density of the quantization levels

It has been shown (Ritter, 1991) that the asymptotic density of the quantization levels $Q(\boldsymbol{x})$ of a one-dimensional SOM depends on the number n of neighbors on each side of the best matching unit (the step function neighborhood is assumed) according to the law

$$Q(\boldsymbol{x}) = CP(\boldsymbol{x})^{\alpha} \ , \tag{3.6}$$

where C is some constant, $P(\boldsymbol{x})$ the input signal distribution and

$$\alpha = \frac{2}{3} - \frac{1}{3(n^2 + (n+1)^2)} \ . \tag{3.7}$$

Applying the equation 3.7 for $n = 0$, i. e. when none of the neighbors participates to the learning, produces $\alpha = 1/3$. This comprises (in one-dimensional case) the result by Zador (Zador, 1982) for the LBG algorithm that

$$\alpha = d/(d + r) \ , \tag{3.8}$$

for $r = 2$ and d being the dimensionality of the data.

For the continuum limit $n = \infty$ the equation 3.7 gives $\alpha = 2/3$. This result has been separately derived in (Ritter & Schulten, 1986).

Since α increases monotonically with n the density of the quantization levels approaches to the pdf of $\boldsymbol{x}$ as the number of neighbors increases.

With a finite number of quantization levels it is intuitively clear that as the densest parts of $p(\boldsymbol{x})$ will be represented with many quantization levels and the quantization errors will be small the immediate consequence is to have few quantization levels and high errors in the sparsest parts of $p(\boldsymbol{x})$. So the minimization objective will not be exactly like 3.4 any more, because the importance of single high errors decays in relation to density of errors.

Using the equation 3.6 with different values of α changes the minimization criteria from 3.4 to

$$E = \int ||\boldsymbol{x} - \boldsymbol{m}_c||^r p(\boldsymbol{x}) d\boldsymbol{x} \ , \tag{3.9}$$

where r is determined using the equation 3.8 by

$$r = d\frac{1-\alpha}{\alpha} \ . \tag{3.10}$$

In the one-dimensional case $d = 1$ applying the equation 3.7 produces

$$r = \frac{1}{2} + \frac{3}{2(2n+1)^2} \ . \tag{3.11}$$

In the interesting limit cases where $n = 0$ and $n = \infty$ the equation 3.11 gives $r = 2$ and $r = 1/2$, respectively. The former yields the common least-square error measure (3.4) and the latter a kind of square-root error measure in which single large errors are not so significant for the total result coinciding neatly with the intuition of unfolding map.

3.3 Some general properties of SOMs

3.3.1 Smoothing by using the neighborhoods

The similar asymptotic equations (3.7 and 3.11) as presented for the one-dimensional case have not yet been derived for $d > 1$. However, intuitively it seems obvious that larger neighborhoods n give larger values for α also when $d > 1$. This phenomenon can be exploited when the minimization criteria is not necessarily known to be like equation 3.4 but some smoothing is needed so that the quantization levels are spred more like the pdf of the data vectors. An example of the benefit of smoothing was experimentally observed when the best possible initialization method for the HMMs in this work was explored. The following simple examples try to illustrate the smoothing.

A simple experiment like that of figures 3.2 and 3.3 reveals that a large neighborhood leads to a stiff unfolding SOM (the line connecting the SOM units in numerical order). In the figure 3.2 all the SOM units lie in the centers of the two input clusters (except those which will remain between the clusters, because their neighbors lie in different clusters). The experiment was started with poor initial values (far outside of the picture). The large neighborhood was used to bring all the units quite quickly near to the middle of the most interesting area. Then the neighborhood was decreased in order to give the units more freedom to find the best places to minimize the total quantization error 3.4. Finally when trained without any neighbors the units are situated like, for example, in the figure 3.3.

A small experiment with cepstral features of spoken (finnish) phonemes (without /K/, /P/ and /T/) is presented in figure 3.4. The units of SOM are given phonetic labels according to their nearest neighbors (by euclidean distance) in the training data. This example illustrates the phenomenon that when the neighborhood is kept large, the corners of the SOM will remain less folded. Some phonemes like /S/ and /U/ which have a tendency to lie in the corner of the SOM and which do not belong to the (top three) commonest phonemes in the training material will be poorly represented and some others like /H/ and /V/ which occur more rarely but tend to lie nearer of the center of SOM will be better represented.

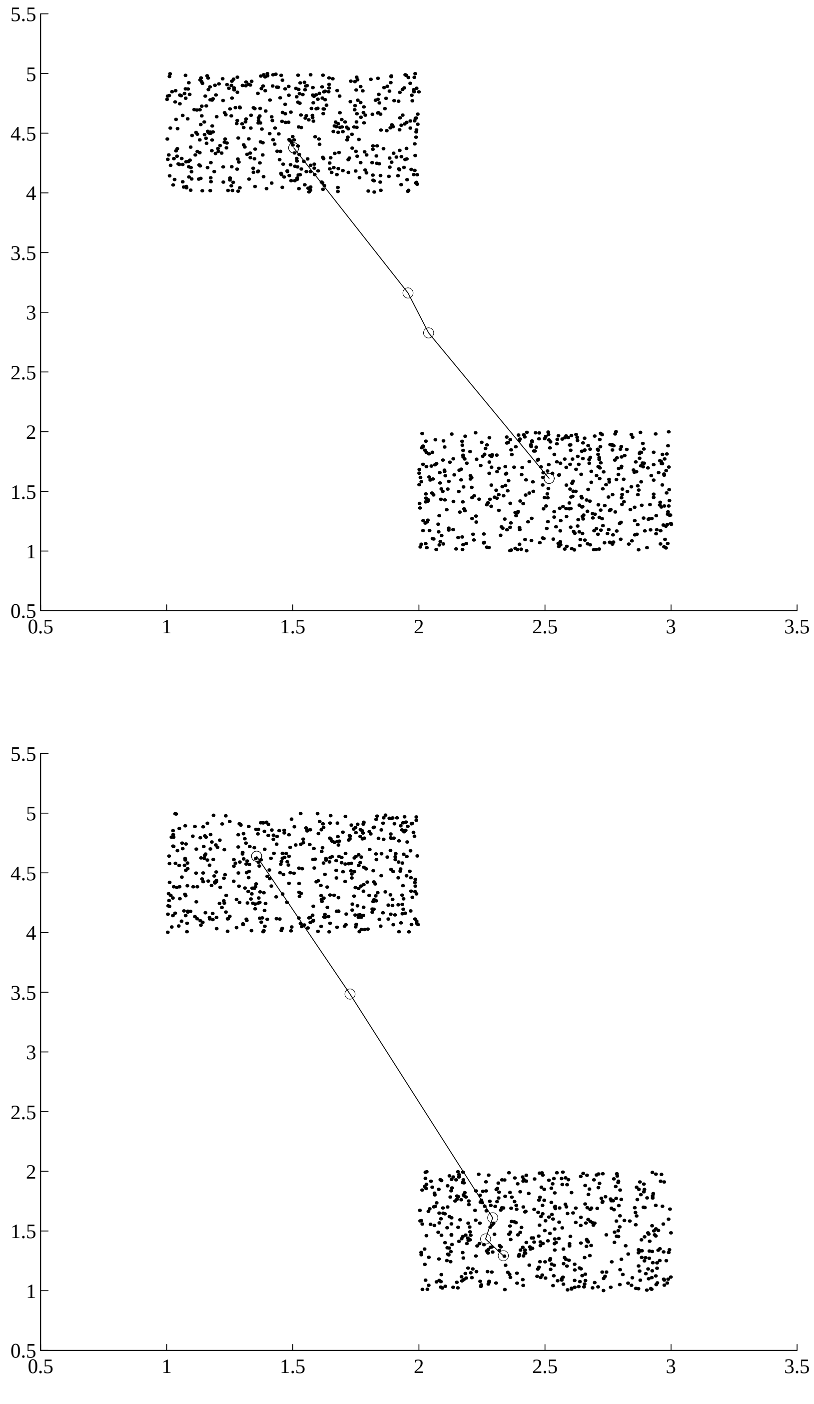


Figure 3.2: *Small one-dimensional SOMs for a special two-dimensional pdf (small dots). The SOM has six units (small circles) connected in numerical order. In the first picture 1000 iterations are performed using a learning neighborhood of the two nearest units in both directions. In the second the learning is continued with another 1000 iterations with a neighborhood of only the nearest unit in both directions. The learning rate was a constant 0.2.*

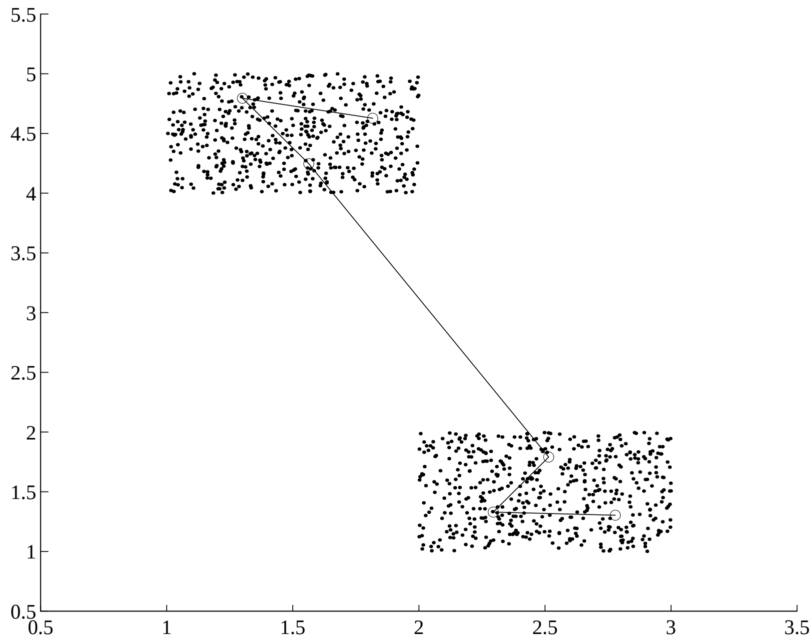


Figure 3.3: *The training of the SOM is continued with an additional 1000 iterations without training any neighbors of the bmu. The learning rate was still a constant 0.2.*

An important caution in interpreting these two experiments is that they are absolutely inadequte for drawing any conclusions with, but they merely only illustrate what the author means by smooth representation.

3.3.2 Applicability of unlabeled training data

The common situation in practical pattern recognition problems is that the support of manually recognized training data is severely limited, but plenty of raw unprocessed data is readily available. One favourable property of the SOM is that the self-organization manages well the situations with limited support of labeled training data.

The SOM clusters automatically unlabeled training vectors which lie near each other in the feature vector space by tuning one or several map units to be especially sensitive to features like those vectors maintaining a well-balanced set of weight vectors in respect to the input density function.

For visualization purposes or for further enhancements of the class discrimination ability the clusters of the SOM can be labeled by a rather small set of labeled training vectors.

Figure 3.4: *The SOM has 96 units organized into a 12 by 8 hexagonal array. The feature vectors are 20-dimensional cepstra computed from a 20 ms signal window in the center of a phoneme. The SOM in the top-left picture is taught by decreasing the neighborhood radius from 10 to 0 and learning rate from 0.02 to 0 in 10000 iterations. In the top-right, bottom-left and bottom-right pictures the teaching was similar except that the neighborhood radius was decreased from 10 to 1, 3 and 5, respectively. To best illustrate the differences of these SOMs the labels were assigned to the units according to the K-Nearest Neighbors rule using the training data.*

3.3.3 Using the topology of SOM

The preservation of the high-dimensional input space topology while mapping the feature vectors to a one or two-dimensional coordinates has proven to be a generous source of different applications. A brief summary of some of those applications provides a general idea how the topology can be exploited.

The most obvious (but not so unimportant as it may be considered at the first sight) consequence of the topology preservation is the *visualization* of the features for human observers. It is a valuable tool for process monitoring and complex fault diagnosis (see e. g. (Kasslin et al., 1992)).

In (Luttrell, 1990) it is shown that for code transmission through channels in the presence of small transmission errors the SOM provides an optimal way to organize the *coding* in order to minimize the effect of incorrectly transmitted bits. When similar features are mapped to similar codes, an incorrectly received part of a code word does not cause a severe error in decoding.

For *data compression* the situation is quite similar as in coding. To compress data of small local spatial or temporal variation a SOM in which certain units are especially sensitive to some common input area, the preserved quality and obtained compression rate can be efficiently optimized for the given input density.

When sequential information of the inputs are essential, the topologically correct mapping can be used to constitute *trajectories* where the best-matching units of the SOM are recorded during the process. These trajectories and their visualization are exploited in e. g. the analysis of voice disorders (Leinonen et al., 1991).

In certain *combinatorial optimization* problems, e. g. the travelling salesman and VLSI design, the topology of the SOM is applied to keep neighboring objects close to each other while searching for the optimal combinations (Fort, 1988; Zhang & Mlynski, 1990). This prevents some unnecessary exploration of obviously unworthy solutions.

The clustering of the input density produces sometimes unwanted results if the training data set is too small. Instead of giving a well-balanced presentation of the continuous input density some unimportant discrete points appearing in the training set might gain too much importance in the clustering. This phenomenon is conveniently avoided by using SOM and starting the learning with a large radius covering the all units. Organizing such a smooth mapping intends to prevent the appearance of unwanted outliers in the clustering due to limited training data.

3.3.4 Probability approximation by SOM

Reminding the basic property of SOM that the density of the reference vectors approaches the pdf of the input source it sounds reasonable to use SOMs somehow to directly estimate the pdf values.

It is intuitively plausible to suggest that when the minimization criteria has been $\int ||\boldsymbol{x} - \boldsymbol{m}_c||^2 p(\boldsymbol{x}) d\boldsymbol{x}$ as given in section 3.2.3, the probability of observation $\boldsymbol{x}$ would be inversely propotional to the square of the distance to the best-matching unit $||\boldsymbol{x} - \boldsymbol{m}_c||^2$.

Using a standard nonparametric density estimator called Parzen estimators the SOM units and their Voronoi cells can be considered as the Parzen kernels. The Parzen density estimate is formulated as

$$\hat{p}(\boldsymbol{x}) = \frac{1}{N} \sum_{j=1}^{N} \gamma(||\boldsymbol{x}_j - \boldsymbol{m}||) \quad , \tag{3.12}$$

where γ is the Parzen kernel function and m its center point. The inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N$ are independent samples of the random variable $\boldsymbol{x}$. The most common kernel types are rectangular, triangular, exponential and Gaussian. To obtain valid pdf estimates the kernel must be positive and normalized so that it integrates to one.

It is often desired that the density approximation should be continuous. With multiple Parzen kernels this follows when the kernels are chosen to be continuous (e. g. Gaussians) and all probability estimates given by different kernels are summed for the final estimate. However, in practice it is not necessary to compute all the kernels since a couple of nearest ones will give the desired accuracy.

By giving different weights to the Gaussian kernels according to the number of training samples belonging to the Voronoi cells of the corresponding SOM units, it is possible to obtain better accuracy with fewer kernels. This is evident since one kernel with a large weight approximates the effect of many smaller kernels situated very near to each other as often happens in some part of the SOM. If the number of SOM units were very large and if the reference vectos were perfectly distributed according to the real pdf $p(\boldsymbol{x})$, the unit weights would be almost equal since the probability mass inside each Voronoi cell would be the same.

The density estimate by weighted Gaussian kernel estimators based on SOM can be expressed as

$$\hat{p}(\boldsymbol{x}) = \sum_{i=1}^{M} c_i p_i(\boldsymbol{x}) \quad , \tag{3.13}$$

where c_i is the weight factor for i'th kernel and

$$\hat{p}_i(\boldsymbol{x}) = \frac{1}{(2\pi)^{K/2} \sigma_i^K} e^{-\frac{||\boldsymbol{x} - \boldsymbol{\mu}_i||^2}{2\sigma_i^{2K}}} \quad . \tag{3.14}$$

In Gaussian kernel the σ determines the width and $\boldsymbol{\mu}$ the center of the kernel. The obtained estimator is quite near to the observation density function of continuous density HMM (equation 2.16). Actually, the observation density used in CDHMMs is just a generalization of equation 3.14 so that kernel width is not the same for different directions.

The so called radial basis functions (RBF) network (Broomhead & Lowe, 1988) is another version of weighted Gaussian kernel estimators in which the Gaussian means, variances and weight factors are iteratively learned by a neural network. The training of RBFs rely, however, on a different learning philosophy than SOMs or MLE and is not discussed in this work.

In (Hämäläinen, 1992) the weighted Gaussian kernel estimator is further transformed to adjust the slight difference in the asymptotic quantization level distribution of the SOM and the input distribution described by equation 3.8. However, this correction can be neglected when the dimensions are high and zero-neighborhood is not excessively involved.

3.4 Using SOM for observation density initialization of HMMs

3.4.1 Motivation

In this work SOMs are applied to produce adaptively a representative clustering of the observed feature vectors for a state of HMMs.

When the HMMs with discrete observation symbol densities are used, a vector quantizer is applied to map the continuous observation vectors into a discrete codebook symbols. The mapping is done by choosing the symbol of the nearest codebook vector for each feature vector. As explained in section 2.4 each state in each Markov model assigns an observation probability value for each codebook symbol. Before any probabilities can be estimated an appropriate quantization codebook must be estimated.

The codebook design for DHHMs should minimize the average distortion and also keep the number of codebook vectors small enough to maintain the ability to estimate reliably all the observation probabilities within the training data available. The finiteness of the training data causes also that some kind of smoothing for the HMMs is advantageous since the database does not give a complete representation of the signal space that has to be recognized by the system. The SOM can be successively applied for the codebook design because it provides smoothing by using the neighborhood of similar vectors in learning while also minimizing the average distortion (Monte, 1992; Zhao & Rowden, 1991; Zhao, 1992).

When using the CDHMMs with mixture density functions that have many components their initialization becomes especially important. The suitable initial placement of the mean vectors is essential because the estimation of the mixture weights and the covariance matrices is based on them. The initialization of the mixture weights and the covariance matrices is more straightforward because adequate estimates can be obtained by analyzing separately the clustered observations of each state when the mean vectors of mixture components are first determined. The quality of the initial estimates depends naturally on the quality of the achieved clustering. Compared to the more traditional K-means clustering the SOM seems to produce a more proper set of vectors for further training due to the use of neighborhoods in training which organizes the similar vectors close to each other and provides a kind of smoothing for the Gaussians (Kurimo & Torkkola, 1992b; Kurimo & Torkkola, 1992a).

3.4.2 Implementation issues

CDHMMs

To use the SOMs for the initialization of the Gaussian mean vectors of the CDHMMs separate small SOMs are trained for each state of each HMM. The set of weight vectors of such a SOM then representates the observation density of the corresponding state. After training the weight vectors are directly transformed to the Gaussian mean vectors to serve as an initialialization for the observation densities.

To perform the training the data needs to be initially somehow divided into phoneme samples and further between different states. This division can be computed by some initial HMM model designed for another speaker or a group of speakers or by some more straightforward detection of certain difference characteristics. The feature vectors initially associated with certain state are then selected in random order to train the SOMs. The initialization of the SOMs does not have much effect here and even the division of the training data is not required to be accurate. In fact, adequate results are obtained by dividing the phoneme samples for sequences of equal lengths and giving random initial reference vectors for SOMs.

It could also be possible to implement the SOMs for CDHMMs so that the small SOMs are at first treated as a single large SOM, which could be trained unsupervised like for SCHMMs and later labeled using a smaller set of presegmented training samples. The division of SOM units based on the majority voting should, however, take care that each state in each phoneme would have an equal, or at least an almost equal, number of units.

SCHMMs

With SCHMMs, where the Gaussians used for the mixture density functions are tied between all the states in all the HMMs, the neighboring property is more clearly visible than in CDHMMs. As in figure 3.5, the mean vectors of Gaussians that strongly response to the same phonemes will be ordered near each other. Since there are, in general, less strongly responsing Gaussians for each state than in corresponding CDHMMs (the mixtures would otherwise be oversized) the placement of the mean vectors may be even more crucial in order to get good observation probability estimates. In (Kurimo, 1993) the final recognition accuracies when using SOMs and K-means for the mean vector clustering are compared and the best results are obtained by SOMs where the final neighborhood radius is set to 1.

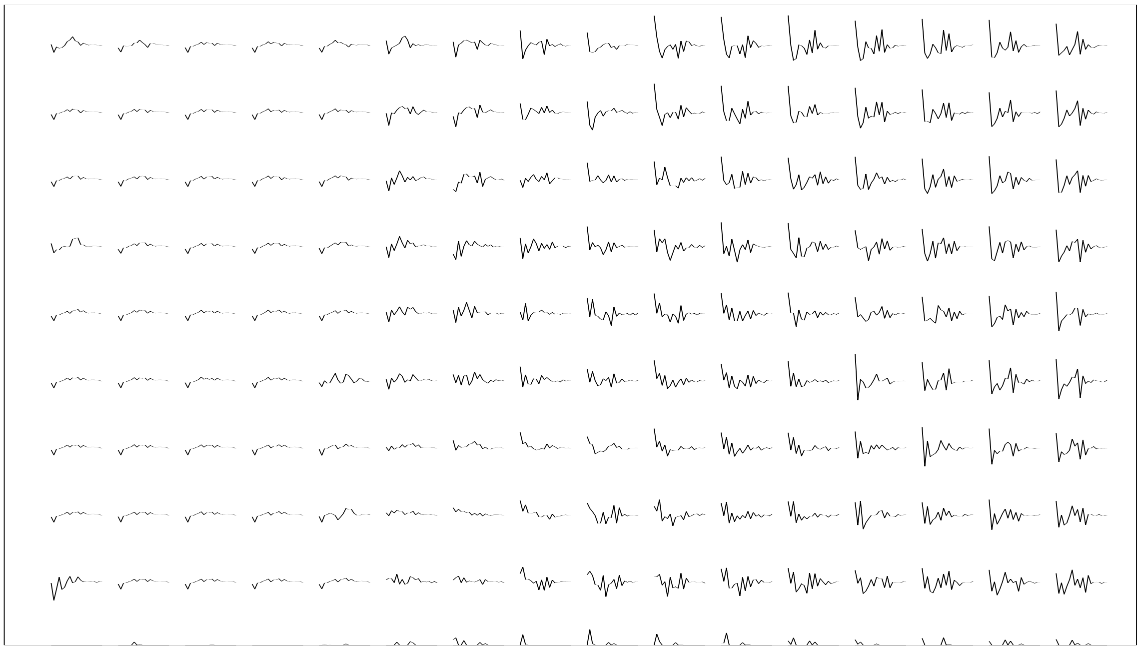
The main implementional difference between SOMs for SCHMMs and CDHMMs is that with SCHMMs the Gaussians are the same for each state and so only one SOM is required instead of, e. g. 100 (20 HMMs, 5 states per HMM). To accurately approximate the observation densities for presumably very different states the size of the SOM needs to be considerably larger, i. e. 10–50 times more units than that for CDHMMs. However, the training of the SOM is simpler, because all the training data can be circulated as itself without any segmentation or division into states.

3.4.3 Analysis of mixture weights

The mixture weights estimated for the mixture density HMMs offer an interesting way to analyze the mutual relations between the mixture densities. Since the weights are estimated using the ML principle (section 5.1) the straightforward interpretation is that a weight of a mixture density presents the conditional probability $P(\boldsymbol{x} \in N_k | \boldsymbol{x} \in C_i)$ for a feature vector to be generated by density N_k given that it belongs to state C_i.

When the mixture densities of SCHMMs are organized by a two-dimensional SOM and plotted in three dimensions as a function of mixture weights, like in figure 3.6, the large weight values for a certain state will lie nicely clustered in a certain part of the SOM. This reveals the fact that the mean vector initialization made by a SOM gives the mixture densities a well preserving structure which will not be broken by the ML estimation.

An interpretation of the figure 3.6 could be that the strict localization of large mixture weights allows that using only a small part of the set of the mixture densities gives a robust approximation of the observation densities that would be accurate enough for practical recognition purposes and spare a considerable amount of pdf computation.



<	<	<	<	<	<	<	<	>	U	U	O	O	O	A	O
<	<	<	<	<	<	<	<	K	U	U	O	O	O	A	A
<	<	<	<	<	<	H	<	V	M	U	L	L	A	A	A
<	<	<	<	<	<	<	<	V	M	L	L	L	A	A	A
<	<	<	<	<	<	<	R	M	M	N	U	A	A	A	A
<	<	<	<	<	<	<	<	V	N	N	N	R	O	A	A
<	<	<	<	<	<	<	V	>	N	N	N	<	R	A	A
<	<	<	<	<	<	V	D	L	L	N	N	R	<	Ä	Ä
K	<	<	<	<	<	<	J	L	L	Y	Y	E	E	Ä	Ä
K	K	>	>	H	<	<	J	J	I	Y	I	E	E	Ä	Ä
S	S	S	S	T	H	I	I	I	I	I	E	E	E	Ä	Ä
S	S	S	S	H	I	I	I	I	I	I	I	E	E	E	Ä

Figure 3.5: *The Gaussian mean vectors of a 192-mixture example are organized into a 16x12 table by SOM, labeled by majority voting and trained futher by LVQ. The upper picture shows the cepstra and the lower shows the corresponding phoneme labels. The < and > labels are for silence.*

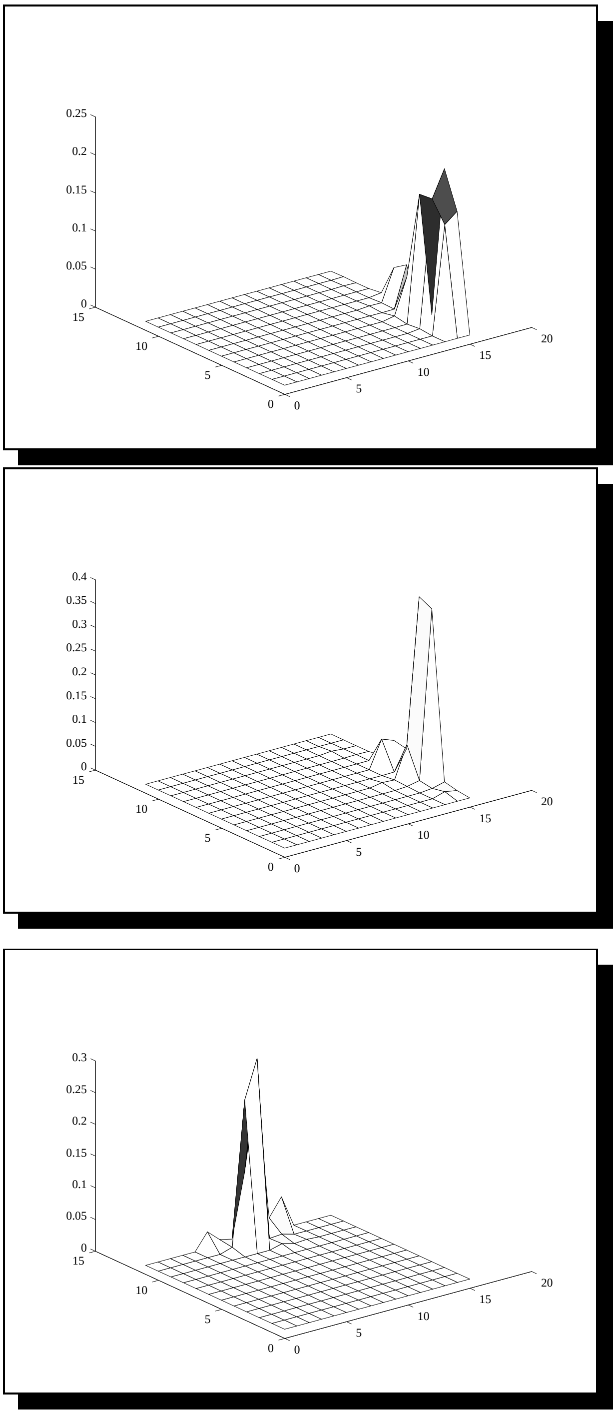


Figure 3.6: *The mixture coefficients in output density functions for states A1, A2 and I1. The 16x12 codebook of Gaussians is the same as in figure 2.*

Chapter 4

DENSITY INITIALIZATION BY LVQ

4.1 Increasing discrimination between the models

In this work the *Learning Vector Quantization* (LVQ) is applied for CDHMMs to place the mixture means in hope to enhance the discriminatory power of phoneme models. Compared to the mixture densities selected to closely approximate the conditional output pdfs of each state of the CDHMMs the mixture densities for optimal classification ability may not produce as accurate probability estimates but, hopefully, do generate likelihoods that discriminate the states better. For the correct phoneme recognition the selection of the correct state sequence is, after all, more important than the correct probabilities of different state sequences.

It is uncertain and rarely studied that in which phase of the model training the discrimination should be applied. The best solution might be that the ML approach could be entirely abandoned and discriminative training would be used alone for the model estimation. However, the discriminative methods presented so far for the CDHMM training require quite good initial parameter values to be really effective and the normal procedure is to use ML estimation for model initialization.

As pointed out earlier in section 3.1 the initialization of the mean vectors affect considerably to the ML estimation when long observation vectors and a large number of mixture densities lead to a large number of parameters to be estimated. So it becomes intuitively appealing to test whether a discriminative initialization might guide the ML estimation towards parameter values with good discrimination between models.

4.2 LVQ

4.2.1 Minimization of the classification error

When multiple codebook vectors are used to represent each class of the observed feature vectors, the optimal training of the codebook vectors is an essentially different problem as that of classical vector quantization as stated in section 3.2.1. The standard approach to the minimization of the misclassification leads to the Bayesian decision theory, according to which each sample $\boldsymbol{x}$ is classified into the class C_k of highest likelihood $p(\boldsymbol{x}|\boldsymbol{x} \in C_k)P(C_k)$. Here, $P(C_k)$ is the prior probability of class C_k.

Instead of the direct estimation of the conditional probability density $p(\boldsymbol{x}|\boldsymbol{x} \in C_k)$ for each class the LVQ approach is to generate a set of codebook vectors suitably labeled using the class labels. The classification is then performed by choosing the label of the nearest codebook vector selected according to 3.2. The most accurate representation is then required in the border areas between the classes. This method minimizes well the average rate of misclassifications if the codebook vectors and their labels are selected so that the Nearest Neighbor rule 3.2 minimizes the expected misclassification probability.

4.2.2 LVQ algorithms

Training by any of the following LVQ-variations (Kohonen, 1990b; Kohonen et al., 1992) is adaptive in such a way that the training samples can be presented one at a time in a random order and the codebook absorbs the inherent statistical features of the training data set. The policy is to create codebooks with performance arbitrarily near the optimal misclassification rate by avoiding local optimas and complex gradient calculations by using small enhancement step in locally correct directions. In this way LVQ is an iterative probabilistic gradient method with guaranteed asymptotic minimization of the average classification error.

LVQ1

For a randomly selected training vector $\boldsymbol{x}$, the nearest codebook vector $\boldsymbol{m}_c$ [1] is updated by LVQ1 (Kohonen, 1986b) as follows:

- If the training vector $\boldsymbol{x}$ and its nearest codebook vector $\boldsymbol{m}_c$ belong to the same class

$$\boldsymbol{m}_c(t+1) = \boldsymbol{m}_c(t) + \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_c(t)]. \tag{4.1}$$

- If they belong to different classes, the update occurs to the opposite direction

$$\boldsymbol{m}_c(t+1) = \boldsymbol{m}_c(t) - \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_c(t)]. \tag{4.2}$$

The teaching gain $\alpha(t) \in (0,1)$ is decreased monotonically.

[1] The notation here follows mainly that of (Kohonen et al., 1992).

LVQ2 and LVQ2.1

In LVQ2 the *two* best-matching codebook vectors representing different classes are adjusted, if the sample vector $\boldsymbol{x}$ appears to be near the border between two classes. The original version of LVQ2 (Kohonen, 1988a) restricts further that the nearest codebook vector $\boldsymbol{m}_c$ must belong to a different class than $\boldsymbol{x}$ and the second-nearest $\boldsymbol{m}'_c$ to the same class as $\boldsymbol{x}$. The version LVQ2.1 ((Kohonen, 1990a; Kohonen, 1990c)) only requires only that either $\boldsymbol{m}_c$ or $\boldsymbol{m}'_c$ should belong to the correct class and the other to an incorrect class.

Under these conditions the update by LVQ2 occurs as follows:

$$\begin{aligned} \boldsymbol{m}_c(t+1) &= \boldsymbol{m}_c(t) - \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_c(t)] \\ \boldsymbol{m}'_c(t+1) &= \boldsymbol{m}'_c(t) + \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}'_c(t)] \ . \end{aligned} \tag{4.3}$$

LVQ2.1 is similar except that if $\boldsymbol{m}_c$ is the one having the correct label and $\boldsymbol{m}'_c$ the incorrect the update is also performed but naturally in reversed directions

$$\begin{aligned} \boldsymbol{m}_c(t+1) &= \boldsymbol{m}_c(t) + \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_c(t)] \\ \boldsymbol{m}'_c(t+1) &= \boldsymbol{m}'_c(t) - \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}'_c(t)] \ . \end{aligned} \tag{4.4}$$

The confirmation that $\boldsymbol{x}$ is near the border between these two classes is inspected by defining a window of relative width $w \in (0,1)$ so that the ratio of the distances $d_c = ||\boldsymbol{x} - \boldsymbol{m}_c||$ and $d'_c = ||\boldsymbol{x} - \boldsymbol{m}'_c||$ must fulfill

$$\frac{d_c}{d'_c} > \frac{1-w}{1+w} \ . \tag{4.5}$$

LVQ3

LVQ3 (Kohonen, 1990b) is a combination of LVQ1 and LVQ2 (or LVQ2.1) having a good long run behavior. The codebook updates are similar as for LVQ2.1 extended for cases where all $\boldsymbol{x}, \boldsymbol{m}_c$ and $\boldsymbol{m}'_c$ belong to the same class. For such cases the update rule is

$$\begin{aligned} \boldsymbol{m}_c(t+1) &= \boldsymbol{m}_c(t) + \epsilon\alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_c(t)] \\ \boldsymbol{m}'_c(t+1) &= \boldsymbol{m}'_c(t) + \epsilon\alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}'_c(t)] \ , \end{aligned} \tag{4.6}$$

where $\epsilon \in (0,1)$ is a stabilizing constant factor. The value of ϵ should reflect the width of the window w in equation 4.5 so that with a narrow window the stabilizing learning steps (equation 4.6) are small (i. e. ϵ is small).

OLVQ1

OLVQ1 (Kohonen et al., 1992; Kohonen, 1992) is LVQ1 with optimized learning rates $\alpha_i(t)$ defined individually for each $\boldsymbol{m}_i$. The learning rates are optimized in the sense of maximal statistical accuracy, which means that all the training samples contribute by corrections of equal magnitudes.

It can be shown (Kohonen et al., 1992) that the optimal learning rates are achieved by adjusting the learning rate of the best-matching unit $\alpha_c(t)$ in each update step by

$$\alpha_c(t) = \frac{\alpha_c(t-1)}{1 + s(t)\alpha_c(t-1)} \quad , \tag{4.7}$$

where $s(t) = 1$, if the classification by $\boldsymbol{m}_c$ is correct and $s(t) = -1$ otherwise. In practice, it can be necessary to set an upper bound to $\alpha_c(t)$ so that it would not rise above 1 in the case of several sequential misclassifications.

4.2.3 Codebook initialization

The LVQ can be used as a fine tuning method after organizing the codebook vectors by SOM or K-means, depending on the application, but the codebook vectors of LVQ can also be initialized by finding a group of vectors which satisfy the K-nearest neighbors (KNN) criterion as suggested in (Kohonen et al., 1992). The KNN criterion states that from the K-nearest neighbors in the training data the majority must belong to the same class as the tested vector.

4.2.4 Other variations

Inspired by the success of LVQ2 (Kohonen et al., 1988a) several variations have been proposed by different authors, e. g. (McDermott, 1990; Makino et al., 1992), to relax the rather strict learning conditions defined by LVQ2. By these mainly intuitively motivated modifications the learning can be made to perform faster which can be useful with poor initialization.

In (Makino et al., 1992) the LVQ2 is modified so that it is no longer necessary that the codebook vector with the correct label belongs to the two closest codebook vectors. The condition for updates is that it belongs to the set of N closest codebook vectors, where the value of N can be determined suitably to achieve the requested learning behavior. Another new parameter is suggested to control the number of incorrectly labeled codebook vectors (instead of only the closest one) that are pushed away from the sample vector $\boldsymbol{x}$.

The LVQ2 variation suggested in (McDermott, 1990) is to completely reject the condition that any codebook vector with the correct label should be near the sample vector. The update is done according to equation 4.3 with the modification of notation that $\boldsymbol{m}'_c$ is still the nearest codebook vector among those of the correct label but not necessarily the second-nearest in the whole codebook.

4.3 Using LVQ for density initialization of HMMs

4.3.1 Motivation

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

After applying SOMs for the density initialization of CDHMMs as described in the previous chapter (section 3.4) it is intuitively appealing to assume that LVQ could be applied likewise. With LVQ the mixture means could be placed in order to enhance the discriminatory power of the CDHMM phoneme models. As the LVQ algorithms (Kohonen, 1990b) approximate the optimal Bayes decision boundaries between classes in Nearest Neighbor classification by dividing the vector space using a set of reference vectors, the resulting situation is that the quantization accuracy will be very high near the class boundaries. When the quantization errors are small, the conditional observation probabilities obtained by Gaussians centered using the LVQ reference vectors will be high providing thus clearly distinctive probability values for different classes even near the class boundaries.

4.3.2 Implementation issues

CDHMMs

In CDHMMs the classes roughly correspond to the states. Instead of one large codebook there will be many small codebooks, one for each state, because for CDHMMs there will be a unique set of Gaussians for each state. This, however, does not have to affect the LVQ training, because the codebook vectors can be still treated and updated like they would form a single large codebook. It is not even necessary, although advisable at least initially, to have an equal number of codebook vectors for each state.

Since the boundaries of different states inside a HMM are quite unstable and since the correctness of a recognized word is decided solely based on the labels of HMMs visited by the best path without any information about the state transition inside the HMMs, the update decisions by LVQ can be well approximated by comparing only HMM labels. For example, when a feature vector with label /A/ is considered, the exact state of the nearest codebook vector does not matter as long as it belongs to a HMM with label /A/.

Since LVQ is a supervised training method, a sufficiently large set of preclassified training vectors are required. The codebook vectors can then be adaptively modified by circulating the training samples on at a time and updating the nearest codebook vectors towards better positions according to the class of the training vector. The vector to be updated corresponding to a training vector is selected by finding the closest Gaussian mean vector in the group of all mixture components in all state output density functions (Kurimo &

Torkkola, 1992c; Kurimo & Torkkola, 1992a). After LVQ the codebook vectors are used as the mean vectors of the Gaussian density functions.

SCHMMs

As suggested in (Kurimo, 1993) the LVQ training of the Gaussians mean vectors is also applicable for the SCHMMs and leads to lower recognition error rates than initializations by only SOMs or K-means. In the same way as for SOMs (section 3.4), the main implementational difference for LVQ between CDHMMs and SCHMMs is that in SCHMMs the Gaussians are tied between all states leading to a single large codebook featuring all the states.

The problem in having a common tied Gaussian codebook is how to perform LVQ training when all the codebook vectors actually have the all the labels, because each of them belong to each class. The solution (Kurimo, 1993) relies on the fact that since only a couple of nearest Gaussians to an observed feature vector will have outputs significantly different from zero, only those will, in practice, affect to the density estimation. So the majority voting, where each labeled training vector gives its vote to the nearest codebook vector, can be applied to determine the *main* labels for the codebook. These *main* labels are used in LVQ training, which can then be applied normally as explained previously in this chapter (in section 4.2.2).

Chapter 5

PARAMETER ESTIMATION

5.1 Maximum likelihood training

5.1.1 Baum-Welch reestimation

With the HMM defined by $\lambda = (\pi, A, B)$ [1] in (2.8) the ML principle can then be formulated as selecting λ in order to maximize $\Pr(\boldsymbol{O}|\lambda)$ for the given training sequence $\boldsymbol{O}$. The Baum-Welch reestimation updates an existing model λ' by a two-step procedure to a new model λ (Baum & Petrie, 1966). In short the first step is to determine the state probabilities in the observation sequence and the second is to estimate the model parameters using the observations weighted by the state probabilities. A similar estimation idea has also been presented by Dempster (Dempster et al., 1977), but without HMM framework.

The theoretic convergence of the algorithm can be studied by defining an auxiliary function based on the Kullback-Leibler statistic (Kullback & Leibler, 1951). The auxiliary function $Q(\lambda', \lambda)$ is a kind of measure of the divergence between the initial model λ' and an updated model λ (Liporace, 1982).

$$Q(\lambda', \lambda) = \sum_{\boldsymbol{q}} \Pr(\boldsymbol{O}, \boldsymbol{q}|\lambda') \log \Pr(\boldsymbol{O}, \boldsymbol{q}|\lambda) \tag{5.1}$$

The $\boldsymbol{q}$ is a possible "hidden" state sequence behind the observations.

In (Liporace, 1982) it was shown that since $Q(\lambda', \lambda) \geq Q(\lambda', \lambda')$ implies that $\Pr(\boldsymbol{O}|\lambda) \geq \Pr(\boldsymbol{O}|\lambda')$ the likelihood $\Pr(\boldsymbol{O}|\lambda)$ can be increased optimally by maximizing the auxiliary function $Q(\lambda', \lambda)$ over λ. It can be further proved that the $Q(\lambda', \lambda)$ has a unique global maximum as a function of λ, and this maximum is the one and the only critical point. This proof involves the assumption that the observation distributions are mixtures of strictly log-concave or elliptically symmetric multivariate distributions (Juang, 1985). The widely Gaussian mixture density functions naturally fall into such a category.

The reestimates can be explicitly derived as the critical point of Q for a fixed λ' by solving

[1] The notation here follows mainly that of (Juang & Rabiner, 1991).

the augmented equations using Lagrange multipliers (Liporace, 1982). The resulting estimation formulas for the case of Gaussian mixture density functions as output distributions (equations 2.9 and 2.15) are as follows, where the summation over L denotes involving all the training samples:

$$\hat{\boldsymbol{\mu}}_{im} = \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\gamma_t^l(i,m)\boldsymbol{O}_t^l}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\gamma_t^l(i,m)} \tag{5.2}$$

$$\hat{\boldsymbol{\Sigma}}_{im} = \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\gamma_t^l(i,m)(\boldsymbol{O}_t^l-\boldsymbol{\mu}_{im})(\boldsymbol{O}_t^l-\boldsymbol{\mu}_{im})^T}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\gamma_t^l(i,m)} \tag{5.3}$$

$$\hat{c}_{im} = \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\gamma_t^l(i,m)}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\sum_{m=1}^{M}\gamma_t^l(i,m)} \tag{5.4}$$

$$\hat{a}_{ij} = \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\xi_t^l(i,j)}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\gamma_t^l(i)} \tag{5.5}$$

where the weighting factor $\gamma_t^l(i,m)$ is the probability that the system is in state i at time t in training sample l calculated by the mixture component m. The ranges of the indexes are:

$$\begin{aligned} i &= \text{Every state in every HMM} \\ j &= \text{Every possible state that can be visited after } i \\ m &= 1,\cdots,M \quad \text{(Number of mixtures)} \\ t &= 1,\cdots,T_l \quad \text{(Length of training sample)} \\ l &= 1,\cdots,L \quad \text{(Number of training samples)} \end{aligned}$$

The state probabilities γ can be conveniently calculated via the following forward-backward procedure:

The forward probability $\alpha_t(j)$ for one training sample is the probability that state j is reached at time t after the sequence of observations $\boldsymbol{O}_1, \boldsymbol{O}_2, \cdots, \boldsymbol{O}_t$:

$$\alpha_t(j) = b_j(\boldsymbol{O}_t)\sum_{i=1}^{N}\alpha_{t-1}(i)a_{ij}\ , \quad t=1,\cdots,T \quad j=1,\cdots,N \tag{5.6}$$

and $\alpha_0(j) = \pi_j$.

Similarly, the backward probability $\beta_t(i)$ is defined as the probability that starting from the state i at time t the sequence $\boldsymbol{O}_{t+1}, \boldsymbol{O}_{t+2}, \cdots, \boldsymbol{O}_T$ is observed:

$$\beta_t(i) = \sum_{j=1}^{N}a_{ij}b_j(\boldsymbol{O}_{t+1})\beta_{t+1}(j)\ , \quad t=T-1,T-2,\cdots,1 \quad i=1,\cdots,N \tag{5.7}$$

and $\beta_T(i) = 1.0$, if i is an acceptable final state and 0 otherwise.

Using the α and β the branch probability $\xi_t(i,j)$, i. e. being in state i at the time t and moving next to state j, can be calculated.

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(\boldsymbol{O}_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(\boldsymbol{O}_{t+1})\beta_{t+1}(j)} \tag{5.8}$$

The state probabilities $\gamma_t(i)$ follow easily from the results above as $\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$, or directly

$$\gamma_t(i) = \frac{\alpha_t(i) \sum_{j=1}^{N} a_{ij} b_j(\boldsymbol{O}_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_t(i) \sum_{j=1}^{N} a_{ij} b_j(\boldsymbol{O}_{t+1}) \beta_{t+1}(j)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{5.9}$$

If mixtures of M density functions are used instead of a single density function, the total observation probabilities are calculated as

$$b_i(\boldsymbol{O}_t) = \sum_{m=1}^{M} c_{im} b_{im}(\boldsymbol{O}_t) \quad , \tag{5.10}$$

where $b_{im}(\boldsymbol{O}_t)$ is the observation probability by the component m. Thus the components $\gamma_t(i,m)$ of the total state probabilities $\gamma_t(i)$ are

$$\gamma_t(i,m) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \frac{c_{im} b_{im}(\boldsymbol{O}_t)}{b_i(\boldsymbol{O}_t)} \tag{5.11}$$

The new parameter values λ are then applied as λ' to redefine the Q which can be in turn maximized again. This two-step procedure iterates until some criterion concerning either the difference between the successive models or the iteration number is met.

5.1.2 Viterbi training

Estimation of HMM parameters by Viterbi training (also called Segmental K-means algorithm (Rabiner et al., 1986)) is a slightly different variation of the MLE principle.

The objective in Viterbi training is to maximize the so-called state-optimized likelihood of the observation sequence

$$L(\bar{\boldsymbol{q}}) = \max_q \Pr(\boldsymbol{O}, \boldsymbol{q}|\lambda) \quad , \tag{5.12}$$

where

$$\bar{\boldsymbol{q}} = \arg\max_q \Pr(\boldsymbol{O}, \boldsymbol{q}|\lambda) \quad . \tag{5.13}$$

This criterion differs from the Baum-Welch criterion so that instead of the probability over all possible state sequences generated by the model λ only the probability of the most probable state sequence is used.

As can be easily verified by taking logarithms inside the maximization

$$\begin{aligned} \max_\lambda \max_q \log \Pr(\boldsymbol{O}, \boldsymbol{q}|\lambda) &= \max_\lambda \max_q [\log \Pr(\boldsymbol{O}|\boldsymbol{q}, \lambda) + \log \Pr(\boldsymbol{q}|\lambda)] && (5.14) \\ &= \max_\lambda [\log \Pr(\boldsymbol{O}|\bar{\boldsymbol{q}}, \lambda) + \log \Pr(\bar{\boldsymbol{q}}|\lambda)] \quad , && (5.15) \end{aligned}$$

the maximization can be divided into two terms that can be separately optimized since the latter depends only on transition probabilities A and the former on observation probabilities B of the model λ. This leads to a more straightforward implementation with less computation and numerical difficulties that could be caused by very low weighting probabilities appearing in long training sequences.

The most probable state sequence $\bar{\boldsymbol{q}}$ for each training sequence can be conveniently computed by the Viterbi search (see section 2.2.3) based on dynamic programming (Forney, 1973). After the best paths are found for the model λ the path probabilities are no longer needed since the new parameters can be directly calculated by averaging without weights. The iteration formulas for the maximization step are similar as in Baum-Welch reestimation 5.2, 5.3, 5.4 and 5.5 except that the state probabilities γ and branch probabilities ξ are replaced by indicator functions η and ζ

$$\hat{\boldsymbol{\mu}}_{im} = \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\eta_t^l(i,m)\boldsymbol{O}_t^l}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\eta_t^l(i,m)} \tag{5.16}$$

$$\hat{\boldsymbol{\Sigma}}_{im} = \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\eta_t^l(i,m)(\boldsymbol{O}_t^l-\boldsymbol{\mu}_{im})(\boldsymbol{O}_t^l-\boldsymbol{\mu}_{im})^T}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\eta_t^l(i,m)} \tag{5.17}$$

$$\hat{c}_{im} = \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\eta_t^l(i,m)}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\sum_{m=1}^{M}\eta_t^l(i,m)} \tag{5.18}$$

$$\hat{a}_{ij} = \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\zeta_t^l(i,j)}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\eta_t^l(i)} \quad , \tag{5.19}$$

where the indicator functions $\eta_t(i,m)$ and $\zeta_t(i,j)$ are defined simply using the most probable state sequence $\bar{q}$ for each training sequence as

$$\eta_t(i) = \begin{cases} 1 & \text{if } \bar{q}_t = i \\ 0 & \text{otherwise} \end{cases} \tag{5.20}$$

$$\eta_t(i,m) = \begin{cases} 1 & \text{if } \bar{q}_t = i \text{ and } \boldsymbol{\mu}_{im} = \boldsymbol{\mu}_{ic} \\ 0 & \text{otherwise} \end{cases} \tag{5.21}$$

$$\zeta_t(i,j) = \begin{cases} 1 & \text{if } \bar{q}_t = i \text{ and } \bar{q}_{t+1} = j \\ 0 & \text{otherwise} \end{cases} \quad . \tag{5.22}$$

The Gaussian mean vector $\boldsymbol{\mu}_{ic}$ is the closest Gaussian mean vector of the state i compared to observation vector $\boldsymbol{O}_t$. The ranges of the indexes are:

$$\begin{aligned} i &= \text{Every state in every HMM} \\ j &= \text{Every possible state visited after } i \\ m &= 1,\cdots,M \quad \text{(Number of mixtures)} \\ t &= 1,\cdots,T_l \quad \text{(Length of training sample)} \\ l &= 1,\cdots,L \quad \text{(Number of training samples)} \end{aligned}$$

The new parameter values λ are then used to compute the most probable paths $\bar{\boldsymbol{q}}$ again and the next iteration of the parameter values is performed. The process can be iterated in the same way as in the Baum-Welch reestimation until the stopping criterion is fulfilled. Despite of being an approximation of the Baum-Welch reestimation the Viterbi training has, however, the same convergence properties as the Baum-Welch as shown in (Juang & Rabiner, 1990).

5.1.3 Problems

Training objective

ML training is not guaranteed to minimize the recognition error rate; it provides only indirect means of doing so.

Independence of observations

The joining of observation probabilities of sequential observations as proposed in equation (2.6) involves an implicit assumption of the independence of neighboring events. In practice this is often not exactly valid since the features are spectral coefficients computed from partially overlapping time windows. When applying HMMs for speech signal correlations between several successive samples exist, since the changes in the human articulatory apparatus occur relatively slowly.

Mismatch of model family

Although the states of HMMs are modeled as Gaussian mixture sources, they certainly are not such sources in applications like speech recognition. Also for correlating observations the Markov assumption, i. e. the state transition probabilities depend only on one previous state, is invalid. This false assumption leads to the model mismatch phenomenon, which guarantees the non-optimality of ML training despite the fact that it may, of course, produce good approximative models.

Numerical difficulties

The calculations of the conditional probabilities of observation sequences involve a broad dynamical range, because the results may vary between $10^{-10} \cdots 10^{-100}$. A considerable numerical accuracy is therefore required and some normalizations are necessary during the computations to avoid too small numbers.

Number of parameters

In a speech recognition application the HMMs are created for, for example, 20 phonemes and each model may have e. g. 4 states. Then using mixtures of 10 Gaussians for each state results to huge number of unknown parameters, which should be estimated.

5.2 Maximum mutual information training

As proposed in (Bahl et al., 1986) the parameters of HMMs can be estimated in order to maximize the mutual information (MMI) between an acoustic observation sequence $\mathbf{O}$ and and the corresponding sequence of class labels C:

$$I(C, \boldsymbol{O}) = \sum_i \log \frac{\Pr(\boldsymbol{O}, C)}{\Pr(\boldsymbol{O}) \Pr(C)} \quad . \tag{5.23}$$

By maximizing the mutual information $I_\Lambda(C, \boldsymbol{O})$ the probability of the correct state sequence given the training data will be maximized. This estimation principle is called the conditional maximum likelihood estimation (CMLE) (Nadas et al., 1988). To formulate the CMLE for the set of L HMMs $\Lambda = \{\lambda_i\}_{i=1}^L$ of the corresponding classes $\{C_i\}_{i=1}^L$ the following probability measure is used:

$$\Pr_\Lambda(\boldsymbol{O}) = \sum_i \Pr_\Lambda(\boldsymbol{O}|C_i) \Pr_\Lambda(C_i) \quad . \tag{5.24}$$

The prior probability $\Pr^*_\Lambda(C_i) = \Pr_0(C_i)$ is determined independent of the observed training data set $\{\boldsymbol{O}^{(i)}\}$, where the subset $\boldsymbol{O}^{(i)}$ contains samples that are labeled as C_i. The CMLE can be now expressed as

$$\Lambda^*_{CMLE} = \arg\max_\Lambda \prod_i \Pr_\Lambda(C_i|\boldsymbol{O}^{(i)}) \tag{5.25}$$

This maximization is equivalent to the maximization of the information provided by $\mathbf{O}$ about C.

$$\Lambda^*_{MMI} = \arg\max_\Lambda \sum_i \log \frac{\Pr_\Lambda(\boldsymbol{O}^{(i)}, C_i)}{\Pr_\Lambda(\boldsymbol{O}^{(i)}) \Pr_0(C_i)} \tag{5.26}$$

For the implementation of the MMI (or CMLE) training there has not yet been presented such an efficient and robust procedure similar to the Baum-Welch reestimation (or Viterbi training) that is guaranteed to converge to an optimal solution. In (Bahl et al., 1986; Merialdo, 1988) the optimization problem is solved using gradient search techniques with projection on the constraints.

A related information theoretically justified method is called minimum discrimination information (MDI) approach (Epraim et al., 1989). In MDI the goal of the HMM design is choose parameter values which minimize the discrimination information between the set of signal probability densities $q()$ and the set of HMM probability densities $p(|\lambda)$

$$D(Q : P) = \int q(\boldsymbol{O}) \log \frac{q(\boldsymbol{O})}{p(\boldsymbol{O}|\lambda)} d\boldsymbol{O} \; . \tag{5.27}$$

The MDI approach is more general than the ML and MMI, but as reported in (Epraim et al., 1989) it has some computational difficulties and, in practice, several approximations are necessary to reduce the complexity of the problem. However, there seems to be no evidence that the MDI approach could bring significant improvements to the ML and MMI (Epraim et al., 1989).

5.3 Corrective training

5.3.1 Discrete output density

The corrective training for discrete output HMMs was first proposed by Bahl et al. (Bahl et al., 1988). The method was motivated by training experiments and intuition rather than theoretical convergence proofs. In experiments reported also by other authors different versions of the corrective training led to significantly fewer recognition errors than ML or MMI estimations. The corrective training presented in this section is the original version (Bahl et al., 1988).

The corrective training is appealing because it aims directly at the maximization of the recognition performance with the training data. It does not assume that the model is of correct type and no complicated gradient calculations are involved. The method simply tries to recognize the training words and when an error is observed, modifies the parameters in order to avoid this recognition error. The convergence of this method is not proved but in practice it appears to work very well. However, choosing parameters that minimize the error rate on training data, do not, in general, minimize the error rate on test data. Therefore it is necessary to have a large training database as representative as possible of the intended application.

The corrective training algorithm proposed in (Bahl et al., 1988) has two tuning parameters β and δ, which affect the rate of convergence and the extent to which the correct and incorrect probabilities are driven apart.

1. Use MLE to obtain the initial parameter values of the HMMs $\Lambda = \lambda_j$ and the approximative frequencies $\hat{b}(y, j)$ for each label y and each state j.

2. For each utterance $\boldsymbol{O}$ labeled as C_c compute the probabilities $\Pr(\boldsymbol{O}|\lambda_c)$ and $\Pr(\boldsymbol{O}|\lambda_i)$ for the correct class C_c and for each incorrect class C_i using the initial model λ. (In practice, the $\Pr(\boldsymbol{O}|\lambda_i)$:s are computed only to some confusable classes.) Choose incorrect words satisfying the condition $D(c, i) < \delta$, where $D(c, i) = \log \Pr(\boldsymbol{O}|\lambda_c) - \log \Pr(\boldsymbol{O}|\lambda_i)$.

3. Apply one iteration of MLE to obtain estimates $\hat{b}_{C_c}(y, j)$ and $\hat{b}_{C_i}(y, j)$ using the labeled utterances $\boldsymbol{O}$ only, for the correct word C_c and each incorrect word C_i respectively. Update
$$\hat{b}^{new}(y, j) = \hat{b}^{old}(y, j) + \gamma(\hat{b}_{C_c}(y, j) - \hat{b}_{C_i}(y, j)) \ , \tag{5.28}$$
where
$$\gamma = \begin{cases} \beta & \text{if } D(c, i) \leq 0 \\ \beta(1 - D(c, i)/\delta) & \text{if } 0 < D(c, i) < \delta \end{cases} . \tag{5.29}$$

4. If any adjustments were made, replace any negative frequencies $\hat{b}(y, j)$ by 0, recompute the parameters λ and return to Step 2.

The adjustments in Step 3 are designed to increase the probability $\Pr(\boldsymbol{O}|\lambda_c)$ and decrease the probabilities $\Pr(\boldsymbol{O}|\lambda_i)$.

5.3.2 Continuous output density

A similar approach for the training of continuous density HMMs has been proposed by Mizuta et al. (Mizuta & Nakajima, 1990; Mizuta & Nakajima, 1992) called optimal discriminative training (ODT). In this method the Gaussian mean vectors of the continuous mixture output densities of the HMMs are modified to minimize the misrecognition and near-misrecognition rates for the training data. As for the error corrective training by Bahl et al. (Bahl et al., 1988) no rigorous analytical treatment of the method is provided.

For one misrecognized or nearly misrecognized training sample only two HMMs are modified by ODT: the correct one and the most probable incorrect one. For each frame of the sample the corresponding states c and i of these two HMMs are picked from the Viterbi paths. The most probable Gaussian densities (indexed by k and l) of these states according to the feature vector of the current frame $\boldsymbol{x}(t)$ are updated by moving their mean vectors $\boldsymbol{\mu}_{ck}$ and $\boldsymbol{\mu}_{il}$ as follows:

$$\begin{aligned} \boldsymbol{\mu}_{ck}^{new} &= \boldsymbol{\mu}_{ck}^{old} + \gamma(\boldsymbol{x}(t) - \boldsymbol{\mu}_{ck}^{old}) \\ \boldsymbol{\mu}_{il}^{new} &= \boldsymbol{\mu}_{il}^{old} - \gamma(\boldsymbol{x}(t) - \boldsymbol{\mu}_{il}^{old}) \ , \end{aligned} \tag{5.30}$$

where

$$\gamma = \begin{cases} \beta & \text{if } D(c,i) \leq 0 \\ \beta(1 - D(c,i)/\delta) & \text{if } 0 < D(c,i) < \delta \end{cases} \tag{5.31}$$

similarly as for discrete HMMs 5.29. (Actually, in (Mizuta & Nakajima, 1990) the probability difference was defined without taking logarithms of the probabilities as $D(c,i) = \Pr(\boldsymbol{O}|\lambda_c) - \Pr(\boldsymbol{O}|\lambda_i)$). These adjustments tend also to increase the probability $\Pr(\boldsymbol{O}|\lambda_c)$ and decrease the probabilities $\Pr(\boldsymbol{O}|\lambda_i)$.

The training is performed for each misrecognized (or nearly misrecognized) sample until some convergence is observed for the training data set.

5.4 Generalized probabilistic descent method

5.4.1 Definitions

The following GPD formalization presented in (Katagiri et al., 1991; Komori & Katagiri, 1992) is described here for convenience, because it provides a way to generalize a group of probabilistic descent estimation methods to minimize classification errors in the same manner as, e. g. LVQ does.

The M-class classifier is assumed to consist of a set of reference vectors $\Lambda = \lambda_m$, $m = 1, \cdots, M$ where each class C_m has several representative patterns $\lambda_m = \{\boldsymbol{r}_m^b\}$, $b = 1, \cdots, B_m$ ordered so that the $\boldsymbol{r}_m^b$ is the b-th closest variable duration sequence of acoustic feature vectors to the sequence of observations $\boldsymbol{x}$.

A measure of the degree to which the observation sequence $\boldsymbol{x}$ belongs to class C_m is called the *class distance* $g_m(\boldsymbol{x};\Lambda)$ and is defined by

$$g_m(\boldsymbol{x};\Lambda) = [\sum_{b=1}^{B_m}\{D(\boldsymbol{x},\boldsymbol{r}_m^b)\}^{-\zeta}]^{-1/\zeta} . \tag{5.32}$$

ζ is a positive number.

A distance called the *reference distance* between $\boldsymbol{x}$ and $\boldsymbol{r}_m^b$ is defined by

$$D(\boldsymbol{x},\boldsymbol{r}_m^b) = [\sum_{\theta=1}^{\Theta_m}\{D_\theta(\boldsymbol{x},\boldsymbol{r}_m^b)\}^{-\xi}]^{-1/\xi} . \tag{5.33}$$

ξ is a positive number.

The $D_\theta(\boldsymbol{x},\boldsymbol{r}_m^b)$ denotes a *path distance* computed by the θ-th best path between $\boldsymbol{x}$ and $\boldsymbol{r}_m^b$, which can be further decomposed as

$$D_\theta(\boldsymbol{x},\boldsymbol{r}_m^b) = \sum_{t=1}^{T} w_{m,t}^b \delta_{m,\theta_t}^b . \tag{5.34}$$

Here, $w_{m,t}^b$ is a weighting factor corresponding to the t-th frame and δ_{m,θ_t}^b is a *local* distance between the t-th frame of $\boldsymbol{r}_m^b$ and the corresponding frame of $\boldsymbol{x}$ along the θ-th best path.

The local distance can be measured, for example, by the Euclidean distance

$$\delta_{m,\theta_t}^b = \sum_{s=1}^{S}(\boldsymbol{r}_{m,t,s}^b - \boldsymbol{x}_{\theta_t^{m,b},s})^2 , \tag{5.35}$$

where $\boldsymbol{r}_{m,t,s}^b$ and $\boldsymbol{x}_{\theta_t^{m,b},s}$ are the s-th elements of $\boldsymbol{r}_{m,t}^b$ and $\boldsymbol{x}_{\theta_t^{m,b}}$ respectively, and $\boldsymbol{x}_{\theta_t^{m,b}}$ is the frame of $\boldsymbol{x}$ corresponding to $\boldsymbol{r}_{m,t}^b$ along the θ-th best path.

The classification is performed simply according to the following rule:

$$C(\boldsymbol{x}) = C_i \text{ , if } i = \arg\min_j g_j(\boldsymbol{x};\Lambda) . \tag{5.36}$$

A measure of misclassification of $\boldsymbol{x} \in C_k$ is defined by

$$d_k(\boldsymbol{x}) = g_k(\boldsymbol{x};\Lambda) - [\frac{1}{M-1}\sum_{j,j\neq k} g_j(\boldsymbol{x};\Lambda)^{-1/\mu}]^{-\mu} . \tag{5.37}$$

μ is a positive number. As expected, the larger $d_k(\boldsymbol{x})$ the more definite misclassification.

The minimum classification error criterion is properly approximated by a monotonically increasing, differentiable cost function defined by

$$l_k(\boldsymbol{x};\Lambda) = l_k(d_k(\boldsymbol{x};\Lambda)) = \frac{1}{1+e^{-\alpha(d_k(\boldsymbol{x};\Lambda)+\beta)}} \text{ , } \alpha > 0 . \tag{5.38}$$

5.4.2 Adjustment step

It has been proved (Amari, 1967) that adjusting Λ by $-\epsilon U \nabla l_k(\boldsymbol{x};\Lambda)$, where ϵ is a small positive number and U a positive definite matrix, leads to at least a locally optimal minimum error classifier in a probabilistic sense.

The adjustment step for the reference pattern $\boldsymbol{r}^b_{m,t,s}$ given in (Komori & Katagiri, 1992) is

$$\begin{aligned} \boldsymbol{r}^b_{m,t,s}(n+1) &= \boldsymbol{r}^b_{m,t,s}(n) - 2\epsilon_n v_k w^b_{m,t} \Phi_m \Psi_m \tau_m \,, \quad \text{for } m = k \,, \\ \boldsymbol{r}^b_{m,t,s}(n+1) &= \boldsymbol{r}^b_{m,t,s}(n) + 2\frac{\epsilon_n v_k w^b_{m,t}}{M-1} \kappa_m \Phi_m \Psi_m \tau_m \,, \text{for } m \neq k \,, \end{aligned} \tag{5.39}$$

where

$$v_k = l'_k(d_k(\boldsymbol{x};\Lambda)) = \alpha l_k(d_k(\boldsymbol{x};\Lambda))(1 - l_k(d_k(\boldsymbol{x};\Lambda))) \,, \tag{5.40}$$

$$\Phi_m = [\sum_{b'=1}^{B_m} \{\frac{D(\boldsymbol{x}, \boldsymbol{r}^b_m)}{D(\boldsymbol{x}, \boldsymbol{r}^{b'}_m)}\}^{\zeta}]^{-(1+\zeta)/\zeta} \,, \tag{5.41}$$

$$\Psi_m = [\sum_{\theta=1}^{\Theta} \{D_\theta(\boldsymbol{x}, \boldsymbol{r}^b_m)\}^{-\xi}]^{-(1+\xi)/\xi} \,, \tag{5.42}$$

$$\tau_m = \sum_{\theta=1}^{\Theta} \frac{\boldsymbol{r}^b_{m,t,s} - \boldsymbol{x}_{\theta^{m,b}_t, s}}{\{D_\theta(\boldsymbol{x}, \boldsymbol{r}^b_m)\}^{\xi+1}} \,, \tag{5.43}$$

$$\kappa_m = [\frac{1}{M-1} \sum_{j, j \neq k} \{\frac{g_m(\boldsymbol{x};\Lambda)}{g_j(\boldsymbol{x};\Lambda)}\}^{\mu}]^{-(1+\mu)/\mu} \,. \tag{5.44}$$

5.5 Minimum error classification training

A suggestion for implementation of the GPD for training CDHMMs to minimize the classification error rate is presented in (Rainton & Sagayama, 1992b). It includes some very useful simplifications for the definitions presented in the previous chapter which will also benefit the corrective LVQ based tuning presented in the next chapter (6).

The discriminant function of the GPD (5.32) can be as well defined logarithmically as

$$g_k(\boldsymbol{x};\Lambda) = \ln([\sum_{\text{all } \Theta} \{P(\boldsymbol{x}, \Theta)\}^{\xi}]^{1/\xi}) \,. \tag{5.45}$$

Θ is any allowable model/state sequence for Λ'th composite HMM. Letting $\xi \to \infty$ reduces the equation (5.45) to the likelihood of the best path computed by the Viterbi search.

$$g_k(\boldsymbol{x};\Lambda) = \ln(P(\boldsymbol{x}, \Theta)) \,. \tag{5.46}$$

(Here Θ is now only one path.)

The misclassification measure (5.37) can be also defined logarithmically

$$d_k(\boldsymbol{x}) = -g_k(\boldsymbol{x};\Lambda) + \ln([\frac{1}{M-1} \sum_{j, j \neq k} e^{\eta g_j(\boldsymbol{x};\Lambda)}]^{1/\eta}) \,. \tag{5.47}$$

This will simplify considerably, when $\eta \to \infty$ giving

$$d_k(\boldsymbol{x}) = -g_k(\boldsymbol{x};\Lambda) + g_j(\boldsymbol{x};\Lambda)\ , \tag{5.48}$$

where

$$j = \arg\max_{p,p\neq k} g_p(\boldsymbol{x};\Lambda)\ . \tag{5.49}$$

If it is preferred to update HMMs only using incorrectly classified utterances as e. g. in (Rainton & Sagayama, 1992b), this can be conveniently ensured by dropping the inequality off from equation (5.49) to get

$$d_k(\boldsymbol{x}) = -g_k(\boldsymbol{x};\Lambda) + g_j(\boldsymbol{x};\Lambda)\ , \tag{5.50}$$

where

$$j = \arg\max_{p} g_p(\boldsymbol{x};\Lambda)\ . \tag{5.51}$$

A simple gradient descent method to train the models by minimizing the cost function can be composed using the gradient of the cost. For the Gaussian mean vectors that would be

$$\boldsymbol{\mu}_{ik}(n+1) = \boldsymbol{\mu}_{ik}(n) - \epsilon(n)\frac{\partial l(\boldsymbol{x},\Lambda)}{\partial \boldsymbol{\mu}_{ik}(n)}\ . \tag{5.52}$$

For the mixture weights and covariance matrices similar iteration formulas can be easily constructed (Rainton & Sagayama, 1992b).

5.5.1 Differentiating the cost function

The profit from the previous simplifications is that the derivative of the cost function (5.38) will take a relatively simple and implementable form. The partial derivative of the cost function by the Gaussian mean vectors is formed here only to be able to make comparisons with the corresponding form of LVQ derived in the next section.

First the chain rule is applied as in (Rainton & Sagayama, 1992b) to split the cost into two terms.

$$\frac{\partial l(d(\boldsymbol{x},\Lambda))}{\partial \boldsymbol{\mu}_{ik}} = \frac{\mathrm{d}l(d(\boldsymbol{x},\Lambda))}{\mathrm{d}d(\boldsymbol{x},\Lambda)}\frac{\partial d(\boldsymbol{x},\Lambda)}{\partial \boldsymbol{\mu}_{ik}}\ , \tag{5.53}$$

where $\boldsymbol{\mu}_{ik}$ means the mean vector of the k'th mixture of the i'th state.

Substitution of equations (5.38), (5.51) and (5.46) and differentiation gives

$$\frac{\mathrm{d}l(d(\boldsymbol{x},\Lambda))}{\mathrm{d}d(\boldsymbol{x},\Lambda)} = \frac{\alpha[l(d(\boldsymbol{x},\Lambda))]^2}{e^{\alpha d(\boldsymbol{x},\Lambda)}} \tag{5.54}$$

$$\frac{\partial d(\boldsymbol{x},\Lambda)}{\partial \boldsymbol{\mu}_{ik}} = -\frac{\partial g_m(\boldsymbol{x},\Lambda)}{\partial \boldsymbol{\mu}_{ik}} + \frac{\partial g_j(\boldsymbol{x},\Lambda)}{\partial \boldsymbol{\mu}_{ik}}\ . \tag{5.55}$$

The joint probability of the observation sequence $\boldsymbol{x}$ was defined (in section 2.3) for mixture Gaussian density HMMs with diagonal covariance matrices by the set of equations

$$P(\boldsymbol{x}, \Theta|\Lambda) = \prod_{t=1}^{T_n} a_{\theta_{t-1}\theta_t} b_{\theta_t}(\boldsymbol{x}_{n,t}) \tag{5.56}$$

$$b_i(\boldsymbol{x}_{n,t}) = \sum_{k=1}^{M} w_{ik} N_{ik}(\boldsymbol{x}_{n,t}) \tag{5.57}$$

$$N_{ik}(\boldsymbol{x}_{n,t}) = \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\sigma_{ikd}^2}} e^{-\frac{(\boldsymbol{x}_{n,t,d}-\mu_{ikd})^2}{2\sigma_{ikd}^2}} \quad . \tag{5.58}$$

Thus the derivative of the discriminant function $g(\boldsymbol{x}, \Lambda)$ will be

$$\frac{\partial g(\boldsymbol{x}, \Lambda)}{\partial \boldsymbol{\mu}_{ik}} = \frac{\partial \ln P(\boldsymbol{x}, \Theta)}{\partial \boldsymbol{\mu}_{ik}} = \frac{\partial}{\partial \boldsymbol{\mu}_{ik}} \sum_{t=1}^{T_n} \ln(a_{\theta_{t-1}\theta_t} b_{\theta_t}(\boldsymbol{x}_{n,t})) \tag{5.59}$$

$$= \sum_{t=1}^{T_n} \frac{1}{b_{\theta_t}(\boldsymbol{x}_{n,t})} \frac{\partial b_{\theta_t}(\boldsymbol{x}_{n,t})}{\partial \boldsymbol{\mu}_{ik}} \tag{5.60}$$

$$\frac{\partial b_i(\boldsymbol{x}_{n,t})}{\partial \mu_{ikd}} = w_{ik} N_{ik}(\boldsymbol{x}_{n,t}) \frac{\boldsymbol{x}_{n,t,d} - \mu_{ikd}}{\sigma_{ikd}^2} \quad . \tag{5.61}$$

When using the Viterbi algorithm the equation (5.60) will be

$$\frac{\partial g(\boldsymbol{x}, \Lambda)}{\partial \boldsymbol{\mu}_{ik}} = \sum_{t=1}^{T_n} \delta(\theta_t - i) \frac{1}{b_{\theta_t}(\boldsymbol{x}_{n,t})} \frac{\partial b_{\theta_t}(\boldsymbol{x}_{n,t})}{\partial \boldsymbol{\mu}_{ik}} \tag{5.62}$$

The derivatives of the cost by the weights and the covariances can be computed in the same manner, if necessary.

Chapter 6

CORRECTIVE TUNING BY LVQ

6.1 Intuitive motivation

Error correction is intuitively quite an obvious way to enhance the recognition ability of the models. In training by error correction the training samples are first recognized and when an error is encountered the model is modified in order to achieve the correct transcription. The convergence of the model to 100% recognition accuracy in the training data is, however, not guaranteed because a modification caused by one training sample may force an error when recognizing another sample.

The way to achieve as high classification accuracy as possible is the application of probabilistic descent methods like LVQ in which the observed misclassifications are not fully corrected at once, but the models are only updated towards the locally correct decisions using a small step size. To facilitate a probabilistic convergence to a local minimum as good as possible the learning step size is gradually decreased towards zero and considerable efforts are made to have already very good initial models from which to start the learning.

However, if the training data covers insufficiently all the variability in phonemes as it usually, unfortunately, does in practical applications, because the availability of the data is limited, a too close adaptation to the training data may decrease the recognition accuracy for the independent test data. This common phenomen called *overlearning* is especially important in corrective tuning since by correcting the certain mistakes in training data it is very easy to loose generalization.

Instead of constructing an objective function to reflect the ability of the model to recognize the training data as done in GPD (section 5.4) and MEC training (section 5.5) the approach used here is based on the comparison between the most probable path through the states of the HMMs for a training sample and the most probable correct path (figure 6.1).

6.2 Description of the algorithm

6.2.1 Paths and segmentation

For the update of the CDHMMs using a particular features computed from a training word two different Markov model state sequences called paths are required. First the *most probable path* is computed just as it is done to recognize an unknown test word. The recognition is also proceeded to determine, if the word is correctly recognized, because in that case the tuning is omitted since it would be mostly unnecessary and waste of time.

If it is observed that the phonetic transcription of the training word will be incorrect, a so-called *correct path* is computed. The correct path is the most probable path visiting the correct phonemes in the right order thus producing the correct transcription. This path is obtained by constraining the Viterbi search so that in each phoneme to phoneme transition the next correct phoneme is the only possible choice. If a correct segmentation, like a manual one, is available, it is also possible to use that as a basis to construct the correct path.

```
OOOOOOOOOOUUUUUUUUUULLLLLLLAAAAAAAAAAEEEEEEEEEENNNEEEEEEEEEEEENNNNNNNNN
OOOOOOOOOOUUUUUUUUUULLLLLLLAAAAAAAAAAIIIIIIIIIINNNEEEEEEEEEEEENNNNNNNNN
O......U.............L.......A........I.......N.....E.........N........
```

Table 6.1: *The first row contains the labels of states visited by the most probable path through a part of a training sample of the word "OULAINEN". The second row is the most probable correct path obtained by the constrained Viterbi search. The last row presents the manually marked starting points of phonemes in that sample.*

The two paths described above in addition to a manual segmentation are presented in the table 6.1 for a sample training word.

6.2.2 Density tuning

The tuning of the output densities of the CDHMM states is applied to eliminate the differences between the most probable path and the correct path. For the time instants where the recognized path differs from the correct path the output density function of the correct state is altered to give a larger probability for the current feature vector. The output density function of the incorrect state can be respectively altered to the opposite direction.

There are naturally several different ways to modify the Gaussian mixture output densities of the CDHMM, for example, moving the mean vectors, adjusting the covariance matrices, balancing the mixture weights or applying some or all these three ways simultaneously. There are also many alternative objectives by which the modifications can be formulated, for example, the conditional ML representation of the sample or a gradient of an approximative cost function (Komori & Katagiri, 1992).

As the leading idea in developing this corrective tuning method was to avoid any complicated mathematical operations like complex gradient calculations the simplest reasonable tuning operation was chosen. The output density function is modified by finding the closest Gaussian mean vector in the mixture density function and moving it towards (or respectively away from) the current feature vector according to the LVQ learning laws (see section 4.2.2).

	Word	Labels on the best path
Incorrect	KEMU	KKKKEEEEEEEEEMMMMMMMMMUUUUUU U UUU
Correct	KEMI	KKKKEEEEEEEEEMMMMMMMMMIIIIII I III

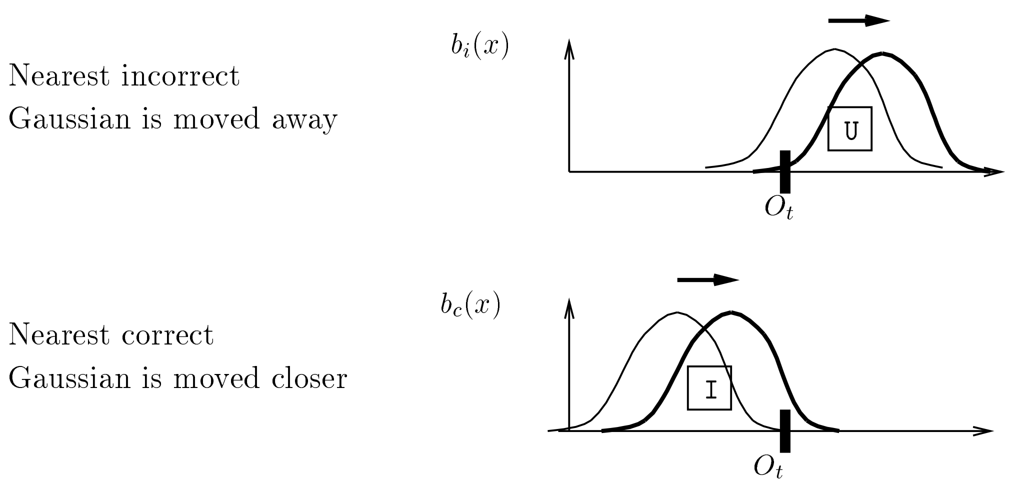


Figure 6.1: *Corrective tuning by LVQ. The incorrect but most probable HMM path and the correct path through a training word are compared and for different labels the output densities of corresponding states are modified by moving nearest Gaussian mean vectors to the observed feature, respectively.*

An attempt to illustrate the method briefly is presented in figure 6.1.

6.3 Connections to GPD and ODT

Although the tuning algorithm was heuristically motivated its close connection to an approximation of the Generalized probabilistic descent (GPD) formalism can be shown. This convinces the intuition about the asymptotic convergence to the optimal discrimination ability if sufficient training data is available.

Since the adjustment rule for GPD (5.39) was defined for positive values of parameters ζ, ξ and μ it can be considerably simplified by an approximation of the situation where $\zeta, \xi, \mu \rightarrow \infty$. From the approximation it follows that the various distance measures (5.32) and (5.33) are substituted by a minimum operator.

The approximative misclassification measure depends only on two classes; the correct class C_k and the most probable incorrect class C_i

$$d_k(\boldsymbol{x}) \approx g_k(\boldsymbol{x}; \Lambda) - g_i(\boldsymbol{x}; \Lambda) \ . \tag{6.1}$$

The class distance is computed using only the closest reference of the class

$$g_m(\boldsymbol{x}; \Lambda) \approx D(\boldsymbol{x}, \boldsymbol{r}_m^1) \ . \tag{6.2}$$

Also the reference distance is approximately the path distance of the best path selected by the Viterbi search.

$$D(\boldsymbol{x}, \boldsymbol{r}_m^1) \approx D_1(\boldsymbol{x}, \boldsymbol{r}_m^1) \ . \tag{6.3}$$

The adjustment rule (5.39) is then reduced to

$$\begin{aligned}
\boldsymbol{r}_{m,t,s}^b(n+1) &= \boldsymbol{r}_{m,t,s}^b(n) - 2\epsilon_n v_k w_{m,t}^b(\boldsymbol{r}_{m,t,s}^b(n) - \boldsymbol{x}_{1_t^{m,b},s}) \text{ ,for } m = k \text{ and } b = 1 \ , \\
\boldsymbol{r}_{m,t,s}^b(n+1) &= \boldsymbol{r}_{m,t,s}^b(n) + 2\epsilon_n v_k w_{m,t}^b(\boldsymbol{r}_{m,t,s}^b(n) - \boldsymbol{x}_{1_t^{m,b},s}) \text{ ,for } m = i \text{ and } b = 1 \ , \\
\boldsymbol{r}_{m,t,s}^b(n+1) &= \boldsymbol{r}_{m,t,s}^b(n) \ , \qquad \text{otherwise} \ ,
\end{aligned} \tag{6.4}$$

where $\boldsymbol{x}_{\theta_t^{m,1}|_{\theta=1}}$ is abbreviated by $\boldsymbol{x}_{1_t^{m,1}}$ (the frame of $\boldsymbol{x}$ corresponding to $\boldsymbol{r}_{m,t}^1$, the class C_m's best reference pattern, along the best path).

A closer examination of the correction term in 6.4, i. e.

$$2\epsilon_n v_k w_{m,t}^1(\boldsymbol{r}_{m,t,s}^1(n) - \boldsymbol{x}_{1_t^{m,1},s}) \tag{6.5}$$

reveals the resemblance to the LVQ2 algorithm (equation 4.3). In LVQ2 the corresponding correction term using the same notation as in equation 6.4 is

$$\alpha_n(\boldsymbol{r}_{m,t,s}^1(n) - \boldsymbol{x}_{1_t^{m,1},s}) \ , \tag{6.6}$$

if the sample falls in the window defined in equation 4.5 (zero otherwise).

The learning rate ϵ_n, which is defined in order to meet the convergence conditions of the stochastic approximation, is essentially the same as for α_n in LVQ.

The derivative of the cost function v_k was introduced in 5.39 to control the amount of modification depending on the degree of misclassification. The v_k is actually a symmetric function of $d_k(\boldsymbol{x})$ having maximum value at the class boundary and decreasing monotonically when $d_k(\boldsymbol{x})$ goes away from 0. For minimizing misclassifications on training data it is essential to apply the *learning-when-incorrect* scheme (Komori & Katagiri, 1992) so that v_k can be set to zero for negative $d_k(\boldsymbol{x})$ values. For positive $d_k(\boldsymbol{x})$ values v_k can be approximated by a step function being one inside a window near the class boundary and zero outside. This roughly same as to generalized the idea of LVQ2 from distances between reference vectors to distances or probability differences of vector sequences.

The weights $w^1_{m,t}$, which were used in equation 5.35 to give a different weighting for each frame t of the closest reference of class C_m to $\boldsymbol{x}$, can be neglected in the simplest case when considering all the frames to be equally significant.

Finally an algorithm, resembling the idea of LVQ2, to update the codebook vectors along the best path by Viterbi search emerges from a heavily simplified version of GPD. For every frame t of the closest reference of class C_k to $\boldsymbol{x}$ and that of the most probable incorrect class C_i the following update is done, if the training sample $\boldsymbol{x}$ is close enough to the class boundary between C_k and C_i so that the misclassification measure fits in the pre-specified window.

$$\begin{aligned} \boldsymbol{r}^1_{k,t,s}(n+1) &= \boldsymbol{r}^1_{k,t,s}(n) - 2\epsilon_n(\boldsymbol{r}^1_{k,t,s}(n) - \boldsymbol{x}_{1^{k,1}_t,s}) \\ \boldsymbol{r}^1_{i,t,s}(n+1) &= \boldsymbol{r}^1_{i,t,s}(n) + 2\epsilon_n(\boldsymbol{r}^i_{i,t,s}(n) - \boldsymbol{x}_{1^{i,1}_t,s}) \end{aligned} \tag{6.7}$$

From the different variations of corrective tuning described in section 6.4 this resembles most the version v4.

The difference of the corrective tuning and the simplified GPD is that in GPD the update above is applied for every vector in the misclassified training sample. In the corrective tuning presented in this chapter the LVQ is used to control which vectors are actually updated as explained more detailed in the section 6.4. This a little more careful selection of the Gaussian mean vectors to be updated does not, however, prevent the use of the convergence results for GPD derived in (Katagiri et al., 1991).

Compared to the heuristically justified corrective training method for CDHMMs (ODT) presented in section 5.3.2 some quite fundamental differences exist although ODT might look similar at the first glance as the corrective tuning.

The basis of the learning rate in ODT (equations 5.30 and 5.31) is a prespecified constant β preventing the stochastic convergence for controversary samples caused by gradually decreasing learning steps. The amount of update also depends on the "degree" of misclassification defined as a probability difference of a correct model λ_c and an incorrect one λ_i, i. e. $D(c,i) = \Pr(\boldsymbol{O}|\lambda_c) - \Pr(\boldsymbol{O}|\lambda_i)$). Unlike the v_k of GPD and the *learning-when-incorrect* of LVQ the dependence is defined to be a constant for $D(c,i) \leq 0$ and to decrease linearly to zero as $D(c,i)$ approaches to a positive constant δ (equation 5.31). This involves learning with both incorrectly and correctly classified samples, but by using different learning steps.

6.4 Implementation

The training words are circulated and recognized one at a time just as they were unfamiliar test words, but after each recognition the obtained phonetic transcription is compared to the correct one. If the recognition fails the corrective tuning is applied to adjust the HMMs slightly in the direction of correct recognition.

The computation of the most probable path and the correct path is a straightforward application of the Viterbi algorithms, where the existing segmentation information can be integrated as explained in section 6.2.1.

The two paths are inspected for each time instant and the adjustments necessary according to one of the five following different variations (v1–v5) are made.

Due the same reasons as discussed in implementation of density initialization by LVQ (see section 4.3.2), the update decisions by the LVQ learning law for corrective tuning are approximated by comparing only HMM labels instead of individual states.

- v1: The nearest Gaussian mean vector of all states and all HMMs compared to the current feature vector is sought and updated according to the LVQ1 learning law (equations 4.1 and 4.2).
- v2: As v1, but the two nearest mean vectors are sought and updated according to the LVQ2 learning law (equation 4.3), if their distances to the current feature vector satisfy the windowing condition (equation 4.5).
- v3: As v2, but the LVQ3 learning law (equations 4.4 and 4.6) is applied.
- v4: The nearest Gaussian mean vector of the current state in the most probable path and in the correct path are sought, respectively. The LVQ2 learning law (equation 4.3) and the windowing condition (equation 4.5) are then applied to these two vectors assuming that the former mean vector is m_c and the latter m'_c. For SCHMMs the situation is slightly different, because it is not trivial to determine the state that the nearest Gaussian represents. Instead the nearest Gaussian mean vector with the same label as the current state in the most probable path and in the correct path are sought, respectively. The labeling used for the Gaussians was already described in section 4.3.2. Another way to determine the required mean vectors is to find the one with the largest weight factor (c_{im} in equation 2.15) for the current state of both the paths. Due to poor initial results this is considered only as a back-up strategy in the rare case that some label would not be represented at all in the labeling by the majority voting.
- v5: As v4, but no manual labeling is used to aid the determination of correct path (see section 6.2.1). The correct path is computed by using the constrained Viterbi search using only the information about the correct phonetic transcription of the training word.

As already noted above for v4, the determination of the state that the nearest Gaussian represents, is not so trivial for SCHMMs as it is for CDHMMs, because all the Gaussians are tied between all the states. Due to the limited number items in the codebook of Gaussians it is not desired to divide them between the states, because some states might get only a few or none representatives. So instead of defining the states of HMMs as the classes used by the LVQ learning laws the labels are used to define the classes. The labels of the mean vectors can be more reliably determined by majority voting (like explained in section 4.3.2), because the number of different labels is much smaller than that of different states and they are perhaps naturally more clearly separable.

The division of the Gaussians according to the HMM labels instead of HMM states is not so significant change in the learning scheme as it might sound, because the objective is still to enhance the output pdfs of the HMM states to better discriminate with different phoneme models. It is merely a question of more efficient parameter tying as discussed in section 2.5.3.

In fact, it would be intriguing to try an intermediate arrangement between SCHMMs and CDHMMs by tying only the Gaussian densities inside the CDHMMs. This would actually suit very well to the corrective tuning performed according to CDHMM labels rather than individual states.

Chapter 7

PHONEME RECOGNITION EXPERIMENTS

In addition to being the stimulating force behind the modeling and algorithmic studies of this project the speech recognition experiments also provide an application challenging enough to reliably test and rank the proposed training method combinations. The variety of different important practical applications ranging from hands-free dictation to on-line database retrieval over telephone lines makes even small achieved improvements in recognition efficacy and accuracy significant, because when brought to the product level misunderstandings and waiting times are expensive in the merciless competition between commercial goods.

While the speaker, task and vocabulary independent continuous speech recognition remains still an unachieved goal, the progress is made on several fronts of restricted requirements. For example, the system used in this project can be categorized as a speaker-dependent dictation tool for unlimited supply of Finnish words to work in office conditions with low background noise. The aim is to be able to dictate messages continuously with careful pronunciation, distinct spacing and uniform stress.

Phoneme recognition is a difficult task since the spectral distributions are considerably overlapping, because the phonemic context around current phonemes affects to the observable short-time auditive features. However, the recognition of phonemes is an important stage in the recognition of words, since in languages like Finnish the number of different phonemes is small but the number of different words is very large due to their numerous inflected forms and connected combinations.

By approaching phoneme recognition as a vector space pattern recognition problem the major decisions to be made are the choice of features, classes and classifiers. Initially it is feasible to choose one class for each phoneme, but it is also possible to split some strongly variating phoneme into two distinct classes or join some rare and similar phonemes into a single class. The features are chosen to separate the classes as accurately as possible using compact representation and minimal computation effort. The cepstral coefficients extracted from a 5 – 30 ms time period have been quite popular lately as short-time features in speech recognition (Davis & Mermelstein, 1980; Furui, 1989).

The classification of the acoustic features into phoneme classes can be performed by many classical pattern recognition algorithms. However, the length of phonemes varies and the features computed near the border areas between phonemes can be quite different from those from the centers of the phonemes. In addition to these facts also the coarticulation effects ensure that the short-time features by themselves do not provide very accurate results and it is thus necessary to take the temporal structures of the speech signal into account in the classification.

Speech is a nonstationary signal where the articulatory apparatus of the speaker modulates the air pressure and flow to produce an audible sequence of sounds. Thus the modeling involves both the short, almost stationary phases of individual sounds and the long-time large variations due to the changing articulatory configuration in development of sound sequences. Another important property is the correlation between successive short-time features due to the continuous nature of the changes in the articulatory configuration and the feature computation mechanism. An example of the integration of the short-time stationarity and the temporal relationships is the hidden Markov modeling as described in chapter 2.

```
K>><><<>PP<<MAAAAAAAAAAAAAA[AAEEEIIJJJJJIJIIIII><>>>>>>TKKKKOAAAAAAAAH<
23323212100133222233132333001132111231313111010303033010333310001113120
>>>>>>>>>AAAAAAAAAAAAAAAAAAAIIIIIIIIIIIIIIIIIIKKKKKKKKKKKAAAAAAAAAAAA<<
00001222300001222222222222223001111111112222223011222222223000000011123000

<<<<<<<<<<<>>K<<<<<K<<K
00010111111133121002222
<<<<<<<<<<<<<<<<<<<<<<<
00011111111111111222223
```

Table 7.1: *The first row contains the labels of the states having output density function with best match to the each individual feature vector. The numbers on the second row reveal the best matching states numbered from 0 to 3 in the corresponding 4-state CDHMM. The third and fourth rows give the same information of the best matching state sequence obeying the Markov properties and found by the Viterbi search. The silence labels preceding and following the word are indicated by* $>$ *and* $<$*, respectively.*

To illustrate the importance of the temporal modeling two different HMM paths through a test word are shown in table 7.1. Each label-number pair (numbers are below the labels) corresponds to the classification of a short-time acoustical feature vector (10 ms apart). The upper path is constructed by the locally best matching HMM states without context relationships. The lower path is obtained by using also the estimated knowledge about state transition probabilities to the state output densities by dynamical programming. The segmentation of the spoken Finnish word (AIKA) is remarkably easier by using the lower path.

7.1 Recognition system

The speech recognition system of the Laboratory of Information and Computer Science at Helsinki University of Technology (Torkkola et al., 1991a) was used in the experiments. The system is used like a phonetic typewriter to transcribe spoken Finnish words into phoneme sequences. The main exception to the ordinary system is the application of continuous density HMMs instead of the discrete HMMs like in (Torkkola et al., 1991a; Torkkola et al., 1991b; Mäntysalo et al., 1992; Utela et al., 1992). The acoustical features used as observations of speech signal are 20-component cepstral vectors computed every 10 ms from a 20 ms window concatenated with the energy of the signal.

The signal window is obtained by emphasizing the 256 successive speech samples (12.8 kHz sampling frequency) by Hamming window ($N = 256$):

$$w_M(n) = 0.54 - 0.46 \cos \frac{2\pi n}{N-1} \ , \quad n = 0, 1, \cdots, N-1 \tag{7.1}$$

The next step is a 256-point FFT used to construct 128-point logarithmic power spectrum illustrated in figure 7.1 (an example).

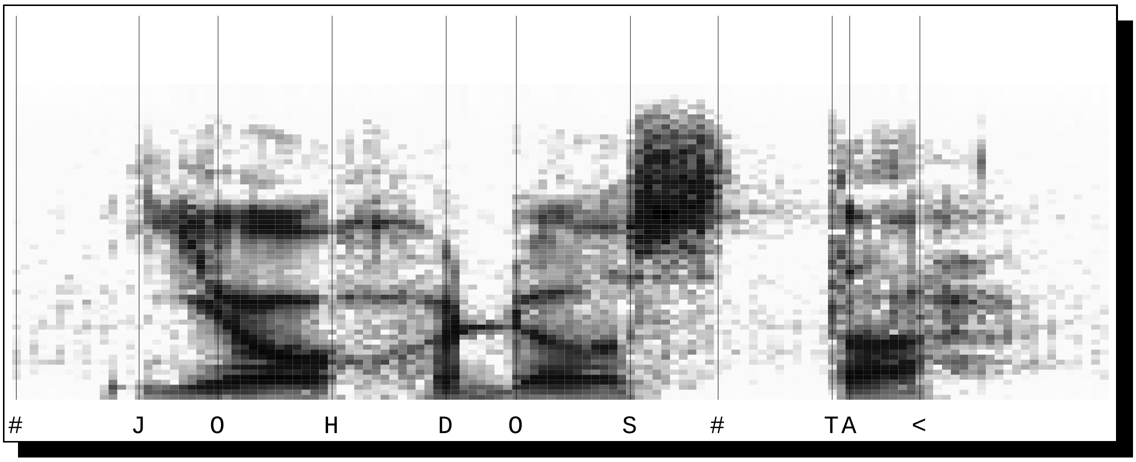


Figure 7.1: *Power spectra from speech signal 10 ms apart. Time increases to right and frequency upward. The darkness of the pixels is related to the value of the corresponding spectral factors. The vertical lines show the starting points of the phonemes. # means the silence before the word or a plosive and < the end of word.*

The power spectrum is filtered using 22 partially overlapping triangular Mel-scaled filters X_k. The Mel-scale is based on the physiological and psychoacoustical properties of the human hearing mechanism and divides the frequency band of the speech into 10 constant (50 – 1000 Hz) and 12 exponentially growing (1 – 5 kHz) regions.

The cepstral coefficients are computed by a discrete cosine transform of the filtered power spectrum (Davis & Mermelstein, 1980)

$$MFCC_i = \sum_{k=1}^{N} X_k \cos\left[i\left(k - \frac{1}{2}\right)\frac{\pi}{N}\right], i = 1, 2, \ldots, M \ , \tag{7.2}$$

where N is the number of filters (here 22) and M the dimension of the cepstrum (here 20). For the optimal classification ability the cepstrum is weighted using a window of raised sinusoid (Tohkura, 1987). The short-time cepstra corresponding to figure 7.1 are illustrated in figure 7.2.

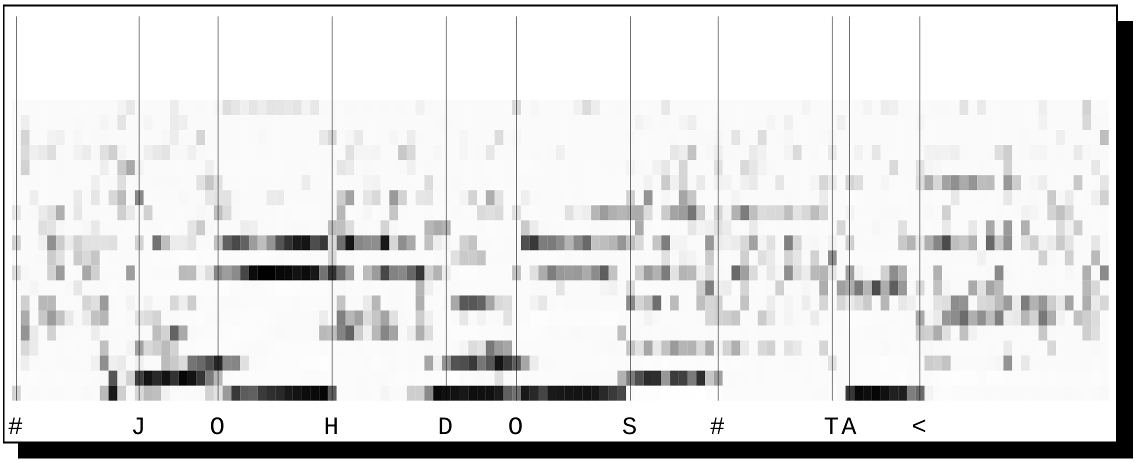


Figure 7.2: *Short-time cepstra from speech signal 10 ms apart. Time increases to right and quefrency (same as frequency for a spectrum) upward. The darkness of the pixels is related to the value of the corresponding cepstral factors.*

In the original speech recognition system with discrete HMMs capable for an online demonstration a codebook trained by LVQ (Kohonen, 1990b) is first used to vector quantize the feature vectors. The quantization procedure produces instantaneous phoneme labels called the *quasiphonemes* corresponding the feature vectors. The discrete HMMs provide probabilities for the observation of each quasiphoneme for each HMM state and the Viterbi search can be performed to find out the most probable state sequence. An example of a quasiphoneme string is this quantization result of the spoken word JONKA:

```
VVJJJIYYYOOOOOOOOVVNNNNNNNNNNNNNNNMNMVKKKKKROOOAAAAAAA
```

When using the continuous density HMMs the LVQ codebook is integrated with the observation density models of the states so that the probabilities to observe each feature vector can be computed directly with the expense of some more computational effort.

The transformation of the most likely HMM path to the labels of the real word is a relatively simple procedure (in Finnish). Because each HMM corresponds to one phoneme the sequence of visited HMMs is directly the sequence of phonemes in the real word which is quite near to the way the words are written in Finnish. Since the long phonemes are represented by the same HMM as the short ones but written differently it is necessary to maintain a table of the time limits between long and short version separately for each phoneme in order to be able to write them correctly. It is possible to also maintain some set of conversion rules between the spoken and written words as well as to correct some typical recognition mistakes for the speakers. For this purpose there is a method called Dynamically Expanding Context (DEC) (Kohonen, 1986a) which is integrated to speech recognition system and trained for each speaker separately to correct the most typical mistakes. Nevertheless, the DEC rules are not used in the following experiments to better discriminate the performance of differently trained HMMs.

7.2 Experiments

7.2.1 Material

The experiments are carried out using a set of 311 Finnish words and three native Finnish speakers. From each speaker four repetitions of the word set are available and the speaker dependent HMMs are trained with three sets per speaker leaving one set for testing.

The recognition performance was determined by calculating the sum of changed, missing and extra phonemes in decoded phonetic transcriptions divided by the correct number of phonemes in the test material. To get the average recognition error rates presented in tables (7.2 – 7.10) the four different training and test set combinations are applied for each speaker yielding a total number of 12 tests.

7.2.2 Matched-Pairs statistical test

The statistical significance of the difference in error rates between different algorithms is verified by the Matched-Pairs test (Gillick & Cox, 1989). The method is based on the division of the speech into segments in such a way that the errors in one segment are statistically independent of the errors made in any other segment. As the words are uttered one at a time in this application the natural choice for such a segment is a word.

The two recognition algorithms which are compared by a recognition test are called A_1 and A_2. N_1^i and N_2^i are the number of errors made on the i'th segment by A_1 and A_2, respectively. The difference of the number of errors per a segment is $Z^i = N_1^i - N_2^i$, $i = 1, 2, \cdots, n$ and the unknown average difference of the number of errors is μ_Z. The distribution of μ_Z is defined as:

$$W = \frac{\hat{\mu}_Z}{\hat{\sigma}_\mu} \ , \tag{7.3}$$

where

$$\hat{\mu}_Z = \frac{1}{n}\sum_{i=1}^{n} Z_i \tag{7.4}$$

$$\hat{\sigma}_Z^2 = \frac{1}{n-1}\sum_{i=1}^{n}(Z_i - \hat{\mu}_Z)^2 \tag{7.5}$$

$$\hat{\sigma}_\mu = \frac{\hat{\sigma}_Z}{\sqrt{n}} \tag{7.6}$$

When n is large, W has approximately a Normal distribution with unit variance. The null hypothesis is $\mathbf{H_0} : \mu_Z = 0$, so that under $\mathbf{H_0}$, the distribution W has zero mean, i. e. there is no statistical difference between A_1 and A_2. The $\mathbf{H_0}$ can be tested by computing

$P = 2\Pr(X \geq |w|)$ where X is random variable with distribution $N(0,1)$ and w is the realization of W. $\mathbf{H_0}$ can be rejected by a significance level α if $P < \alpha$.

Since the number of independently pronounced segments (words) is large enough (3732) in the available test material the difference in the number of errors in a segment made by the compared algorithms can be approximated by a normal distribution and the Matched-Pairs test is applicable.

7.2.3 Model configuration

Design objectives

The choice of the HMM configuration is important having significant effects on the training and recognition behavior of the models. However, there are very few exhaustive comparison results covering all applicable configurations.

Since there are a lot of different parameters to be chosen of and compromises to be made between the model flexibility, the number of parameters, the type of output densities, the computational load, the representation and the discrimination ability and so on, it is usually impossible to involve complete testing. The normal procedure in the literature seems to be to select configurations that have previously yielded good results and perform some tests to optimize only a couple of the properties mentioned above. That will also be the HMM configuring strategy in this work. The main objective is here to optimize the recognition performance while keeping the number of parameters and the computational load inside some reasonable limits.

One HMM per phoneme

The basic principle is the construction of one HMM for each phoneme. To be able to keep the training procedures as simple as possible the HMMs are restricted so that they all have the form and the same number of parameters. It is evident that this is by no means an optimal choice in the terms of representation or efficacy and, for that reason, it is important that the selected HMM configuration has enough flexibility to adequately represent all different types of phonemes. By the training reasons the observed silences occurring immediately before and after the word is uttered are modeled by similar models as the phonemes.

Chain construction

A common and simple choice for HMM type (e.g. (Bakis, 1976; Rabiner et al., 1983; Juang & Rabiner, 1991)) that is also chosen here is the uni-directional chain of three to five states with possible loops inside every state except the last one, which is considered to be a linking state to the first state of the next phoneme. The last state in the chain can be also treated as an empty transition by joining it with the previous state in such a way that the transition to the next phoneme occurs simultaneously as that state is left a separately observable output.

Number of states

The final choice between the number of states in the HMMs can be made by simply testing the recognition performances. As the number of different states is increased it is assumed that each phoneme can be represented more detailed. This is, of course, only possible when sufficient training data is available. It should be noted again that the presentation ability is not so crucial as the discrimination in testing of the recognition performance. Another practical point of view is that the computation time required for the HMM training and testing may also become intolerably long as the model complexity is increased.

Type of output density

The most important thing to be decided in the configuration of the HMMs is the form of the output density functions of the measurable acoustic observations in each state. This decision determines in a great extent the training methods and method combinations as well as the model structure. The two main trends are the discrete HMMs (section 2.4) and the CDHMMs (section 2.3). Because the properties of different HMMs were discussed quite detailed in chapter 2 they are not repeated here. The chapter 2 is also referenced for the properties of SCHMMs (section 2.5) constituting a kind of intermediate between the two main trends. As the purpose of this work was to study the combinations of neural network methods and the training of continuous (semi-continuous) HMMs the discrete density HMMs are not used in these experiements. For experiments of DHMMs applications in the same speech recognition system there are other excellent reports, e. g. (Utela, 1992; Mäntysalo, 1992; Torkkola, 1991).

Selection of the mixture densities

For a continuous pdf the normal choice is a Gaussian pdf or mixtures of Gaussian densities. Mixtures of other continuous pdf could be possible as well; the only theoretical limitation for the convergence of the ML training being the elliptical symmetry as shown in (Juang, 1985). Actually, because the separability of different phonemes is the main objective, the continuous pdfs can be replaced by suitable discriminant functions. In practice the distributions of observations of different phonemes are, however, quite complicated and overlapping. For the mixture Gaussian densities the choice to be made is the amount of mixtures to be used for the output density function of each state. This is again a trade of between the complexity and the efficacy taking account the limitations of training data. Some experimental results concerning different state and mixture combinations are given for SCHMMs in table 7.2 and for CDHMMs in table 7.3. The figure 7.3 illustrates the training behavior of different state and mixture combinations for CDHMMs.

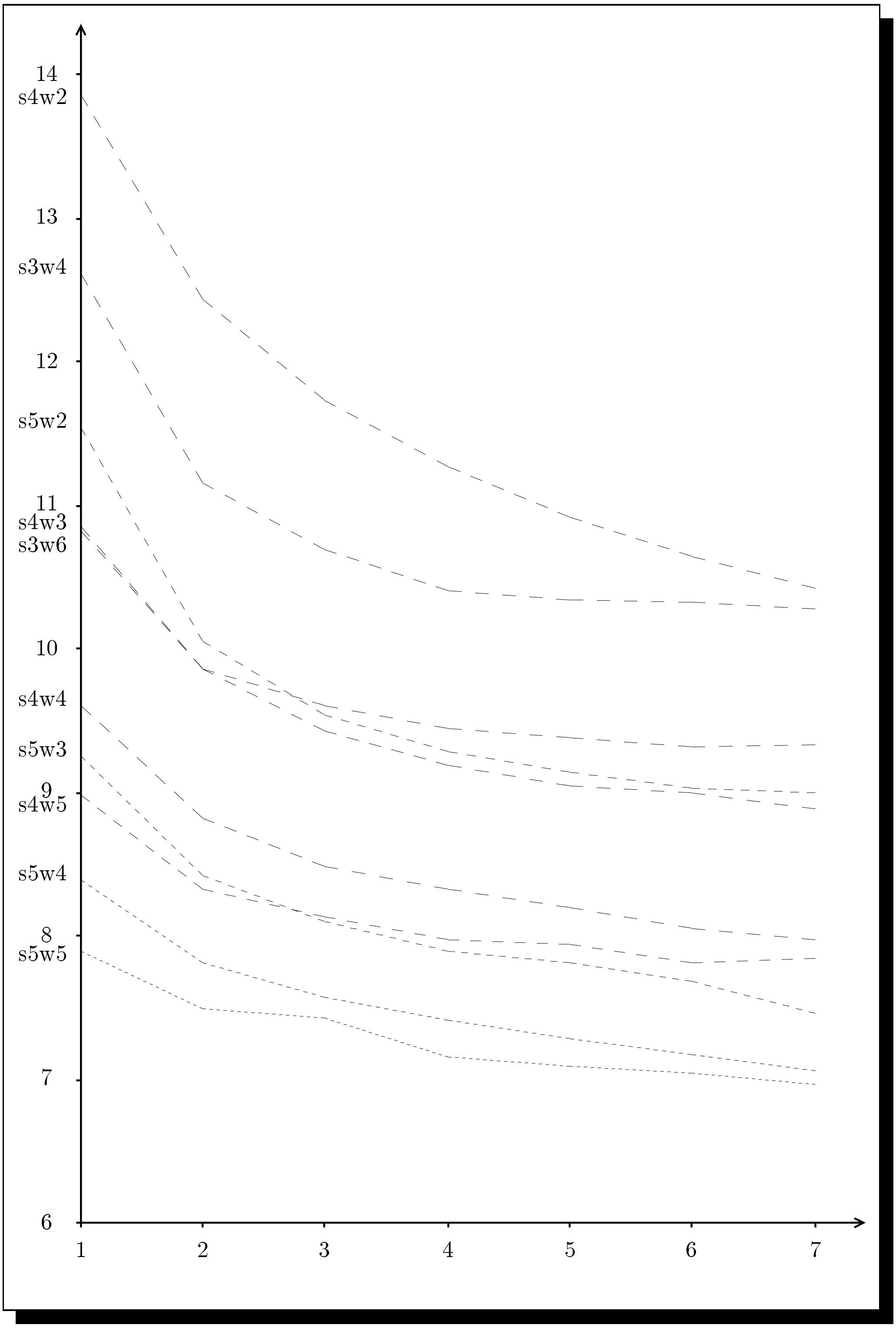


Figure 7.3: *Development of the average error rate during Baum-Welch iterations for different CDHMM configurations. The models are initialized by SOM (5 times the data) and further trained by Baum-Welch (7 iterations). Number of iterations is shown in horizontal axis and percentage of recognition errors for independent test data in vertical axis. yThe labels of the different curves refer to the model configurations so that, for example, in 's5w4' there are 5 states per phoneme and 4x4 Gaussian mixtures per state.*

states	mixtures		
	192	494	999
3	12.0	10.7	
4	10.0	8.6	7.9
5	9.5	8.3	

Table 7.2: *Average recognition error rates for different SCHMM configurations. The number of states per HMM is varied between 3–5 and the number of tied mixtures between 192, 494 and 999. The training of the models is performed as described later as the default training for SCHMMs except no corrective tuning. The dimensions of the SOMs corresponding to different mixture sizes are 16x12, 27x19 and 37x27.*

states	mixtures				
	4	9	16	25	36
3	15.0		10.3		9.3
4	10.4	8.9	8.0	7.8	
5	9.0	7.5	7.1	7.0	

Table 7.3: *Average recognition error rates for different CDHMM configurations. The models are initialized by SOM (5 times the data) and further trained by Baum-Welch (7 iterations). The dimensions of the SOMs corresponding to different mixture sizes are 2x2, 3x3, 4x4, 5x5 and 6x6.*

Reduction of the covariances

The modeling of complex multivariate distributions with high-dimensional Gaussian densities has a problem with the size of covariance matrices growing into square proportional to the dimensionality. A common trick to avoid this inconvenient behavior is to use only diagonal covariance matrices. The accuracy lost by using this approximation can be easily gained back by investing the “spared parameters” for the increase of the number of mixtures densities. An even more radical way to transfer parameters from covariance matrices to mean vectors is to tie all covariances, i. e. use only one global covariance matrix (see section 2.5). This approximation is called the Grand variance theorem by (Paul, 1989).

Default HMMs

- One HMM with 5 successive states per phoneme
- Both CDHMMs and SCHMMs are used (separately tested)
- Gaussian mixture pdfs
- 24 mixtures per state for CDHMMs
- 192 tied mixtures for SCHMMs

- tied diagonal covariance matrices

7.2.4 Training combinations

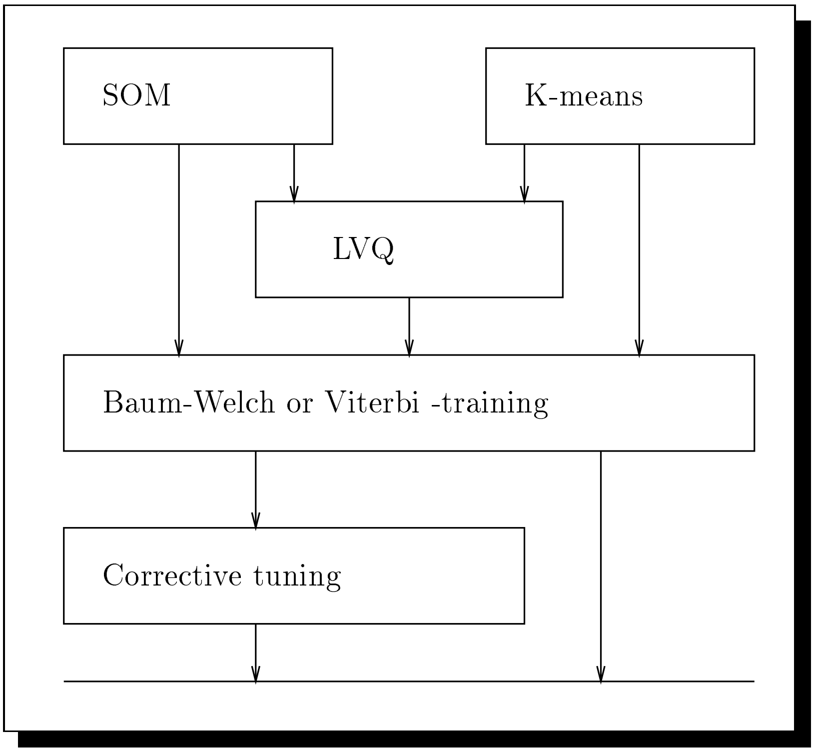


Figure 7.4: *The training combinations experimented with CDHMMs and SCHMMs.*

The training combinations that are used in experiments are visualized by the figure 7.4. The three main phases in the training of the HMMs are initialization, ML training and corrective tuning.

The main concern in the initialization is the selection of a suitable set of Gaussian mean vectors for the mixture density functions of the states. The alternate methods used are SOM and Kmeans algorithms followed by LVQ. These methods are themselves initialized by random vectors or sample vectors arbitrarily selected from the training data. The initial vectors can also be selected using the K-Nearest Neighbor rule.

The alternate algorithms for the ML training are Baum-Welch reestimation and Viterbi training. It is also possible to use both algorithms one after another.

For corrective tuning an algorithm based on the LVQ2 learning law is proposed.

The following default training combinations for CDHMMs and SCHMMs as well as the previously defined default configurations will be later used in the experiments, if not described more detailed.

Default training for CDHMMs

- SOM initialization: first 1 cycle with all training samples decreasing the neighborhood radius down from 5 (in a 6x4 array) and the learning rate down from 0.2 and then 5 cycles with radius down from 2 and learning rate down from 0.02
- Reestimation: 10 iterations of Viterbi training
- Tuning: 5 cycles with all training samples decreasing the learning rate down from 0.02 and using learning law LVQ2 with window 0.3 applied when the most probable path differs from the correct path

Default training for SCHMMs

- SOM initialization: first 1 cycle with all training samples decreasing the neighborhood radius down from 10 (in a 16x12 array) and the learning rate down from 0.2 and then 5 cycles with radius down from 3 and learning rate down from 0.02
- LVQ initialization: 2 cycles with all training samples decreasing the learning rate down from 0.02 and using learning law LVQ3 with window 0.3 and $\epsilon = 0.1$
- Reestimation: 10 iterations of Baum-Welch
- Tuning: 1 cycle with all training samples decreasing the learning rate down from 0.02 and using learning law LVQ2 with window 0.3 applied when the most probable path differs from the correct path

7.2.5 Initialization tests

The suitability of different initialization procedures for the HMM output densities can be most reliably tested by running the whole parameter training by changing only the initialization method and then comparing the results. Any measurable direct comparison of the initial models will probably fail to rank the initialization methods in the respect of the final recognition accuracy, because it seems impossible to formulate the corresponding criteria without comparing the final models.

In table 7.4 the average recognition error rates are presented for various initialization procedures for SCHMMs. The results of the procedures are compared to default initialization made by SOMs. The Matched-Pairs test is used to determine the risk level for each procedure for being better than the default initialization. Using the risk level 0.05 often considered as the limit of the significance the LVQ2 (or 2.1) and LVQ3 when applied after SOM lead to better models than the default. Also the combination KNN+LVQ3 provide a significant improvement to the default.

The figures 7.5 and 7.6 illustrate improvements of the CDHMMs with different initializations while the number of Baum-Welch or Viterbi training iterations, respectively, is increased. The recognition error rates after each training iteration for each initialization-training combination are tested and joined by different line types. These pictures are presented to show the change of convergence speed in the laborious ML estimations gained

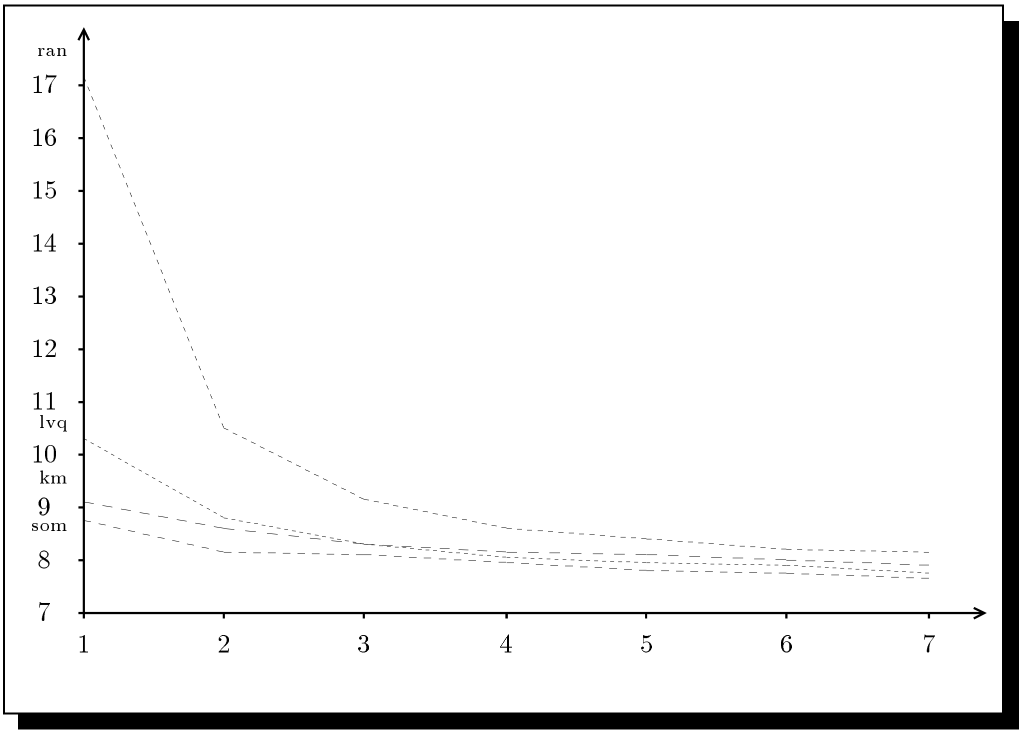


Figure 7.5: *Development of error rate for 4-state CDHMMs during Baum-Welch iterations with random initial values "ran" compared to initialization done by SOM, K-means and LVQ. Number of iterations is shown in horizontal axis and percentage of recognition errors for independent test data in vertical axis.*

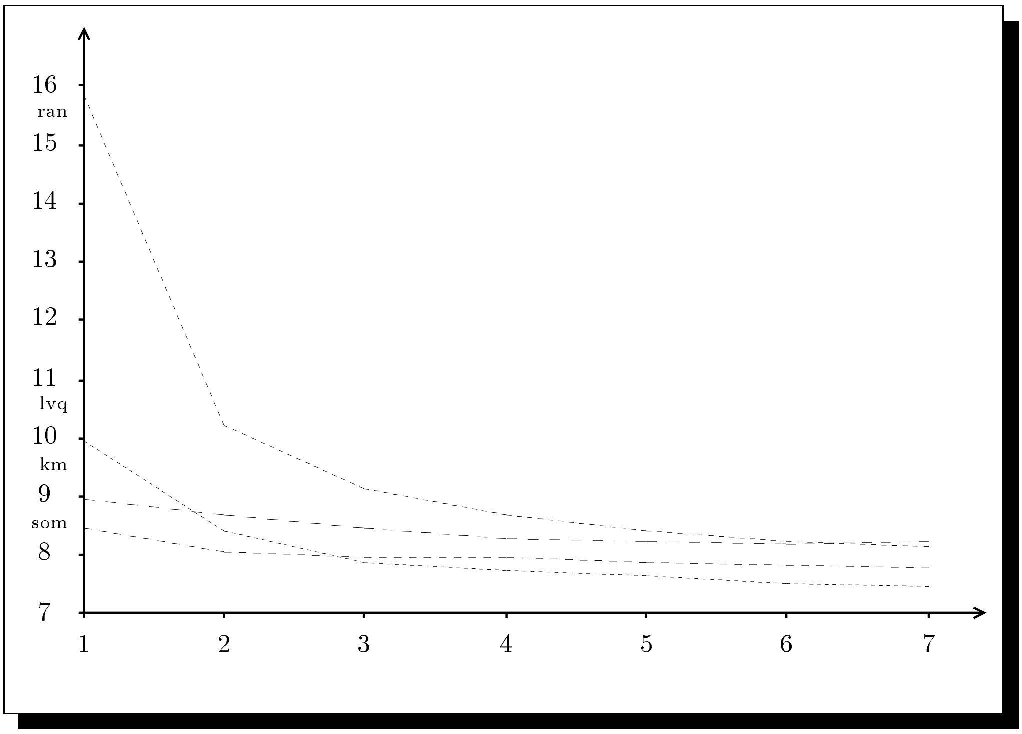


Figure 7.6: *Development of error rate for 4-state CDHMMs during Viterbi training with random initial values compared to initialization done by SOM, K-means and LVQ.*

version	window	error%	risk
SOM		10.5	
SOM+LVQ1		10.3	0.22
SOM+LVQ2	0.3	10.0	0.0001
SOM+LVQ2	0.5	10.0	0.0001
SOM+LVQ2.1	0.3	10.0	0.0136
SOM+LVQ3	0.3	10.0	0.0001
SOM+LVQ3	0.5	10.2	0.0096
KM		10.5	-0.89
KM+LVQ3	0.3	10.8	-0.17
KNN+LVQ3	0.3	10.1	0.0052

Table 7.4: *Recognition error rates for variations of LVQ used in the initialization of the Gaussian mean vectors for 4-state SCHMMs. No tuning is performed after the Baum-Welch reestimation. KM is the Kmeans algorithm and KNN is for K-Nearest Neighbors. The risk level by the Matched-Pairs test on each row determines the risk by which that initialization method is statistically better than the method on the first row. The minus sign on the risk means the method is worse than the default by that risk level.*

by initializing the models carefully. For example, the initialization by SOMs leads to a low recognition error rate after only two ML iterations which seems to take at least three times more (or much more because the improvements are very slow after the sixth iteration) training with random initial values.

The importance of the convergence speed grows substantially as the number of the models is increased, because it is then more difficult to get near to the optimal performance and also more training data is then required.

7.2.6 Recognition tests

The output probability calculations of the mixture density HMMs, which are made for each observed short-time feature vector and for each state, consume a significant part of the total computation time in the recognition of the spoken words.

One way to reduce the computational load of the probability calculations is to approximate the output probabilities by using only the nearest Gaussians (using the distances between mean vectors and the current feature vector) instead of the whole set (Bellegarda & Nahamoo, 1990). This is feasible because finding the set of n nearest vectors is a considerably lighter task than calculation of the full probabilities for the rest of the $N - n$ Gaussians if n is, for example, smaller than half of N where N is the total number of Gaussians used for the mixture. This approximation also tends to increase the discrimination between the states because only a couple of nearest Gaussians affects the output probability (Kurimo, 1994).

In table 7.5 the effect of the density approximation by using only n nearest Gaussians of

$N=35$		$N=24$		$N=15$		$N=9$		$N=4$	
n	e%	n	e%	n	e%	n	e%	n	e%
35	6.1	24	6.2	15	6.3	9	6.9	4	8.0
10	*5.9*								
7	*5.9*	7	*6.1*	5	*6.1*				
5	*5.8*	5	*5.9*	4	*6.0*				
3	**5.6**	3	*5.8*	3	*6.0*	3	*6.8*		
2	*5.7*	**2**	**5.8**	**2**	**5.9**	**2**	**6.6**	**2**	**7.9**
1	*5.8*	1	*5.8*	1	*6.0*	1	*6.6*	1	8.0

Table 7.5: *Recognition error percentages (e%) for various n and N values for CDHMMs. The $n=1$ approximation is also applied in the Viterbi training. The bold-faced percentages indicate the best n values for each mixture size. The emphasized percentages are significantly (by risk < 0.05) different from the original $n=N$.*

$N=35$		$N=24$		$N=15$		$N=9$		$N=4$	
n	e%	n	e%	n	e%	n	e%	n	e%
35	6.6	24	6.8	15	6.8	9	7.2	4	8.3
1	5.7	1	5.9	1	6.2	1	6.7	1	8.1

Table 7.6: *Recognition error percentages for $n=N$ and $n=1$ recognition experiments for CDHMMs when the normal Viterbi training $n=N$ is used.*

the total N has been tested for several mixture sizes to see, how much the recognition accuracy would drop by different levels of approximation. The bold-faced percentages indicate the best n values for each mixture size.

Contrary to the expectations the best results are obtained with low n values instead of $n=N$. This is assumed to be caused by the increased discrimination between the models since the Gaussians are well localized to represent only certain features. The usage of only a few Gaussians emphasizes perhaps more clearly the distinctive features since when all the Gaussians are involved the probability reflects more about how well the current feature vector represents the average features of that state.

The default type of CDHMMs as well as the default initialization procedure is kept unchanged, but in the Viterbi training the $n=1$ approximation is also applied. Comparison to the result with normal $n=N$ Viterbi training (table 7.6) reveals that the density approximation in the training phase also brings some improvements to the results in addition to lighter computation.

The density approximation is also tested for different mixture sizes for SCHMMs (in table 7.7). The increase of the recognition error rate is insignificant until after n falls below approximately $N/20$. This approximation level seems to be very suitable since only the probability computations of about 1/20 of the Gaussian densities are required and the recognition error keeps practically the same as for the full computation of $n=N$.

$N = 999$		$N = 494$		$N = 352$		$N = 192$	
n	e%	n	e%	n	e%	n	e%
999	**7.4**	494	8.2	352	8.6	192	9.6
200	7.4	**100**	**8.1**	**100**	**8.6**	80	9.6
100	7.5	50	8.2	50	8.6	40	9.5
50	7.6	25	8.3	25	8.7	**20**	**9.5**
25	*8.1*	15	*8.8*	18	*8.8*	15	9.5
15	*9.3*	10	*9.5*	10	*9.4*	10	9.5
10	*12.2*	5	*14.7*	5	*12.4*	5	*10.6*

Table 7.7: *Recognition error percentages (e%) for various n and N values for SCHMMs. The default training for SCHMMs is used here. The bold-faced percentages indicate the best n values for each mixture size. The emphasized percentages are significantly (by risk < 0.05) different from the original $n = N$.*

7.2.7 Corrective tuning tests

Version	CDHMM, $N = 24$		SCHMM, $N = 192$	
	$n = 24$	$n = 1$	$n = 192$	$n = 10$
	6.8	5.9	9.6	9.5
v1	6.8	5.9	9.7	9.7
v2	6.8	5.6	9.6	9.5
v3	6.8	5.9	10.8	10.9
v4	6.4	5.5	9.8	9.9
v5	6.3	5.6	9.2	9.5

Table 7.8: *Recognition error percentages for variations of corrective tuning. First row is errors before correction.*

The different versions of the corrective tuning method as described in chapter 6 were tested for the default configurations of CDHMMs and SCHMMs and the resulting average error rates are reported in table 7.8. Also the recognition results after the density approximation presented in the previous section are separately shown in the table. Comparing the error rates after tuning to those before tuning (the first result row of table 7.8) it is observed that the version v5 (see section 6.4 for the description) gives the smallest error rates for most of the models.

To check the significance of the decrease of the average recognition error rates obtained by the corrective tuning (version v5) the results before and after tuning for CDHMMs trained by different ML training algorithms are presented in table 7.9. The middle row is obtained by the default Viterbi training, the upmost row by 7 iterations of Baum-Welch reestimation and the lowest row by the approximative Viterbi training in the same way as in table 7.5, i. e. the recognition phase is done by approximating the mixture pdfs by only the nearest Gaussian.

The risk is the risk level by which the recognition results after tuning are better than

algorithm	error1%	error2%	risk
Baum-Welch	7.1	6.6	0.0001
Viterbi	6.8	6.2	0.0068
Viterbi.a	5.8	5.6	0.0050

Table 7.9: *Recognition error rates for different CDHMM training algorithms (error1%) and after the LVQ-based error correction (error2%). The risk level is from Matched-Pairs test and determines the risk by which the error corrective tuning decreases the error rate for that row. The training by Viterbi.a is performed using approximative pdfs in the same way as in the corrective tuning.*

before tuning on that row. Since the achieved risk levels extremely low ($<< 0.05$) it is clear that the improvements are significant.

algorithm	states	error1%	error2%	risk
Baum-Welch	4	9.8	9.4	0.0005
Baum-Welch	5	9.1	9.0	0.15
Baum-Welch*	5	8.2	7.9	0.0001
Viterbi	4	12.2	11.4	0.0001
Viterbi	5	12.5	11.9	0.0001

Table 7.10: *Recognition error rates for different SCHMM training algorithms (error1%) and after the LVQ-based error correction (error2%). The risk level is from Matched-Pairs test and determines the risk by which the error corrective tuning decreases the error rate for that row. The default mixture size (192) is used except in Baum-Welch* where a larger mixture (494) was experimented.*

In table 7.10 the improvements obtained by corrective tuning (version v5) for different model and ML training combinations are presented for SCHMMs. The Baum-Welch training is the same as the default training for SCHMMs, but the Viterbi training, despite of being iterated 15 times, turned out to be inadequate in some tests lowering the average error rates. Like in table 7.9 for CDHMMs the improvements are clearly significant for most of the combinations.

Chapter 8

CONCLUSIONS

8.1 Final interpretations

In this work several variations of continuous density and semi-continuous hidden Markov models (CDHMM and SCHMM) are tested as a part of the speech recognition system of the Laboratory of Information and Computer Science at Helsinki University of Technology. The closest examination is directed towards the comparison of different learning schemes and especially to study the combinations of neural network methods and more conventional statistical parameter estimation methods based on the maximum likelihood (ML) principle. The experimented neural network methods are Self-Organizing Maps (SOM) and Learning Vector Quantization (LVQ), both developed in this laboratory.

By initializing the mean vectors of the mixture Gaussian densities for CDHMMs and SCHMMs using SOMs, the required number of computationally heavy ML iterations can be remarkably reduced in order to achieve models of low recognition error rates. The advantage offered by SOMs is related to the way by which the topological characteristics of the input vector space is preserved while the dimensionality is vastly reduced and the distortion is minimized. Because in each learning step each input sample affects several nearby codebook vectors the asymptotic density of the codebook vectors is closer to the input density than in, for example, the traditional K-means which minimizes only the distortion. This seems to be a suitable initialization for ML type algorithms aiming to good representation of the features of the input data.

The motivation for the application of LVQ to the HMM initialization is to emphasize the discrimination between different states by minimizing the classification error of the mean vectors. For SCHMMs the initialization by LVQ yielded significantly lower average recognition error rates than the initialization by SOM.

In order to boost up the recognition phase the fast approximation of the Gaussian mixture output density computation is experimented. It is based on the assumption that the search of a few nearest Gaussians can be performed faster than the pdf computation using the whole codebook of mixture Gaussians. According to the experiments this approximation produces lower recognition error rate for the CDHMMs than the calculation observation probabilities using all N Gaussians of the mixture density functions. The lowest error rates were obtained by using only $n = 2$ or $n = 3$ nearest Gaussians which is assumed to be due to the increased discrimination between the different states, because then only the relations to the most representative mixture densitities affects to the resulting observation probabilities. For the SCHMMs the suitable value seems to be $n = N/20$ which reduces the computational load significantly without increasing the error rate essentially.

After the CDHMMs and SCHMMs have been estimated and used for the recognition of spoken words the models can still be enhanced to improve the recognition performance by the corrective tuning. The corrective tuning methods experimented in this work are all based on the application of LVQ to tune the output densities of the states in order to avoid the observed recognition errors. The obtained results indicate that in this way the average error rates can be decreased significantly for both CDHMMs and SCHMMs.

As a summary of the best training combination for CDHMMs the Gaussian mean vectors are trained by SOMs followed by LVQ2 and then the models are completed by Viterbi training and finally tuned by version v5 of the LVQ-based corrective tuning if even higher accuracy is required. Roughly the same scheme suits best also for the SCHMMs except that the Viterbi training can be substituted by the Baum-Welch reestimation for better performance.

The CDHMM configuration of 5 states and 24 mixtures per state seems to suitable leading to an average error rate of about 5.6 % in the experimented database. If more speed is required the number of mixtures can be dropped to a less than half with little loss of accuracy. For SCHMMs the configuration of 5 states and 192 tied mixtures used in most experiments here turned out to be, however, almost at the limit to be inadequate and produced an error rate about 9.0 %. The best error rate for a larger number of tied mixtures (494) was about 7.9 %.

The comparison of the error rates between SCHMMs and CDHMMs is avoided, because due to the different model structures they are not directly comparable. For example, the number of density parameters required for modeling all phonemes by CDHMMs of 5 states and 24 mixtures is about 58000 and the corresponding number of for SCHMMs of 5 states and 192 tied mixtures is only about 25000 (64000 for 494 tied mixtures). Since the tied mixtures are common for all states the parameters for mean vectors in SCHMMs are more efficiently used and estimated than in CDHMMs, but, however, the major part of the mixture weights will be almost zero and mostly useless so that the number of parameters are not fully comparable either.

8.2 Future considerations

To further develop the training algorithms presented in this report the greatest challenge is perhaps the complete substitution of the ML training by creating adaptive and discriminative algorithms for estimation of the covariances and weight factors of the mixture Gaussian pdfs. The related methods, like the radial basis functions (RBF), should then be studied more carefully for possible cross-exploitation of training algorithms.

Some interesting details left without discussion so far in this project concern individual adaptation of the number of mixtures for the HMM states and the further development of labeling of distributions in SCHMMs to be used as the basis of the LVQ tuning. These optimizations would lead to more efficient use of the parameters and better discrimination between the states.

The principle of tying parameters as described in section 2.5.3 could be more carefully exploited by tying parameters inside the HMMs because the states of the same HMM have certain common features. Also the similarities inside certain groups of phonemes offer possibilities for more efficient modeling by applying tying.

Since the different enhancements for HMMs and speech recognition are recently widely researched there are several ideas that would be applied together with the training combinations developed in this work. The duration modeling of the phonemes, the application of higher dimensional feature vectors with more context information and the division of the features into several codebooks to be joined by the HMM framework are perhaps the most promising areas for further studies.

The next practical step in this project is, however, the integration of the proposed modeling and training methods into the online speech recognition demonstration system of the Laboratory of Information and Computer Science at Helsinki University of Technology. The first online experiments seem to indicate that using the current equipments and programs the recognition speed is not yet adequate, but with some boosting the required speed and accuracy combination can probably be achieved.

Bibliography

Amari, S. (1967). A theory of adaptive pattern classifiers. *IEEE Trans. Electron. Comput.*, **16**, 299–307.

Averbuch, A. (1987). Experiments with the TANGORA 20000 word speech recognizer. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 701–704, New York.

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

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

Bahl, L., Jelinek, F., & Mercer, R. (1983). A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **5**(2), 179–190.

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

Baker, J. M. (1989). DragonDictate$^{\mathrm{TM}}$-30K: Natural language speech recognition with 30,000 words. *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech89)*, volume 2, 161–163, Paris, France.

Bakis, R. (1976). Continuous speech word recognition via centisecond acoustic states. *Unpublished paper presented at the meeting of the Acoustic Society of America.*

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

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

Baum, L., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Stat.*, **21**(1), 164–171.

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

Bengio, Y., de Mori, R., Flammia, G., & Kompe, R. (1991). Global optimization of a neural network — hidden Markov model hybrid. *Proceedings of the International Joint Conference on Neural networks (IJCNN)*, volume II, 789–794, Seattle, WA, USA.

Bourlard, H. & Wellekens, C. J. (1988). Links between Markov models and multi-layer perceptrons. Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 1*, 502–510. Morgan Kaufmann Publishers, San Mateo, CA.

Broomhead, D. & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, **2**(2), 321–355.

Davis, S. B. & Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **28**(4), 357–366.

Dempster, A., Laird, N., & Rudin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, **39**(1), 1–38. Ser. B.

Epraim, Y., Dembo, A., & Rabiner, L. (1989). A minimum discrimination information approach for hidden Markov modeling. *IEEE Transactions on Information Theory*, **31**, 1001–1013.

Epraim, Y. & Rabiner, L. (1988). On relations between modeling approaches for information sources. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 24–27.

Forney, G. D. (1973). The Viterbi algorithm. *Proceedings of the IEEE*, volume 6, 268–278.

Fort, J. C. (1988). Solving a compinatorial problem via self-organizing process: an application of the Kohonen algorithm to the Traveling Salesman Problem. *Biological Cybernetics*, **59**(1), 33–40.

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

Furui, S. (1989). *Digital Speech Processing, Synthesis and Recognition*. New York, Basel, Marcel Dekker Inc.

Gillick, L. & Cox, S. (1989). Some statistical issues in the comparison of speech recognition algorithms. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 532–535, Glasgow, Scotland.

Hämäläinen, A. (1992). *Itseorganisoituvan piirrekartan käyttö tiheysfunktion estimoimiseen*. Licentiates thesis, University of Jyväskylä, Jyväskylä, Finland. (in Finnish).

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

Huang, X. & Jack, M. (1989b). Unified techniques for vector quantization and hidden Markov modeling using semi-continuous models. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, 639–642, New York.

Iwamida, H., Katagiri, S., McDermott, E., & Tohkura, Y. (1990a). A hybrid speech recognition system using HMMs with an LVQ-trained codebook. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, 489–492.

Iwamida, H., Katagiri, S., McDermott, E., & Tohkura, Y. (1990b). A hybrid speech recognition system using HMMs with an LVQ-trained codebook. *J.Acoust.Soc.Jpn*, **11**(5), 277–286.

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

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

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

Juang, B.-H. & Katagiri, S. (1992b). Discriminative training. *J.Acoust.Soc.Jpn*, **13**(6), 333–339.

Juang, B.-H. & Rabiner, L. R. (1990). The segmental K-means algorithm for estimating parameters of hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **38**, 1639–1641.

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

Kasslin, M., Kangas, J., & Simula, O. (1992). Process state monitoring using self-organizing maps. Aleksander, I. & Taylor, J., editors, *Artificial Neural Networks, 2*, II–1531–1534. North-Holland.

Katagiri, S., Lee, C.-H., & Juang, B.-H. (1991). New discriminative training algorithms based on the generalized probabilistic descent method. *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, 298–308, Princeton, New Jersey, USA.

Kohonen, T. (1982). Clustering, taxonomy, and topological maps of patterns. *Proceedings of the 6th International Conference on Pattern Recognition*, volume 1, 114–128, Munich, Germany.

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

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

Kohonen, T. (1988a). An introduction to neural computing. *Neural Networks*, **1**(1), 3–16.

Kohonen, T. (1988b). The 'Neural' Phonetic Typewriter. *IEEE Computer*, **21**(3), 11–22.

Kohonen, T. (1989). *Self-Organization and Associative Memory*. Berlin-Heidelberg-New York-Tokio, Springer-Verlag, 3 edition.

Kohonen, T. (1990a). Improved versions of Learning Vector Quantization. *Proceedings of the International Joint Conference on Neural networks (IJCNN)*, volume 1, 545–550, San Diego, California.

Kohonen, T. (1990b). The Self-Organizing Map. *Proceedings of the IEEE*, 1464–1480.

Kohonen, T. (1990c). Statistical pattern recognition revisited. Eckmiller, R., editor, *Advanced Neural Networks*, 137–144. Elsevier (North Holland).

Kohonen, T. (1991). Workstation-based Phonetic Typewriter. *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, Princeton, New Jersey, USA.

Kohonen, T. (1992). New developments of Learning vector Quantization and the Self-Organizing map. *Symposium on Neural Networks; Alliances and Perspectives in Senri (SYNAPSE'92), Senri International Information Institute, Osaka, Japan, June 24-26.*

Kohonen, T., Barna, G., & Chrisley, R. (1988a). Statistical pattern recognition with neural networks: Benchmarking studies. *Proceedings of the International Conference on Neural Networks*, I 61–68, San Diego.

Kohonen, T., Kangas, J., Laaksonen, J., & Torkkola, K. (1992). LVQ_PAK: A program package for the correct application of learning vector quantization algorithms. *Proceedings of the International Joint Conference on on Artificial Neural Networks*, Baltimore.

Kohonen, T., Mäkisara, K., & Saramäki, T. (1984). Phonotopic maps - insightful representation of phonological features for speech recognition. *Proceedings of 7th International Conference on Pattern Recognition*, 182–185, Montreal, Canada.

Kohonen, T., Torkkola, K., Shozakai, M., Kangas, J., & Ventä, O. (1988b). Phonetic Typewriter for Finnish and Japanese. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 607–610, New York City, USA.

Komori, T. & Katagiri, S. (1992). GPD training of dynamic programming-based speech recognizers. *J.Acoust.Soc.Jpn*, **13**(6), 341–349.

Kosko, B. (1990). Stochastic competitive learning. *Proceedings of the International Joint Conference on Neural networks (IJCNN)*, volume 2, 215–226, Kyoto, Japan.

Kullback, S. & Leibler, L. (1951). On information and sufficiency. *Ann. Math. Stat.*, **22**, 79–86.

Kurimo, M. (1991). Continuous density hidden Markov models in a speech recognition system. Special Assignment for the Laboratory of Computer and Information Science in Helsinki University of Technology. (In Finnish).

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

Kurimo, M. (1993). Using LVQ to enhance semi-continuous hidden Markov models for phonemes. *Proceedings of 3rd European Conference on Speech Communication and Technology*, volume 3, 1731–1734, Berlin, Germany.

Kurimo, M. (1994). Corrective tuning by applying LVQ for continuous density and semi-continuous markov models. *Proceedings of International Symposium on Speech, Image Processing and Neural Networks*, Hong Kong. To be published.

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

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

Kurimo, M. & Torkkola, K. (1992c). Training continuous density hidden Markov models in association with self-organizing maps and LVQ. *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, 174–183, Copenhagen, Denmark.

Lee, K.-F. (1988). Large-vocabulary speaker-independent continuous speech recognition: The SPHINX system. Technical Report CS-88-148, Carnegie Mellon University.

Lee, K.-F. (1990). Context-dependent phonetic hidden Markov models for speaker-independent continuous speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **38**(4), 599–609.

Leinonen, L., Kangas, J., Torkkola, K., & Juvas, A. (1991). Pattern recognition of hoarse and healthy voices by the self-organizing map. Kohonen, T., Mäkisara, K., Simula, O., & Kangas, J., editors, *Artificial Neural Networks*, II–1385–1388. North-Holland.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys.-Dokl.*, **10**(8), 107–710.

Levinson, S. (1983). Continuously variable duration hidden Markov models for automatic speech recognition. *Computer, Speech and Language*, **1**(1), 29–45.

Linde, Y., Buzo, A., & Gray, R. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communication*, **28**(1), 84–95.

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

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

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

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

Makino, S., Endo, M., Sone, T., & Kido, K. (1992). Recognition of phonemes in continuous speech using a modified LVQ2 method. *J.Acoust.Soc.Jpn*, **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).

Mäntysalo, J., Torkkola, K., & Kohonen, T. (1992). LVQ-based speech recognition with high-dimensional context vectors. *Proceedings of the International Conference on Spoken Language Processing*, volume 1, 539–542, Banff, Canada.

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

Merialdo, B. (1988). Phonetic recognition using hidden Markov models and maximum mutual information training. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 111–114.

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

Mizuta, S. & Nakajima, K. (1992). A discriminative training method for continuous mixture density HMMs and its implementation to recognize noisy speech. *J.Acoust.Soc.Jpn*, **13**(6), 389–393.

Monte, E. (1992). Smoothin HMMs by means of a SOM. *Proceedings of 1992 International Conference on Spoken Language Processing*, volume 1, 551–554, Banff, Canada.

Nadas, A., Nahamoo, D., & Picheny, M. (1988). On a model-robust training method for speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **36**, 1432–1436.

Nakagawa, S. & Hirata, Y. (1990). Comparison among time-delay neural networks, LVQ2, discrete parameter HMM and continuous parameter HMM. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, 509–512.

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

Poritz, A. B. (1988). Hidden Markov models: A guided tour. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, 7–13.

Rabiner, L., Levinson, S., & 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., & 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.

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

Rainton, D. & Sagayama, S. (1992b). Minimum error classification training of HMMs - implementation details and experimental results. *J.Acoust.Soc.Jpn*, **13**(6), 379–387.

Ritter, H. (1989). Asymptotic level density for a class of vector quantization processes. Technical Report TKK-F-A9, Helsinki University of Technology.

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

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

Ritter, H. & Schulten, K. (1988). Convergence properties of Kohonen's topology conserving maps: Fluctuations, stability, and dimension selection. *Biol. Cybern.*, **60**(1), 59–71.

Robbins, H. & Munro, S. (1951). A Stochastic approximation method. *Annals of Mathematical Statistics*, **22**, 400–407.

Tohkura, Y. (1987). A weighted cepstral distant measure for speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **35**(10), 1414–1422.

Torkkola, K. (1991). *Short-Time Feature Vector Based Phonemic Speech Recognition with the Aid of Local Context.* PhD thesis, Helsinki University of Technology, Espoo, Finland.

Torkkola, K., Kangas, J., Utela, P., Kaski, S., Kokkonen, M., Kurimo, M., & Kohonen, T. (1991a). Status report of the finnish phonetic typewriter project. *Proceedings of ICANN*, volume 1, 771–776, Espoo,Finland.

Torkkola, K., Kokkonen, M., Kurimo, M., & Utela, P. (1991b). Improving short-time speech frame recognition results by using context. *Proceedings of the 2nd European Conference on Speech Technology and Communication*, volume 2, 793–796, Milano,Italy.

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

Utela, P., Kaski, S., & Torkkola, K. (1992). Using phoneme group specific LVQ-codebooks with HMMs. *Proceedings of the International Conference on Spoken Language Processing*, volume 1, 551–554, Banff, Canada.

W.Chou, Juang, B., & Lee, C. (1992). Segmental GPD training of HMM based speech recognizer. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, 473–476, Baltimore,USA.

Young, S. (1992). The general use of tying in phoneme-based HMM speech recognizers. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, 569–572, Baltimore,USA.

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

Zhang, C.-X. & Mlynski, D. A. (1990). VLSI-placement with a neural network model. *Proceedings of the International Symposium on Circuits and Systems, New Orleans, Luisiana, May*, 475–478.

Zhao, Z. (1992). Integration of neural networks and HMMs for continuous speech recognition. *Proceedings of 1992 International Conference on Artificial Neural Networks*, volume 1, 779–782, Brighton, UK.

Zhao, Z. & Rowden, C. (1991). Application of Kohonen self-organising feature maps to smoothing parameters of hidden markov models. *Proc. of ICANN*, 175–179, Bournemouth, UK.