The State Based Mixture of Experts HMM with Applications to the Recognition of Spontaneous Speech

Andreas Tuerk

Emmanuel College
and
Cambridge University Engineering Department

September 2001

Dissertation submitted to the University of Cambridge
for the degree of Doctor of Philosophy

Abstract

Although the performance of speech recognition systems has increased substantially over the last decades, there still remain a number of tasks which pose considerable problems for current state-of-the-art techniques. One of these tasks is the recognition of spontaneous speech which differs from read or planned speech in that its underlying dynamics change frequently over time. The negative effect of changes in acoustic background condition on recognition performance can also be observed in other situations as, for instance, in the case of speech that is corrupted by non-stationary noise.

This thesis is concerned with the development of an acoustic model for speech recognition which automatically detects changes in the background condition of a signal and compensates for the model-data mismatch by combining the information of several expert models. These experts are specialised on the different acoustic conditions under consideration and their influence on the recognition process is determined by how well their associated condition matches the signal. This approach gives rise to the state based mixture of experts hidden Markov model (SBME-HMM) which is studied in this thesis both theoretically and in a number of recognition experiments. In principle, the SBME-HMM can be applied to distinguish implicitly between an arbitrary set of discrete acoustic conditions. Since, however, the main focus in this thesis is the application of the SBME-HMM to spontaneous speech the only conditions considered here will correspond to speech at different speaking rates.

The SBME-HMM is an extension of the standard HMM which uses an additional hidden variable v whose states are meant to correspond to the different acoustic conditions in the speech signal. The decision whether an acoustic condition is present is implemented in the SBME-HMM via a so-called indicating feature which is a continuous feature that contains information about the state of the hidden variable v. This information is expressed in the SBME-HMM by a posterior probability distribution over the states of v given the indicating feature. The theoretical development of the SBME-HMM in this thesis concerns both the estimation of the model parameters with the EM algorithm and its application in speech recognition. Special attention is given to the estimation of the posterior probability distributions over the states of the hidden variable v which are modelled by softmax functions with polynomial exponents. It is shown that training these functions with the EM algorithm leads to an optimisation problem which can be linked to the cross-entropy error function. Although there are no closed form reestimation formulae for the softmax parameters they can be estimated robustly with an iterative scheme like the Newton-Raphson algorithm with line search and back-tracking. This is due to the convexity of the cross-entropy error surface which is asserted by proving the positive definiteness of the Hessian of the cross-entropy error with respect to the softmax parameters. In addition, this thesis addresses the problem of initialising the SBME-HMM and develops two different methods, namely the median split initialisation (MSI) and the relabelled training initialisation (RTI) which can initialise indicating features whose output probability density functions (pdf's) are either Gamma densities or Gaussian mixtures. For recognition three different types of model topology are proposed that either use the state of the hidden variable v explicitly in the recogni-

tion process or which sum over all the states of v and thereby reduce the model topology to that of a standard HMM.

The indicating feature in the SBME-HMM is the main source of information about the state of v. To ensure that the latter model the acoustic conditions of interest and that the indicating feature distinguishes well between the different states of v it is necessary to find an indicating feature that is highly correlated with the acoustic conditions under consideration. Since this thesis applies the SBME-HMM to the recognition of spontaneous speech a number of indicating features are derived whose usefulness is evaluated in classification experiments on fast and slow instances of phones. Two of these features which show good classification performance are later used in recognition experiments.

The performance of the SBME-HMM's for different combinations of initialisation, model topology and indicating features is evaluated on the 1997 Broadcast News task which contains a substantial proportion of spontaneous speech. The SBME-HMM's in the recognition experiments are used to rescore cross-word triphone lattices and hidden variables v with both 2 and 3 states are explored. The recognition experiments with a binary hidden variable show small but consistent improvements over a standard HMM with a comparable number of parameters giving in one particular experiment a statistically significant overall improvement of 2.8% relative. For a variable v with 3 states no additional improvements over the SBME-HMM's with a binary hidden variable can be observed. On the contrary, the high number of model parameters and the fast saturation of the training likelihoods with the EM algorithm lead to a small performance degradation.

Declaration

This dissertation is the result of my own work, and where it draws on the work of others this is acknowledged at the appropriate points in the text. Some of the work has been published previously in conference proceedings (Tuerk and Young, 1999; Tuerk and Young, 2001a) and a technical report (Tuerk and Young, 2001b). The length of this dissertation including appendices and footnotes is approximately 44,000 words and it contains 30 figures.

Acknowledgements

First and foremost I would like to thank my supervisor Steve Young whose constructive criticism ensured that I did not get lost in some of the details of the work presented here and that I stayed on a more or less straight path leading towards the completion of this thesis. Thanks are also due to Phil Woodland for helping me out with rescoring some especially reluctant lattices and to Mark Gales for helpful theoretical discussions. I would also like to express my gratitude to Patrick Gosling for providing such a reliable computer network which ensured that I could concentrate exclusively on my research rather than having to worry about the computational environment. Furthermore, I would like to take the opportunity to thank all the unnamed people in the Fallside and Speechlab for stimulating work related and sometimes even more stimulating not work related discussions.

Thanks are also due to the Engineering and Physical Sciences Research Council which partly funded this work and the Engineering Department and Emmanuel College for providing me with the necessary financial support to attend a number of conferences.

In addition, I would like to thank my girlfriend Birgit especially for her caring support during my long-lasting dental crisis which meant that I did not lose all hope during this critical period. Last but not least, I would also like to express my gratitude to my mother Ada for both her financial and emotional support during the last months of work without which this thesis would not have been completed.

Contents

List of Figures

List of Tables

Notation

o	standard observation vector
d	indicating feature
$O = \{o_1, \dots, o_T\}$	sequence of T standard observation vectors
$D = \{d_1, \dots, d_T\}$	sequence of T indicating features
s, v	hidden variables in HMM and SBME-HMM
$\vec{s} = \{s_1, \dots, s_T\}$	sequence of T states of variable s
$\vec{v} = \{v_1, \dots, v_T\}$	sequence of T states of variable v
$a_{i,j}$	transition probabilities between states i and j of variable s
$b(o\|s), b(o\|s,v)$	output pdf's of feature o given the state of s or the states of s and v
$b(d\|s), b(d\|s,v)$	output pdf's of feature d given the state of s or the states of s and v
$p(v\|d,s)$	probability of observing the states of v given d and the state of s
$L_\lambda(O, \vec{s})$	likelihood of observing O and $\vec{s}$ under model λ
$L_\lambda(O, D, \vec{s}, \vec{v})$	likelihood of observing O, D, $\vec{s}$ and $\vec{v}$ under model λ
$\alpha_t(i)$, $\beta_t(i)$	forward and backward sequence for standard HMM at time t
$\alpha_t(i,k)$, $\beta_t(i,k)$	forward and backward sequence for SBME-HMM at time t
$\gamma_t(i) = \frac{L_\lambda(O, s_t = i)}{\sum_i L_\lambda(O, s_t = i)}$	probability of state i in standard HMM at time t
$\gamma_t(i,k) = \frac{L_\lambda(O, D, s_t = i, v_t = k)}{\sum_{i,k} L_\lambda(O, D, s_t = i, v_t = k)}$	probability of states i and k in SBME-HMM at time t
$Q(\lambda, \bar{\lambda})$	auxiliary function
$\Gamma(\nu)$	Gamma function
$\psi(\nu) = \frac{\Gamma'(\nu)}{\Gamma(\nu)}$	digamma function
$S_{v,s}(d)$	softmax function for states of v and s
$c_l^{v,s}$	softmax parameters
∇f	gradient of function f
$\nabla^2 f$	Hessian of function f
$\mathbf{c}_i = \{c_l^{v,i} : l = 0, \dots, L_v \quad v = 1, \dots, K-1\}$	set of softmax parameters for state $s = i$
M', o'	transpose of matrix M or vector o
$P(\vec{w})$	language model probability of word sequence $\vec{w}$

Abbreviations

ASR	Automatic speech recognition
LVCSR	Large vocabulary continuous speech recognition
HMM	Hidden Markov model
EM	Expectation maximisation
SLSD	Squared length of second order derivative
MLP	Multi layer perceptron
MSI	Median Split initialisation
RTI	Relabelled training initialisation
PLP	Perceptual linear prediction
MFCC	Mel-frequency cepstral coefficients
DBN	Dynamic Bayesian network
IOHMM	Input-output HMM
ME	mixture of experts
SBME-HMM	State based mixture of experts HMM
VTLN	vocal tract length normalisation
ROS	rate of speech

1

Introduction

1.1 Problem statement

The scope and quality of automatic speech recognition (ASR) systems has increased considerably over the last decades, moving from isolated word recognition with small vocabularies (Fanty and Cole, 1991) to large vocabulary continuous speech recognition (LVCSR) systems. The latter now also include automatic segmentation and speaker adaptation (Woodland et al., 1998; Woodland et al., 1999) and have been successfully applied to difficult tasks like the transcription of broadcast news (Pallett et al., 1998; Pallett et al., 1999).

In spite of these big advances, there still remain various types of data on which recognition performance is less than satisfactory. Spontaneous speech, for instance, still poses a considerable problem with error rates that are almost twice as high as on planned speech. This decrease in performance is, among other things, a result of the fact that fast and slow speech exhibit different acoustic and dynamic characteristics which the common models cannot capture all at once. The approach that has been taken to deal with this problem is to train expert models which are appropriate for only one of the conditions, i.e. fast or slow speech, and apply them in the recognition process whenever fast or slow speech is assumed to be present.

This method, of switching between different acoustic models depending on the type of speech to be recognised, is not restricted to spontaneous speech and has been applied to a number of situations where the use of expert models seems more appropriate than the use of one generic model. Gender dependent models are, for instance, employed in many speech recognition systems to recognise male and female speech. Likewise, wide and narrow band models are applied to recognise studio quality and telephone data.

Prior to selecting the expert model the recogniser has to determine which type of acoustic condition is present, e.g. whether a male or female, or whether a fast or slow speaker has to be recognised. In the case of spontaneous speech, for instance, several estimators of the rate of speech (ROS) have been proposed (see chapter 4 for an in-depth discussion) which indicate whether a slow or fast model is appropriate for recognition. The decision which model has to be selected is usually taken for a whole segment of speech which contains several words.

Summarising the previous discussion, the complete model selection process therefore involves the following stages

- Find acoustically homogeneous segments, e.g. segments which only contain either fast or slow speech, or either male or female speech.
- Determine the acoustic condition present in each segment, e.g. whether it was spoken fast or slow or whether it was corrupted by a certain type of background noise.
- Recognise the segment with the expert model appropriate for the acoustic condition, e.g. with a male or female acoustic model or a model that has been trained on the type of background noise detected.

This general frame-work for model selection has a number of disadvantages. First, the segment boundaries have to be placed at word boundaries, in order for the recogniser to be able to hypothesis the correct word sequence. While this makes sense if the changes in acoustic condition are correlated with word boundaries, as in the case of speaker changes (Gish and Schmidt, 1994; Siegler et al., 1997; Hain et al., 1998), it is less reasonable if such changes can occur at arbitrary points in time, as, for instance, in the case of background noise.

Even if the segment boundaries did not have to coincide with word boundaries, segmentation would still be problematic if the changes that need to be detected occur very frequently. In this case, the duration of the segments can be very small and small absolute errors in the placement of the segment boundaries can result in large errors relative to the segment duration. The use of such subword level acoustic segments is, for instance, suggested by a result in (Mirghafori et al., 1995a). This study which evaluated the effects of speaking rate on recognition performance revealed that the duration of phones has a marked influence on their acoustics. As a result, it seems reasonable to group phones into segments such that all the phones in a segment are all long or short in comparison to their average duration. This will, in general, not only produce segments whose boundaries are located within words but also segments that are shorter than a word because durational variation can, for example, be linked to syllable stress which can change several times within a word.

The second disadvantage of the model selection procedure above is that a “hard” decision has to be made as to which acoustic condition is present for a certain segment. Sometimes, however, a “soft” decision seems more appropriate. If, for instance, a speaker speaks neither fast nor slow but somewhere in-between, this speaker’s segments should not be forced into either the slow or fast class, instead they should be labelled according to the extent to which they are fast or slow. In order to implement such a scheme, one needs to replace the binary label of presence or absence of an acoustic condition with a continuous label which encodes the quality of the match between an acoustic condition and a signal.

The disadvantage of the third stage of the model selection process above is, again, that a hard decision has to be made. This time, it is the decision which model is to be used for recognition. In borderline cases where speech cannot be assigned clearly to one of a number of classes, rather than picking one model and ignoring the rest, it seems more sensible to use all the available models and combine their outputs according to how well their corresponding acoustic conditions match the signal.

Summarising the above discussion one arrives at the following desiderata for a model selection method.

- Replace the concept of a segment which has constant background condition with a continuous measure for the quality of the match between the acoustic conditions and the signal.
- Replace the hard model selection with model combination based on the quality of the match between the acoustic signal and the model's corresponding acoustic condition.

In the second of these objectives the quality of match measure determines the influence of a model on the overall system output and can therefore be regarded as the model's continuous activation level in the recognition process. With this interpretation the hard and soft model selection procedures can now be illustrated by the graph in figure 1.1.

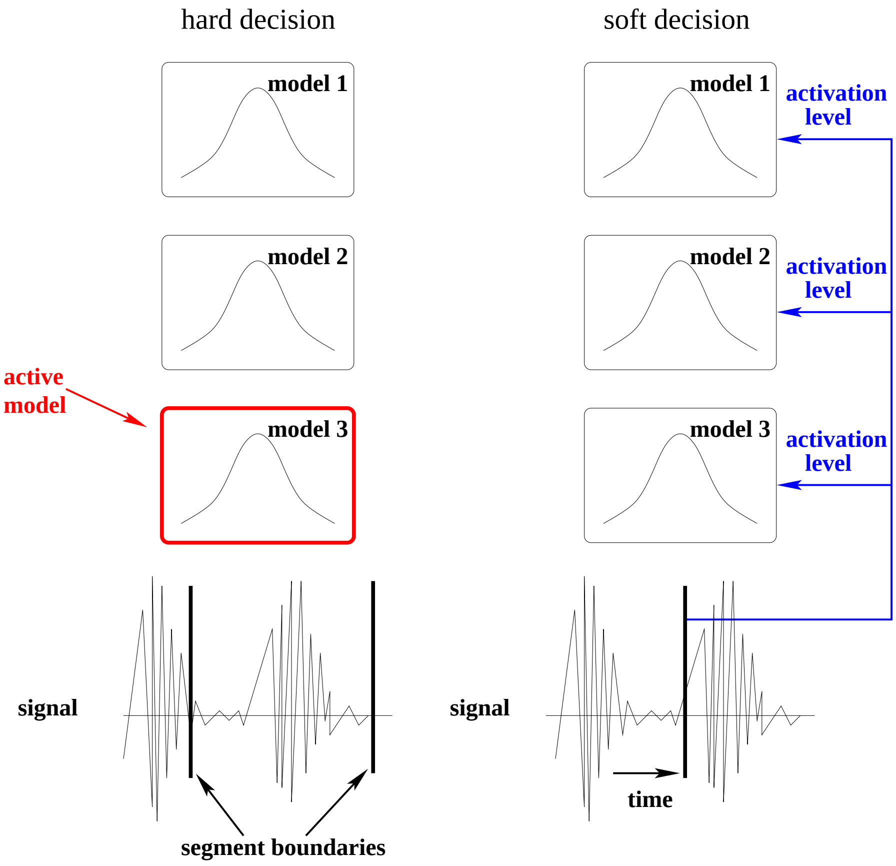


Figure 1.1 *Soft vs. hard model selection.*

1.2 Contribution of this thesis

This thesis addresses the problem of realising the soft model selection approach in an HMM frame-work with a particular focus on the recognition of spontaneous speech. For this purpose, a new model, which will be called a state based mixture of experts hidden Markov model

(SBME-HMM), will be proposed which is an extension of the standard HMM and whose theoretical properties will be studied both with respect to parameter estimation and recognition. In particular, the EM algorithm will be applied to reestimate the model parameters and it will be shown under which conditions there exist solutions to the maximisation step of this algorithm. This thesis will also investigate different methods to derive measures for the quality of the match between acoustic conditions and signals. Since the main interest in this work is the development of a good recognition system for spontaneous speech, these methods are guided by considerations concerning the design of the model rather than high correlation with other speaking rate measures. The measures developed will therefore be optimised with regard to how well they distinguish between the different expert models that will be used for recognition. Finally, the results derived will be combined to evaluate the performance of the proposed model on a large vocabulary speech recognition task with a substantial proportion of spontaneous speech.

1.3 Outline

This thesis is organised in 8 chapters and 3 appendices. The second chapter gives an introduction to the theory of hidden Markov models and presents their application to speech recognition. This chapter also discusses two common feature extraction methods and introduces much of the notation that will be used throughout this thesis.

The third chapter reviews some of the models that were developed to either model hidden dynamics in the speech process more accurately or are conceptually related to these models. As it happens, these models implement different soft model selection schemes some of whose features will later also be used in the development of the state based mixture of experts HMM.

Chapter 4 discusses several of the measures of speaking rate that have been proposed in the literature and relates them to the problem of recognising spontaneous speech.

Chapter 5 introduces the SBME-HMM and compares it to the models in chapter 3. The SBME-HMM uses an additional hidden variable whose states are meant to correspond to the different dynamics of interest. Since this thesis is concerned with spontaneous speech, here, the states will refer to fast and slow instances of phonetic units. The SBME-HMM also uses an additional feature which will be called an indicating feature and from which the activation of the various models, as depicted in figure 1.1, will be inferred. Chapter 5 also contains a comparison between polynomial softmax functions and other models that are used to model posterior probabilities. In addition, this chapter discusses the problem of initialising an SBME-HMM and two different methods will be proposed. Furthermore, 3 different model topologies will be discussed which will be evaluated later together with the initialisation methods in a number of recognition experiments.

Chapter 6 investigates the problem of deriving good indicating features. As mentioned before, the aim here is to develop features that discriminate well between the different expert models and consequently chapter 6 conducts classification experiments on speech segments which are supposed to correspond to sections where one particular expert model has the highest activation level.

Chapter 7 describes recognition experiments that were carried out to evaluate the performance of the SBME-HMM's and compares them to standard HMM's with a similar number of parameters. Here the different initialisation methods introduced in chapter 5 will be investigated as well the different model topologies proposed there.

Chapter 8 reviews the work that has been done in this thesis and proposes new avenues for future research in connection with the SBME-HMM's.

Appendix A gives a list of all the phones which were used in the classification experiments in chapter 6 and lists the sizes of their training and test sets.

Appendix B develops some of the theory of Gamma densities which is relevant to the work in this thesis.

Finally, appendix C gives the exact statements and proofs of the existence and uniqueness results for solutions to the softmax parameter estimation problem.

2

Fundamentals of speech recognition

This section reviews some the basic concepts of speech recognition systems and introduces notation and terminology that will be used throughout this thesis.

2.1 System Overview

The way in which a speech recogniser processes an acoustic signal can be broken down into a number of stages. Figure 2.1 identifies several of the key components that most state-of-the-art recognition systems have in common.

First, the signal is transformed into a sequence of feature vectors $O = \{o_1, \dots, o_T\}$ which are meant to capture the information relevant to the distinction between different sounds, like the transfer function of the vocal tract or certain elements of the short-time spectrum of the signal.

In the next step, the quality of a word hypothesis W given the sequence of features O is determined. In figure 2.1 this hypothesis is "hello world". The quality is evaluated by combining two sources of information. The first source is a measure of the quality of the match between the features o_t and the acoustic models of the recogniser, while the second determines the probability of observing the word sequence W without referring to any acoustic information. In figure 2.1, both numbers are multiplied to give the likelihood of observing W and O simultaneously. In practice, however, these numbers are usually scaled to control the influence of the language model on the overall likelihood score.

The various components of the speech recogniser in figure 2.1 will be discussed in greater detail in the following sections. Section 2.2 considers some of the issues involved in the correct choice of the phonetic units in figure 2.1. Section 2.3 will introduce two of the most common approaches to feature extraction and, finally, section 2.4 discusses the structure of the acoustic models which are indicated in figure 2.1 by Gaussian-shaped curves.

2.2 Phonetic units

The link between the acoustic models and the words of the recognition dictionary is implemented via phonetic units which correspond to certain parts of a word sequence. In figure 2.1,

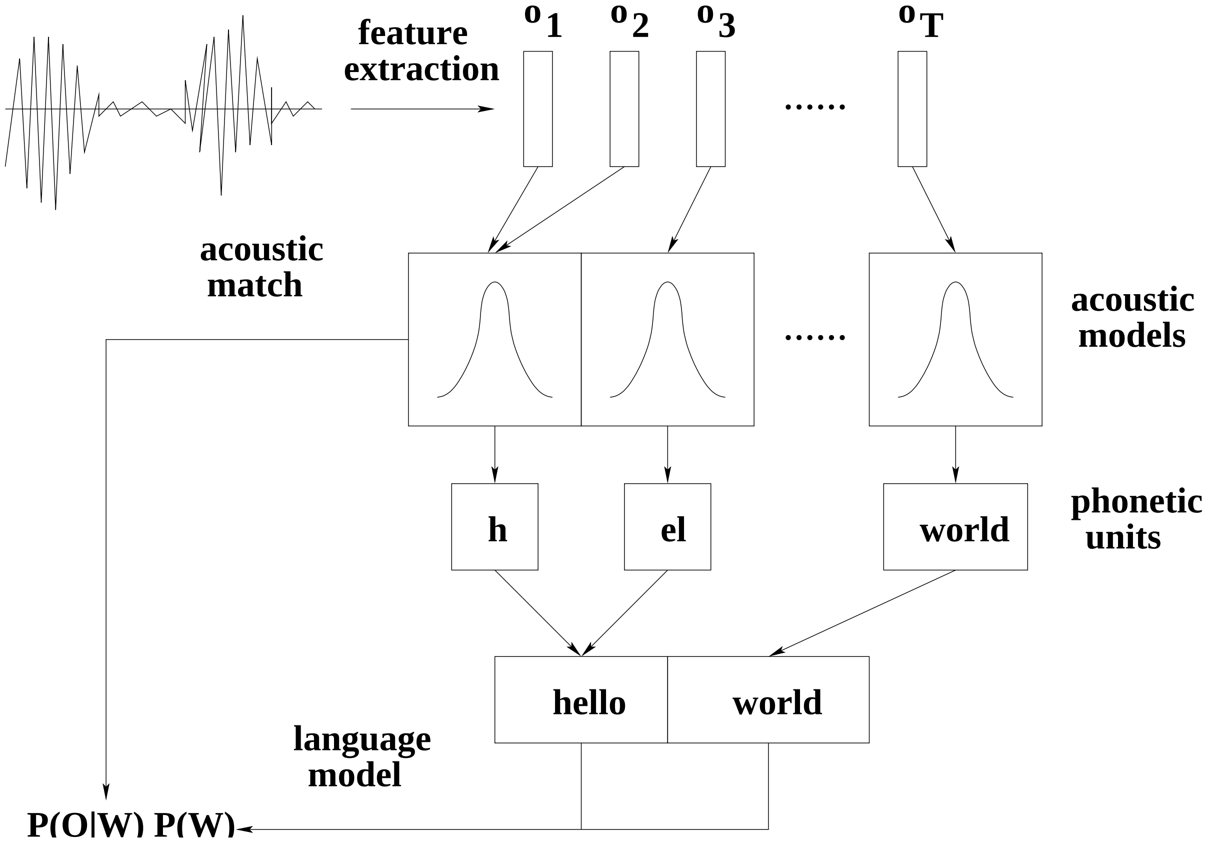


Figure 2.1 *Recognition system overview.*

for instance, the beginning of the word “hello” is made up of two phonetic units namely “h” and “el” while the word “world” has only one phonetic unit whose associated model is meant to capture the whole word.

There are two problems one has to consider when choosing the phonetic units of a speech recogniser. First, there has to be enough training data to train the acoustic models corresponding to the units robustly. Second, the units should exhibit little variability in order for the corresponding models to be discriminative. These two problems are, obviously, at odds. While the first problem can be resolved by defining few phonetic units that have a short time span the second problem is alleviated by defining a large number of phonetic units with a wide time span. In practice, one has to find a trade-off between model robustness and discrimination. This is typically achieved by replacing phonetic units which correspond to complete words or even word sequences by subword phonetic units that take the surrounding context into consideration. While such units are not as discriminative as complete words, the dependency on the context ensures that their variability is kept within reasonable bounds and that coarticulation effects are reflected in the model. Consider, for instance, the phone “ay” in the following phonetic transcriptions

horizon hh axr ay z ax n

horrified hh ao r ax f ay d

where the phonetic symbols and their values are defined in appendix A. Clearly the two instances of the phone "ay" sound rather different and it is therefore sensible to distinguish between "ay" in the context of "axr" and "z" and "ay" in the context of "f" and "d". This gives rise to a subword unit called a triphone, which will be denoted as follows

(left context)-(central phone)+(right context)

The two instances of phone "ay" above would therefore be denoted by "axr-ay+z" and "f-ay+d", respectively. This frame-work can easily be extended to include contexts of different types and length. A subword unit that considers two predecessors and two successors is, for instance, called a quinphone. Subword units which span word boundaries are referred to as cross-word units. The models that were used in the experiments in section 7 are, for example, cross-word triphones.

2.3 Feature Extraction

This section will give a brief overview over two feature extraction methods which together with some of their variants have become common practice in most speech recognition systems. The methods discussed are perceptual linear prediction (PLP) and mel-frequency cepstral coefficients (MFCC). Since both draw upon the properties of the human auditory system, some of the relevant psychoacoustical results will be presented in the next section.

2.3.1 Psychoacoustical motivation

The scales which are used to measure the frequency and intensity of a sound are usually Hertz and dB. Although these are useful scales from a physical point of view they are less appropriate when measuring the perception of sound. This section will discuss some of the discrepancies of these two approaches and introduce some of the elements of psychoacoustical research which are relevant to feature extraction in speech recognition.

2.3.1.1 Auditory filters

In 1940 Fletcher conducted an experiment (Fletcher, 1940) to determine the influence of noise on the perception of a pure sinusoid. The centre frequency of the noise was identical to the frequency of the masked signal and the bandwidth of the noise was varied. The subjects were asked to determine the minimal intensity level of the sinusoid at which the sinusoid could be distinguished from the noise. The experiment showed that up to a certain noise bandwidth the minimal intensity had to be increased, but beyond this bandwidth the minimal intensity level remained constant. Since this indicated that noise outside a certain frequency band is removed in the auditory process, Fletcher assumed that the pure sinusoid was perceived after having been filtered by a bandpass filter whose centre frequency is the frequency of the sinusoid. This filter has been called an auditory filter and its bandwidth is the critical bandwidth.

The shape of the auditory filter depends on its central frequency. For high frequencies, for instance, the critical band is wider than for low frequencies. The exact shape of the auditory filter has been the subject of a number of studies. In (Patterson, 1976), Patterson developed the "notched noise method" to show that the frequency response of an auditory filter is close to a triangle. This method, which assumed that the auditory filters are symmetric was later generalised to avoid this assumption (Patterson and Nimmo-Smith, 1980). The results of this study show that the auditory filters are reasonably symmetric for moderate sound levels but become increasingly asymmetric for higher levels, the low frequency side becoming flatter than the high frequency side.

The existence of auditory filters motivates the introduction of filter banks into the processing of an acoustic signal. Such filter banks are defined by selecting a number of representative frequencies together with approximations to their auditory filters. The intensity of an acoustic signal in a particular critical band is then calculated by taking the sum of the power spectrum of the signal weighted by the frequency response of the corresponding filter. An example of such a filter bank is given in figure 2.3. The output of the filter bank represents an approximation to the excitation of the basilar membrane at a set of chosen frequencies. How a set of representative frequencies can be chosen will be the topic of the next section which will also explain the form of the filter bank in figure 2.3 in greater detail.

2.3.1.2 Frequency perception

A human listener perceives frequency on a scale that is different from the linear Hertz scale. This means that a person listening to a test tone which is twice as high on the Hertz scale than a reference tone, will not necessarily perceive the test tone to be twice as high. This phenomenon lead to the introduction of perceptive frequency scales like the Bark (Zwicker, 1961) and the Mel (Fant, 1973) scale. The Bark scale has been derived by segmenting the frequency scale so that each segment corresponds to one critical band and the segments do not overlap and span the complete frequency axis. An approximation to the relationship between Bark Ω and Hertz ω has been proposed in (Schroeder, 1977), and is given by the following expression

$$\Omega(\omega) = 6\ln(\omega/1200\pi + ((\omega/1200\pi)^2 + 1)^{0.5}) \tag{2.1}$$

Unlike the Bark scale, the Mel scale has been defined by directly measuring the perceived pitch of a tone (Stevens and Volkmann, 1940). Test subjects were asked to adjust the frequency of a tone until it was perceived to be a certain factor higher or lower than the original. According to (Merwe and du Preez, 1991), the relationship between Hertz and the Mel frequency M can be approximated as follows

$$M(\omega) = 2595\log_{10}(1 + \omega/700) \tag{2.2}$$

The values on the Mel scale differ from the values on the Bark scale by several orders of magnitude, but if scaled properly they look very similar, as can be seen from figure 2.2. This figure shows the curves defined by (2.1) (solid line) and (2.2) (dotted line), where the latter has been scaled so that both curves have the same maximum at the right hand side of the graph. Both

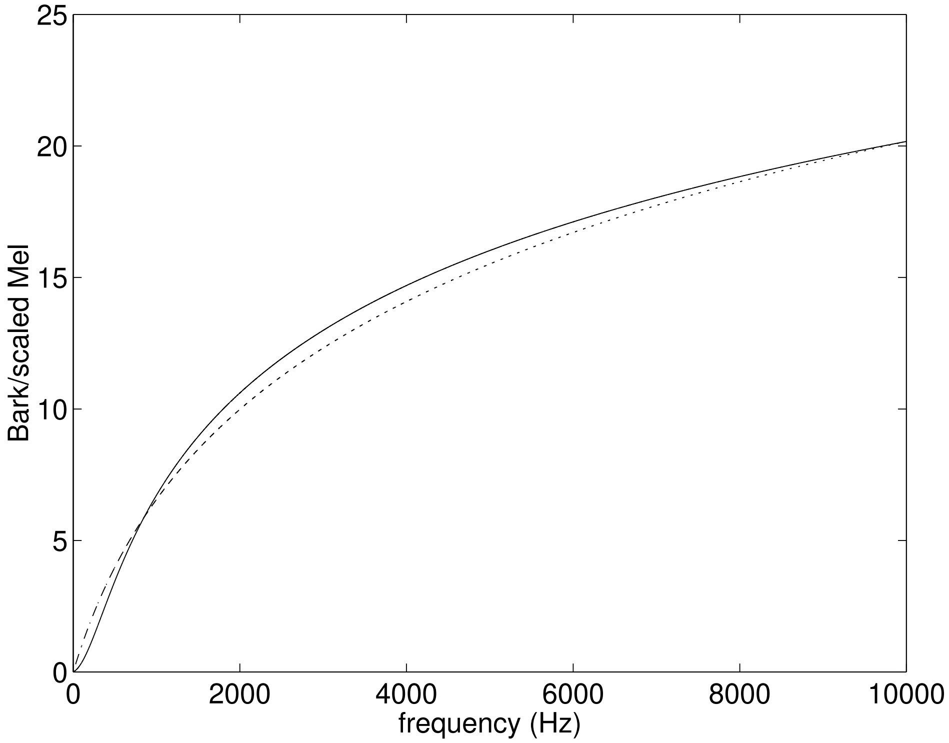


Figure 2.2 *The relationship between the Hertz scale and the Bark (solid line) and Mel (dotted line) scales.*

curves show that the sensitivity to frequency changes decreases with increasing frequency. The varying sensitivity to frequency changes has been incorporated in the processing of acoustical signals by warping the Hertz scale onto a non-linear perceptive scale. This is done by transforming the Fourier transform $\mathcal{F}(\omega)$ of the signal to the Fourier transform on the perceptive scale $\mathcal{F}(P^(\Omega))$ where $P(\Omega)$ is the map from the perceptual scale onto the Hertz scale. As mentioned in the previous section, the signal is also often processed by a filter bank whose filters are located at representative frequencies. Due to the varying sensitivity of the human auditory system, these frequencies are usually chosen to have a constant distance on the perceptive scale. One can realise such filter banks by either warping the signal onto the perceptive scale and applying the filter at equal distances or by warping the filter bank and keeping the original signal unchanged. Figure 2.3 gives an example of a warped filter bank. Here the centre frequencies of adjacent filters have a constant distance on the Bark scale where the filter bandwidths are also constant. In addition, the shape of the filters have been chosen to be triangular which is in accordance with the experimental evidence discussed in the last section.

2.3.1.3 Loudness perception

The relationship between sound intensity and perceived loudness can be determined by requiring test persons to adjust the intensity of a reference tone until it corresponds to the perceived loudness of a test tone. The reference tone is usually a pure sinusoid at 1000Hz. In this case the loudness level of the test tone is measured in "phons" which is equal to the dB level of the

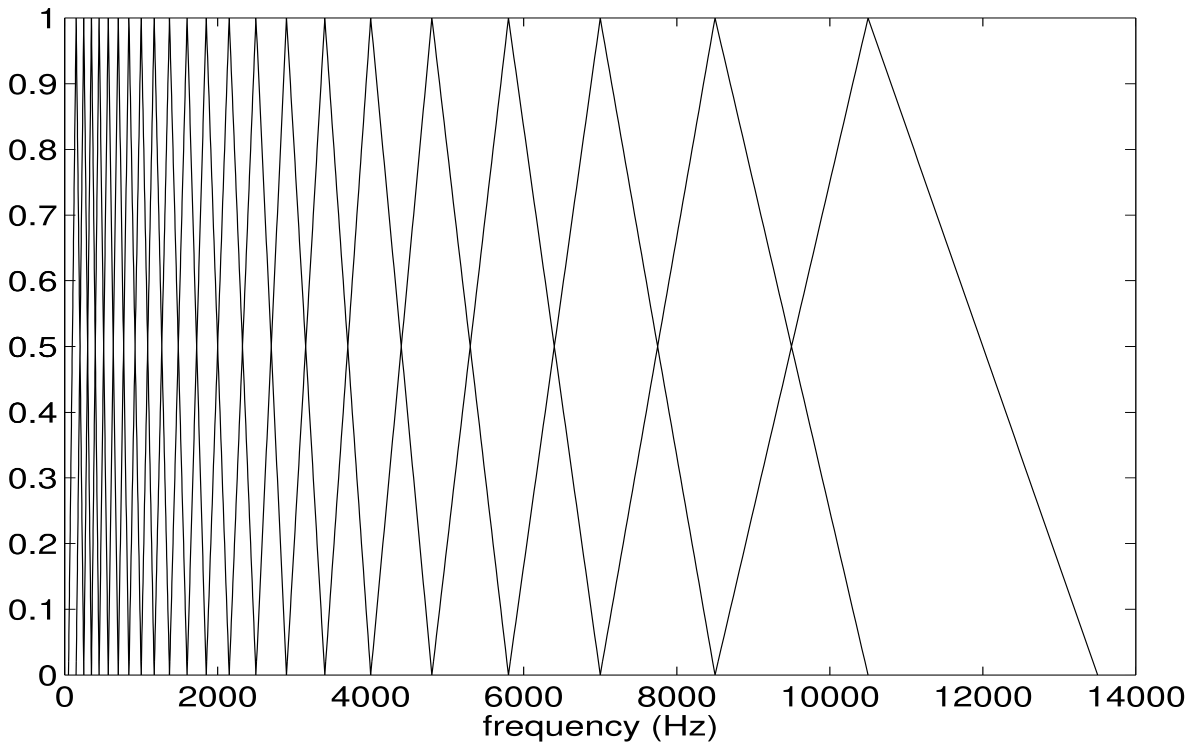


Figure 2.3 *Overlapping filter bank on the Bark scale. The distance between adjacent filter centres is constant on the bark scale.*

adjusted reference tone. If this procedure is repeated for various frequencies of a sinusoidal test sound, an equal loudness contour is generated (Fletcher and Munson, 1933). Experiments as in (Robinson and Dadson, 1956) show that the relationship between sound intensity and perceived loudness varies non-linearly with frequency. Typically, a medium frequency tone will be perceived as loud as a low or high-frequency tone even though its intensity is lower. This is a result of the fact that the human ear is most sensitive to medium frequency signals in a range of approximately 500-5000Hz.

The problem of defining a perceptive loudness scale and its relation to the dB scale has also been addressed by Stevens in (Stevens, 1936; Stevens, 1957). He suggested that the perceived loudness L is related to the intensity I of the sound by the following relationship

$$L = kI^{0.3} \tag{2.3}$$

where k is a constant that depends on the test person. As a consequence of this loudness power law, an increase in intensity of around 10 dB corresponds to a two-fold increase in the perceived loudness.

2.3.2 Perceptual linear prediction

Perceptual linear prediction (PLP) (Hermansky, 1990) attempts to incorporate the differences between physical and perceived signal properties, which were discussed in section 2.3.1, into the feature extraction stage. This is done by first mapping the signal onto a representation that encodes the excitation of the human auditory system at certain frequencies. In a second stage this representation is then approximated by the spectrum of an all-pole model.

To start with the short-time Fourier transform of the signal is calculated, for instance, with a Hamming window which is given by

$$w(n) = 0.54 + 0.46\cos(2\pi n/(N-1)) \tag{2.4}$$

where N is the size of the window and n is the sample index. The power spectrum of the signal is then passed through a filter bank similar to the one in figure 2.3. The main difference to this filter bank is that the individual filters in (Hermansky, 1990) are approximations to the asymmetric masking curve of Schroeder (Schroeder, 1977) and do therefore not have triangular shape.

To accommodate for the difference between perceived and dB loudness at different frequencies the output of the filter bank is preemphasised by a simulated equal-loudness curve which encodes the relationship between frequency and loudness sensitivity. An example of such a function has been given in (Makhoul and Cosell, 1976). Finally, the preemphasised signal is transformed according to the intensity-loudness law mentioned in section 2.3.1 by exponentiation with 0.33.

The output from the previous stage which can be interpreted as an approximation to the excitation of the human auditory system to a particular sound is finally approximated by the spectrum of an all-pole model (Makhoul, 1975) whose coefficients are the PLP features.

2.3.3 Mel-frequency cepstral coefficients

Mel-frequency cepstral coefficients are the real cepstrum of a frequency warped version of the original signal where the real cepstrum of a signal X is defined as follows

$$\mathcal{C}(X) = \mathcal{F}^{-1}(\log|\mathcal{F}(X)|) \tag{2.5}$$

Here $\mathcal{F}$ is, again, the Fourier-transform. The real cepstrum differs from the complex cepstrum in that the logarithm of the absolute value of the Fourier-transform is calculated instead of the logarithm of the Fourier-transform itself.

The advantage of using the cepstrum as a representation for signal X is that it transforms convolution into addition, i.e. $\mathcal{C}(X_1 * X_2) = \mathcal{C}(X_1) + \mathcal{C}(X_2)$. This means that channel and excitation can be easily separated since the variation of the source is usually much slower than that of the excitation and their cepstra are therefore well separated. In this case the so-called process of liftering, which is filtering in the cepstral domain, can either remove the excitation or the channel from the signal representation. This is very useful if the quality of the signal has been corrupted by channel noise as might, for instance, be present when a low quality microphone is used for recording.

Initially the calculation of the Mel-frequency cepstral coefficients (MFCC) proceeds in a similar manner as PLP. First the spectrum of the windowed signal is calculated and the Hertz scale is mapped onto a perceptual scale. In contrast to PLP, however, the calculation of MFCC's uses the Mel-frequency scale M which is approximately related to the frequency in Hertz by the function in (2.2). Now the warped spectrum is sent through a filter bank whose filters are placed at equal distances on the Mel scale and with increasing bandwidth. This can also be achieved by warping the filter bank and processing the unwarped spectrum. In this case the filter bank is similar to the one depicted in figure 2.3. One can see that the width of the filter increases with increasing frequency and that the frequency response of the filters is triangular. This agrees with the results about auditory filters mentioned in section 2.3.1. The final stage of the calculation of the MFCC's is the derivation of the cepstral coefficients which are given by a cosine transform of the logarithm of the filterbank outputs.

2.3.4 Derivatives

The feature extraction methods discussed in section 2.3.2 and 2.3.3 gather information that is relevant at a particular point in time but do not account for temporal changes in the speech process. In order to determine how fast a speaker speaks it might, for instance, be interesting to measure the speed at which the vocal tract of a speaker changes. For this reason, first and second order derivatives of the sequence of feature vectors are often appended to the original feature. Calculating the derivatives directly by taking the difference between adjacent observations can result in a noisy estimator. One, therefore, usually approximates the sequence of features O by a second degree polynomial prior to calculating the derivative. Fitting such a polynomial according to the minimum squares principle results in the following estimate for the derivative.

$$\frac{\sum_{\theta=1}^{\Theta} \theta(o_{t+\theta} - o_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2} \tag{2.6}$$

Here $2\Theta + 1$ is the size of the window where the polynomial is fitted to the data. For many applications it is sufficient to have $\Theta = 2$ and this will be the case in all the experiments that will be described in chapters 6 and 7. Equation (2.6) can easily be adapted to calculate the second order derivatives of the feature sequence O, one only has to substitute o_t by the first order derivative.

2.4 The hidden Markov model

This section will discuss one possible choice of acoustic models in the recognition system in figure 2.1. Other models have also been investigated (Bourlard and Morgan, 1994; Robinson and Fallside, 1991) but hidden Markov models represent the current state-of-the-art.

A Hidden Markov Model models a stochastic process whose realisation at time t is the feature vector o_t. An HMM represents the sequence $O = \{o_1, \ldots, o_T\}$ with two distinct components. The first component is responsible for modelling the temporal evolution of the process and the second is a model of the stationary aspects of the process. The temporal evolution of the stochastic process is represented by a sequence of states $\vec{s} = \{s_1, \ldots, s_T\}$ in the HMM which follow a Markov chain. The fact that only sequence O is observed but not the sequence of states $\vec{s}$ explains why these models are called "hidden" Markov models.

2.4.1 Markov models

A Markov process is defined by a set of states $\{1, \ldots, S\}$, which are manifestations of the variable s, and by transitions between these states. Each transition has a certain probability associated with it, which depends on some of the states that have already been visited. If the transition probability is only dependent on the current state, the Markov process is said to satisfy the first order Markov property. These are the most common Markov processes in speech recognition. The probability of moving from state i to j in such a process depends only on the pair of states i, j and will be denoted by $a_{i,j}$. Since the numbers $a_{i,j}$ are probabilities, they satisfy

$$0 \leq a_{i,j} \leq 1 \tag{2.7}$$

and sum to 1 for each i, i.e.

$$\sum_j a_{i,j} = 1 \tag{2.8}$$

In addition to the transition probabilities, a Markov model also needs a set of initial state probabilities a_i to completely determine the probability of observing a particular state sequence $\vec{s}$. For a first order Markov process this probability is then given by

$$P(\vec{s}) = a_{s_1} \prod_{t=1}^{T-1} a_{s_t, s_{t+1}} \tag{2.9}$$

A typical Markov model of the HMM of a phonetic unit is given in figure 2.4. Here the

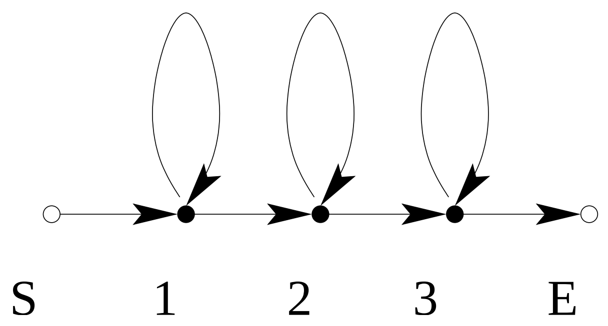


Figure 2.4 *Typical Markov model of a phonetic unit.*

different states of the Markov process are denoted by $1, 2, 3$, S and E and the transitions between states i and j are indicated by an arrow pointing from i to j. The Markov process in figure 2.4 therefore exhibits a left-to-right topology which is common in speech recognition HMM's. The reason for this is that the states in this Markov process are assumed to correspond to successive points in time. A left-to-right topology therefore ensures that the process moves forward.

The S and E states in figure 2.4, which are called the start and exit state, respectively, have a special meaning. They are used to concatenate the HMM's of the different phonetic units to form models that correspond to complete word hypothesis.

2.4.2 Output probabilities

The stationary component of an HMM is a probability density function (pdf) that models the likelihood of observing feature o given that the Markov process is in state i. These output probabilities are denoted by $b(o|i)$. Usually, not all the states in an HMM have output pdf's associated with them, those that do are called emitting states. The start and exit states in the HMM in figure 2.4, for instance, are non-emitting states. As mentioned previously, their only purpose is to connect different HMM's together and they are therefore not associated with a particular observation in the observation sequence O.

There are many possible choices for output pdf's in an HMM. In principle, any non-negative function whose integral is 1 over the feature space is a potential candidate. The most common choice, however, are Gaussian mixture models (GMM's) which are given by

$$\sum_{n=1}^{N} w_n \mathcal{N}(o, \mu_n, \Sigma_n) \tag{2.10}$$

where $\mathcal{N}(o, \mu_n, \Sigma_n)$ is the normal or Gaussian distribution with mean μ_n and covariance matrix Σ_n, i.e.

$$\mathcal{N}(o, \mu_n, \Sigma_n) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_n|}} e^{-\frac{1}{2}(o-\mu_n)' \Sigma_n^{-1} (o-\mu_n)} \tag{2.11}$$

where $|\Sigma_n|$ is the determinant of Σ_n, w_n is the weight of the n-th component in the mixture and D is the dimension of o. The w_n are positive and satisfy

$$\sum_{n=1}^{N} w_n = 1 \tag{2.12}$$

The advantage of Gaussian mixture models over other output pdf's is that they can approximate almost any function with arbitrary precision while, at the same time, their parameters can easily be estimated with the EM algorithm. For some special cases, however, other output pdf's are more appropriate. In chapter 5.5.1, for instance, it will be seen that for a particular one-dimensional feature d, Gamma densities are a more natural choice than GMM's. Gamma densities have a zero likelihood for negative values of d and for $d > 0$ are given by

$$\frac{\eta^\nu}{\Gamma(\nu)} d^{\nu-1} e^{-\eta d} \tag{2.13}$$

Here $\Gamma(\nu)$ is the Gamma function and η and ν are the two parameters of the Gamma density that determine its mean and variance, i.e.

$$\mu = \frac{\nu}{\eta} \qquad \sigma^2 = \frac{\nu}{\eta^2} \tag{2.14}$$

In Speech Recognition, Gamma densities have also been used in the context of duration modelling (Levinson, 1986) where they have been employed to model the probability of remaining in one state of an HMM for a certain amount of time. Although Gamma densities can, in principle, be approximated by GMM's their asymmetric structure means that a GMM may need many more parameters to model the density than the two parameters ν and η of the Gamma density.

In the following, the set of model parameters will always be denoted by λ. For an HMM with GMM's as output pdf's λ is therefore given by

$$\lambda = \{A, \{w_j^i, \mu_j^i, \Sigma_j^i : i \in S,\ j \in C(i)\}\} \tag{2.15}$$

where A is the matrix of transition probabilities and $C(i)$ is the set of components of state i. The weight w_j^i belongs to the j-th component of state i and similar interpretations apply to the other objects μ_j^i and Σ_j^i.

Combining the dynamic and stationary components of an HMM, the likelihood of observation sequence O and state sequence $\vec{s}$ can now be calculated as follows.

$$L_\lambda(O, \vec{s}) = a_{s_1} b(o_1|s_1) \prod_{t=2}^{T} a_{s_{t-1},s_t} b(o_t|s_t) \tag{2.16}$$

Here, the likelihood L was indexed by λ to emphasis the dependency of this value on the model parameters. If this dependency is not important, the index λ will be dropped.

2.5 The HMM/ANN frame-work

Both the standard HMM structure, as described above, and the HMM/ANN frame-work use a Markov process to model the temporal evolution of the speech process. In contrast to the standard HMM, however, the HMM/ANN school of thought replaces the output probability density functions $b(o|i)$ with posterior probabilities $p(i|o)$ of observing state i given observation o. This can be done in a number of ways which are discussed in (Bourlard and Morgan, 1994; Bourlard et al., 1995; Bourlard and Morgan, 1997). The simplest approach uses Bayes theorem to derive a scaled version of the posterior $p(i|o)$, i.e.

$$\frac{p(i|o)}{p(i)} = \frac{b(o|i)}{b(o)} \tag{2.17}$$

The scaled probabilities on the left-hand side of this equation can now be used instead of the output pdf's $b(o|i)$ because $b(o)$, which is the state independent output pdf of o, does not change with state i and does therefore not contribute any information to the recognition process.

Although both models have, in principle, the same modelling power, it is usually argued that the state probabilities $p(i|o)$ have the advantage that they result in a model that discriminates better between different states i.

The reason why these models are mentioned here is that in chapter 5 a model will be proposed that also uses posteriors similar to $p(i|o)$. There, however, the state i and feature o will have a different meaning.

2.6 Recognition

An automatic speech recogniser searches for the most likely sequence of words given a sequence of features O. Since direct optimisation of $P(\vec{w}|O)$ is difficult, Bayes theorem is usually applied to give

$$P(\vec{w}|O) = \frac{P(O|\vec{w})P(\vec{w})}{P(O)} \tag{2.18}$$

Optimising the right-hand side of (2.18) with respect to $\vec{w}$ is equivalent to optimising the following product

$$P(O|\vec{w})P(\vec{w}) \tag{2.19}$$

because $P(O)$ does not depend on the word sequence $\vec{w}$. The recognition problem in a standard HMM therefore usually takes the form of optimising the expression (2.19), which is the same as in figure 2.1.

The first term in (2.19) represents the likelihood of the observation sequence O under the acoustic model, which will from now on be denoted by $L(O|\vec{w})$, while the second term represents the language model of the recogniser. The latter is usually represented by a product of conditional word probabilities, i.e.

$$P(\vec{w}) = \prod_{n=1}^{W} p(w_n|w_{n-1}, \dots, w_1) \tag{2.20}$$

To decrease the complexity of the model the history of the conditional probability in (2.20), $p(w_n|w_{n-1}, \dots, w_1)$, is restricted to a small number of words, say $p(w_n|w_{n-1}, \dots, w_{n-N})$. Such a language model is called a word $N + 1$-gram model. Calculating the language model probability of the word sequence $\vec{w}$, therefore, involves looking up the individual conditional word probabilities $p(w_n|w_{n-1}, \dots, w_{n-N})$ and multiplying them together. Compared to this procedure, the calculation of the acoustic likelihood $P(O|\vec{w})$ in (2.18) is more complicated and an efficient method to calculate its value is mandatory. This will be subject of the next section.

2.6.1 Forward Backward Algorithm

Given an observation sequence O and a sequence of words $\vec{w}$ the acoustic likelihood of O in (2.19) is calculated by summing over the acoustic likelihoods of all state sequences that are possible for word sequence $\vec{w}$, i.e.

$$L_\lambda(O|\vec{w}) = \sum_{\vec{s}\in\vec{w}} L_\lambda(O, \vec{s}) \tag{2.21}$$

Here $\vec{s} \in \vec{w}$ denotes the fact that $\vec{s}$ is a state sequence that can be generated by word sequence $\vec{w}$. Direct evaluation of the sum in (2.21) is computationally too expensive, because the number of terms is of the order of S^T, where S is the number of possible states at any given time and T

is the length of the observation sequence. However, taking advantage of the underlying lattice structure of the problem an algorithm of reasonable complexity can be derived. Firstly, define

$$\alpha_t(i) = L_\lambda(O_1^t, s_t = i) \tag{2.22}$$

$$\beta_t(i) = L_\lambda(O_{t+1}^T | s_t = i) \tag{2.23}$$

where $O_1^t = \{o_1, \dots, o_t\}$ is the observation sequence truncated at time t and $O_{t+1}^T = \{o_{t+1}, \dots, o_T\}$ is the observation sequence starting at $t+1$. The sequences α and β are called the forward and backward sequences, respectively, and their definition implies the following

$$L_\lambda(O, s_t = i) = \alpha_t(i)\beta_t(i) \tag{2.24}$$

The overall likelihood of observation sequence O and the probability of observing state i at time t, which will be denoted by $\gamma_t(i)$, can be derived from the α's and β's as follows

$$\sum_{\vec{s}} L_\lambda(O, \vec{s}) = \sum_i \alpha_t(i)\beta_t(i) \qquad \gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_i \alpha_t(i)\beta_t(i)} \tag{2.25}$$

Here the sum ranges over all states i which are possible at time t. For the special cases $t = T$ and $t = 1$, this reduces to the following equations

$$\sum_{\vec{s}} L_\lambda(O, \vec{s}) = \sum_i a_{i,E}\alpha_T(i) \tag{2.26}$$

$$\sum_{\vec{s}} L_\lambda(O, \vec{s}) = \sum_i a_i b(o_1|i)\beta_1(i) \tag{2.27}$$

where $a_{i,E}$ is the transition probability from state i to the exit state E of the HMM. This shows that only one of the sequences α and β is necessary to calculate the likelihood of O and $\vec{s}$. However, since the $\gamma_t(i)$ have to be derived for each time instant t in the training algorithm, efficient calculation of both sequences is important. The key to this is the following relation between the α's and β's of two successive time instants.

$$\alpha_{t+1}(j) = \sum_i \alpha_t(i) a_{i,j} b(o_{t+1}|j) \tag{2.28}$$

$$\beta_t(i) = \sum_j \beta_{t+1}(j) a_{i,j} b(o_{t+1}|j) \tag{2.29}$$

These formulae can now be used to calculate the forward and backward sequences iteratively by the following algorithm.

1. Initialise α by setting $\alpha_1(i) = a_i b(o_1|i)$ and β by setting $\beta_T(i) = 1$.
2. Given $\alpha_t(i)$ and $\beta_{t+1}(j)$ for all i and j calculate $\alpha_{t+1}(j)$ and $\beta_t(i)$ from (2.28) and (2.29), respectively.

This algorithm involves only $2TS$ multiplications, as compared to the brute force approach which needed operations that were of the order of S^T, and constitutes therefore a major reduction in the complexity of the calculation.

2.6.2 Viterbi algorithm

Unlike the forward backward algorithm, the Viterbi algorithm searches for the most likely sequence of states $\vec{s}^{\,*}$, i.e.

$$\vec{s}^{\,*} = \arg\max_{\vec{s}} L_\lambda(O, \vec{s}) \tag{2.30}$$

and hypothesises the word sequence $\vec{w}$ that generated the sequence $\vec{s}^{\,*}$.

The problem of finding $\vec{s}^{\,*}$ in (2.30) can be solved by dynamic programming which can be implemented in the current situation by the following recursive algorithm.

1. Initialise $l_1(j) = b(o_1|j)$ for all permissible states j at frame number one.

2. Once $l_t(j)$ has been calculated for all possible states at time t define $l_{t+1}(i)$ to be

$$l_{t+1}(j) = \max_{i:a_{i,j}\neq 0} l_t(i) a_{i,j} b(o_{t+1}|j) \tag{2.31}$$

   Furthermore define the back pointer for state j at time $t+1$ to be

$$B_{t+1}(j) = \arg\max_{i:a_{i,j}\neq 0} l_t(i) a_{i,j} b(o_{t+1}|j) \tag{2.32}$$

3. The likelihood of the most likely state sequence in (2.30) is $\max_j l_T(j)$, where the maximum is taken over all legal end states j. Following the back pointers for $\arg\max_j l_T(j)$ back to time $t = 1$, one obtains the most likely state sequence $\vec{s}^{\,*}$.

Comparing (2.28) and (2.31) one can see that the Viterbi algorithm simply substitutes the sum in (2.28) by a maximisation over all possible states at time t. This further reduces the amount of computation and makes the Viterbi algorithm therefore a useful tool in the recognition process. The forward-backward algorithm can easily be modified to work with logarithmic likelihoods in which case the products are replaced by sums. One may consult standard texts such as (Deller et al., 1993; Rabiner and Juang, 1993) for the details.

2.7 Parameter Estimation and the EM Algorithm

2.7.1 General Discussion

The problem of parameter estimation consists in finding the model that best explains the training data. This is typically done by maximising a certain optimality criterion such as the likelihood of the training data under the current model. This so-called maximum likelihood (ML) criterion is one of the most common in statistical data modelling and also plays an important role in speech recognition. Due to the special nature of HMM's the problem of maximising the training data likelihood is not tackled directly but via the so-called expectation maximisation algorithm (Dempster et al., 1977). This algorithm takes into consideration that the state sequence $\vec{s}$ of the HMM is not observed in the training phase and maximises an auxiliary function which is based on the expected values of the hidden variable s. This explains the name expectation maximisation because in the first stage of the algorithm the expected values of the hidden states are calculated while in the second stage the model parameters are estimated by maximising the optimality criterion encoded in the auxiliary function. By increasing the value of this function, each iteration of the EM algorithm is also guaranteed to increase the likelihood of the training data. Successive applications of the EM algorithm therefore result in a sequence of models whose fit to the training data increases.

2.7.2 The auxiliary function and its optimisation

In the speech recognition literature (see, for instance, (Baum et al., 1970; Levinson, 1986)), the auxiliary function of the EM algorithm is usually defined by

$$Q(\lambda, \bar{\lambda}) = \sum_{\vec{s}} L_{\lambda}(O, \vec{s}) \log L_{\bar{\lambda}}(O, \vec{s}) \tag{2.33}$$

where λ and $\bar{\lambda}$ are the old and the new sets of model parameters, respectively. This definition differs from the one given in (Dempster et al., 1977) which uses $L_{\lambda}(\vec{s}|O)$ instead of $L(O, \vec{s})$ and is therefore the sum of the likelihoods $L_{\bar{\lambda}}(O, \vec{s})$ for different state sequences $\vec{s}$ weighted by the probability of observing $\vec{s}$. Since both definitions differ only by a normalising factor the two optimisation problems are equivalent. The auxiliary function is also related to the Kullback-Leibler distance which is, for instance, discussed in (Kullback and Leibler, 1951) and (Cover and Thomas, 1991).

The model parameters are usually determined by calculating the partial derivatives of the auxiliary function with respect to the model parameters and finding a set of parameters for which these derivatives are zero. This ensures that the model corresponds to a local extremum or a saddle-point of the auxiliary function. Since one is only interested in local maxima it also has to be checked that the second order derivatives at such a point in the parameter space are negative definite. Such issues have, for instance, been addressed in (Baum et al., 1970) and (Liporace, 1982).

Setting the gradient of the auxiliary function with respect to $\bar{\lambda}$ to zero, for an HMM whose output pdf's are single Gaussians, one can find the following estimation formulae for the different HMM parameters.

$$\bar{a}_{i,j} = \left(\sum_{t=1}^{T} \gamma_{t-1}(i) \right)^{-1} \sum_{t=1}^{T} \alpha_{t-1}(i) a_{i,j} b(o_t|j) \beta_t(j) \tag{2.34}$$

$$\bar{\mu}_i = \left(\sum_{t=1}^{T} \gamma_t(i) \right)^{-1} \sum_{t=1}^{T} \gamma_t(i) o_t \tag{2.35}$$

$$\bar{\Sigma}_i = \left(\sum_{t=1}^{T} \gamma_t(i) \right)^{-1} \sum_{t=1}^{T} \gamma_t(i)(o_t - \bar{m}_i)(o_t - \bar{m}_i)' \tag{2.36}$$

where $i, j \in I$. These parameter estimation formulae also have an intuitive appeal since the values $\gamma_t(i)(\sum_t \gamma_t(i))^{-1}$ can be interpreted as the posterior probability of observing state i at time t given that it was observed anywhere during the sequence O.

The extension of the above parameter estimation formulae to the case of output pdf's that are modelled by GMM's with more than one component is straight-forward. One only has to replace the probability of observing state i by the probability of observing state i and component j. The reestimation formula for the weights w_j^i is then given by

$$\bar{w}_j^i = \left(\sum_{t=1}^{T} \gamma_t(i) \right)^{-1} \sum_{t=1}^{T} \gamma_t(i,j) \tag{2.37}$$

For Gamma output pdf's of a one-dimensional feature d, the reestimation formulae take on a different form. In this case, they are given by

$$\frac{\nu_i}{\eta_i} = \frac{\sum_t \gamma_t(i) d_t}{\sum_t \gamma_t(i)} \tag{2.38}$$

$$\frac{\Gamma'(\nu_i)}{\Gamma(\nu_i)} - \log \eta_i = \frac{\sum_t \gamma_t(i) \log d_t}{\sum_t \gamma_t(i)} \tag{2.39}$$

Unfortunately, these are no closed form reestimation formulae for the parameters ν and η. Denoting the right hand side of (2.38) by m_i and the right hand side of (2.39) by l_i and substituting (2.38) into (2.39) one arrives at

$$\psi(\nu_i) - \log \nu_i = l_i - \log m_i \tag{2.40}$$

where $\psi(\nu) = \Gamma'(\nu)/\Gamma(\nu)$ is the digamma function. Since the digamma function can be calculated efficiently (Abramovitz and Stegun, 1965), the Newton algorithm can be used to find a ν_i such that equation (2.40) is satisfied. Here one iteration of the Newton algorithm is given by

$$\nu_i^{(k+1)} = \nu_i^{(k)} - \frac{\psi(\nu_i^{(k)}) - \log \nu_i^{(k)} + \log m_i - l_i}{\psi'(\nu_i^{(k)}) - \frac{1}{\nu_i^{(k)}}} \tag{2.41}$$

Finally, the ν_i that has been estimated in this way can be substituted into (2.38) to obtain η_i. The complete derivation of the above equations can be found in appendix B.

2.8 Other Optimisation Criteria

In the context of HMM's, maximum likelihood estimation is usually criticised for training an output pdf $b(o|i)$ without reference to the output pdf's of other states. Since the ultimate goal in training models is to find pdf's that focus on distinguishing between different states the ignorance of other output pdf's might result in suboptimal models. To overcome this problem a number of alternatives to the ML parameter estimation for HMM's has been proposed (Bahl et al., 1986; Bahl et al., 1993; Juang and Katagiri, 1992; McDermott and Katagiri, 1994; Rainton and Sagayama, 1992) which address the problem of maximising discrimination between states by optimising other objective functions than the auxiliary function of the EM algorithm. The most commonly used of these is the MMI approach in (Bahl et al., 1986) which has, for instance, been implemented in an LVCSR frame-work in (Valtchev et al., 1997). Here the objective function is given by.

$$\mathcal{F}(\lambda) = \sum_{r=1}^{R} \log \frac{L_\lambda(O_r|\vec{w}_r)P(\vec{w}_r)}{\sum_{\vec{w}\in O_r} L_\lambda(O_r|\vec{w})P(\vec{w})} \tag{2.42}$$

where the O_r are the R training utterances, $\vec{w}_r$ is the word sequence that corresponds to the r-th utterance and $P(\vec{w})$ is the language model probability of word sequence $\vec{w}$. In this equation $\vec{w} \in O_r$ refers to all the word sequences that are possible for observation sequence O_r and $L_\lambda(O_r|\vec{w}_r)P(\vec{w}_r)/\sum_{\vec{w}\in O_r} L_\lambda(O_r|\vec{w})P(\vec{w})$ can therefore be interpreted as the posterior probability that the model λ assigns to the correct word sequence w_r.

It is interesting to note that similar word based posterior probabilities have also been used successfully in recognition (Mangu et al., 1999; Evermann and Woodland, 2000) where they have been shown to improve recognition performance.

Ideally, the posterior probabilities that the model assigns to the different word hypothesis should be one for the correct hypothesis and zero for all the others. The elements on the right hand side of 2.42 can therefore be written as a sum themselves, i.e.

$$\log \frac{L_\lambda(O_r|\vec{w}_r)P(\vec{w}_r)}{\sum_{\vec{w}\in O_r} L_\lambda(O_r|\vec{w})P(\vec{w})} = \sum_{\vec{w}\in O_r} p_{\vec{w}} \log \frac{L_\lambda(O_r|\vec{w})P(\vec{w})}{\sum_{\vec{w}\in O_r} L_\lambda(O_r|\vec{w})P(\vec{w})} \tag{2.43}$$

where $p_{\vec{w}}$ are the ideal posterior probabilities mentioned above. Rewriting equation (2.43) yet again, one arrives at

$$\log \frac{L_\lambda(O_r|\vec{w}_r)P(\vec{w}_r)}{\sum_{\vec{w}\in O_r} L_\lambda(O_r|\vec{w})P(\vec{w})} = -\sum_{\vec{w}\in O_r} p_{\vec{w}} \log \frac{p_{\vec{w}}}{\frac{L_\lambda(O_r|\vec{w})P(\vec{w})}{\sum_{\vec{w}\in O_r} L_\lambda(O_r|\vec{w})P(\vec{w})}} \tag{2.44}$$

The expression on the right hand side of this equation is commonly known as the cross-entropy error between the true posteriors $p_{\vec{w}}$ and the estimated ones. The objective function in (2.42) is therefore the negative of the sum of the cross-entropy errors for all the training utterances and maximising the former is therefore equivalent to minimising the sum of the cross-entropy errors.

This error function will be encountered again in chapter 5 in connection with the so-called softmax functions where it will be derived directly from the EM algorithm. Both the softmax parameter training and the optimisation of equation (2.42) do, unfortunately, not lead to closed form solutions. However, in the case of the softmax functions it is possible to show under certain conditions that there exist unique optimal solutions to the parameter estimation problem. This topic will be further explored in chapter 5 and appendix C.

3

Hidden mode modelling and soft model selection

It is generally accepted that the standard HMM in the last section does not contain sufficient structure to model all the hidden dynamics that can occur in a speech signal. The HMM is, for instance, restricted by the assumption that successive frames are conditionally independent given the state which is clearly not the case for speech. Especially for spontaneous speech which exhibits a higher number of coarticulation effects this assumption contradicts empirical evidence. Another restriction of the standard HMM is that its hidden state s models only temporal segments of phones. While these segments contain the most relevant information in determining the proper word hypothesis, other hidden states of the speech signal, like the level and kind of background noise or the speaking rate of an utterance, are also important, and a failure to model such hidden states can result in significant performance degradation.

To show the effect of spontaneous speech on recognition performance, in (Weintraub et al., 1996) three different types of data sets were collected. For the first data set Weintraub et al. recorded spontaneous conversations on an assigned topic. For the other two data sets the previously recorded conversations were transcribed and the same speakers were asked to read the transcripts once as if they were having a conversation and another time as if they were dictating to a computer. Recognition experiments on these different types of data showed that the recognition performance on spontaneous speech was by far the worst with an error rate of 52.6%. For the read conversational speech the error rate decreased to 37.6% and the read dictation gave the best performance with an error rate of 28.8%. This shows clearly that speech recognition is very sensitive to variation in speaking style. Weintraub et.al. are not the only ones to point out the adverse effect of spontaneous speech on recognition performance. This has also been observed in (Siegler and Stern, 1995; Mirghafori et al., 1995a) and generally in many of the DARPA speech recognition evaluations that were conducted over the last years. Also in connection with the so-called Lombard reflex, which means that speakers change their vocal effort under the influence of noise, the negative effect of variations in speaking style has been observed (Junqua, 1993).

This chapter will discuss some of the models that have either been directly motivated by the problem of hidden states, other than the ones relating to phone segments, or are conceptually related to such models. The extension to the standard HMM in section 3.1 explicitly refers to the speaking rate of the speech signal, while the models discussed in section 3.2 were developed to cope with the presence of background noise or to combine several sources of information pertinent to the speech process. The model in section 3.3, on the other hand, does not directly derive from speech recognition but represents a rather general frame-work in which the output of a number of pattern recognition systems is combined. An application of this model within the HMM frame-work will be also discussed in this section. The model in section 3.4 is , again, an attempt to model the speech process more accurately and it presents a unifying frame-work under which most of the models in the previous sections can be subsumed. Section 3.5, finally, mentions a number of other models that are also related to either the problem of modelling spontaneous speech or to some of the other models discussed previously.

The term “mode”, which is sometimes used exclusively to refer to speaking rate, will be used more casually in this chapter and will refer to any variable in the speech process other than the one expressed by the hidden states of the standard HMM.

3.1 Hidden mode models for HMM’s

One of the first models to explicitly account for the variability in spontaneous speech was proposed in (Ostendorf et al., 1996). Here two sources of information, in addition to the usual observation sequence O, are used to infer the speaking rate at each point in time. The likelihood function of the model has the following form

$$L(O,\vec{s},\vec{q},\vec{v},\vec{w}|C,D) = L(O,\vec{s}|\vec{q},\vec{v})p(\vec{q}|\vec{w},\vec{v})p(\vec{v}|C,D)p(\vec{w}) \tag{3.1}$$

where $\vec{w}$ is the word sequence, $\vec{v}$ is a sequence of states of the hidden mode and $\vec{q}$ is a sequence of pronunciations associated with each word in $\vec{w}$. Finally, C and D are sequences of language and acoustic features, respectively, that are related to the state of the hidden mode. Since the hidden mode is assumed to change its state only at word boundaries, the sequences $\vec{v}$, C and D have the same length as the word and pronunciation sequences $\vec{w}$ and $\vec{q}$. The factorisation of the joint likelihood $L_\lambda(O,\vec{s},\vec{q},\vec{v},\vec{w}|C,D)$ in (3.1) can therefore be realised on a word by word basis, i.e.

$$L(O,\vec{s},q,v,w|c,d) = L(O,\vec{s}|q,v)p(q|w,v)p(v|c,d)p(w) \tag{3.2}$$

In (Ostendorf et al., 1996), the first term on the right hand side of equation (3.2) is modelled by an HMM whose output probabilities are dependent on the state s and the state of the hidden mode v. The second term in (3.2) is modelled by introducing mode dependent pronunciation probabilities. Finally $p(v|c,d)$ has to be a probability distribution over the states of the hidden mode and (Ostendorf et al., 1996) suggests the use of decision trees for this task (Breiman et al., 1984).

The mode dependent pronunciation probabilities in (Ostendorf et al., 1996) are derived for a set of pronunciation transformation rules. This avoids the need to estimate the probabilities for each pronunciations in the dictionary and makes the model more robust. This approach has been further pursued in (Finke and Waibel, 1997). Evaluating their system on the 1996 Switchboard and Callhome database, they initially reported relatively small improvements. However, retraining their models based on their improved alignments gave them an overall relative improvement of about 15%.

In (Ostendorf et al., 1996), no recognition experiments with mode dependent acoustic models are reported. Instead, Ostendorf et al. address the problem of distribution clustering with a phonetic tree that includes questions which are relevant to the state of the hidden mode. These studies are motivated by the fact that modelling the acoustic features dependent on the mode will increase the number of parameters considerably. In order to obtain a robust model it is therefore necessary to apply some kind of parameter sharing and since distributions for different speaking modes should be kept distinct this suggests including features that are related to the hidden mode into the clustering process. The questions investigated in (Ostendorf et al., 1996) are related to the lexical stress of a phone and the position of the phone within the word. The experiments in (Ostendorf et al., 1996) compare the performance of a standard context depending clustering scheme with the performance of a model that is derived from these additional cues. Ostendorf et.al. give an example of such a tree which illustrates that questions about the mode are asked early in the tree which indicates their relevance to the clustering process. The recognition experiments, however, result in marginal improvements.

The method of using higher level context in phonetic decision tree clustering has become known as tagged clustering and has first been introduced in the context of speech synthesis (Donovan, 1996). Later it has also been used in speech recognition (Paul, 1997; Reichl and Chou, 1999; Shafran and Ostendorf, 2000). In (Shafran and Ostendorf, 2000), which builds on the work in (Ostendorf et al., 1996), syllabic information in the decision tree clustering was found to generalise better than wide phonetic context.

Since the sequences C, D and $\vec{w}$ in (3.1) have all the same length, the calculation of the features c and d has to be carried out for each hypothesis $\vec{w}$ separately which limits the application of the model in (3.1) to an n-best rescoring frame-work.

Since the hidden mode in this model is meant to relate to speaking rate, d will be an acoustic feature that contains information about the "velocity" of the sequence of observation vectors O, like the first or second order derivatives, and c might be a feature that models the context of a word w in the word sequence $\vec{w}$ and therefore contains information about the probability of speaking word w fast or slow.

Together, the features c and d provide information about observing a fast or slow speaking rate which is represented by the probability of observing a particular state of the hidden mode, i.e. by the posterior $p(v|c,d)$. Finally, this probability is used in the model (3.2) to weight the pronunciation probability $p(q|w,v)$ and the acoustic likelihood $L(O,\vec{s}|v)$.

Although this possibility is not pursued further, it has been suggested in (Ostendorf et al., 1996) that the training data could be labelled manually with the states of the hidden mode if the size of the task is reasonably small. It is, however, doubtful if this procedure will result in a good model, because although one might be interested in a particular hidden mode, like speaking rate, the use of such a mode might be severely limited by the correlation to the features c and

d. If the correlation of these features to the hidden mode is weak, the posterior $p(v|c,d)$ will always be close to 0.5, in which case the discrimination between mode dependent acoustic or pronunciation models becomes marginal. While one will usually try to avoid such features, one cannot always ensure that c and d have the highest correlation with the hidden mode one had in mind initially. It seems therefore more sensible to allow the model to learn the hidden mode that has the highest correlation with the features c and d, and therefore results in the highest possible discrimination. This point of view is also supported by the experiments in chapter 7.

The reason why (3.1) models only the conditional likelihood $L(O,\vec{s},\vec{q},\vec{v},\vec{w}|C,D)$ rather than the full joint likelihood $L(O,\vec{s},\vec{q},\vec{v},\vec{w},C,D)$ is due to the assumption that the features c and d are only relevant to the distinction between the states of the hidden mode and do not carry any additional information relevant to the sequence of states $\vec{s}$. Modelling the joint likelihood $L(C,D)$ does therefore not add any discrimination to the model and can be omitted. The model in (3.1) is therefore just a factorisation of the joint likelihood $L(O,\vec{s},\vec{q},\vec{m},\vec{w},C,D)$ where non-discriminant information has been ignored. Other factorisations of this joint likelihood are also possible and in addition to (3.1), the following model has been proposed in (Ostendorf et al., 1996)

$$L(O,\vec{s},\vec{q},\vec{v},\vec{w},D|C) = L(O,\vec{s}|\vec{q},\vec{v})p(\vec{q}|\vec{w},\vec{v})p(D|\vec{v})p(\vec{v}|C)p(\vec{w}) \tag{3.3}$$

This can again be regarded as a factorisation of the joint likelihood with the assumption that feature v does not add any discriminant information apart from the one relevant to the state of the hidden mode. In principal, every factorisation of the joint likelihood gives rise to a hidden mode model.

It is not clear a priori which of the many possible factorisations of the joint likelihood of the observation sequence O and the hidden variables is most appropriate. One will have to decide on a case-per-case basis which one to use. The general framework of factorising joint likelihoods will be discussed in section 3.4.

3.2 Subband and Multi-Stream based models

Another way to model additional hidden variables in a speech recognition frame-work has been explored by the group at IDIAP. This work was motivated by the model-data mismatch due to the presence of noise in the speech signal. The idea of this approach is that different subbands of the speech signal are affected by noise in different ways and that it is therefore desirable to find the frequency bands that contain the most reliable information. The correct combination of subbands that will be used for speech recognition can therefore be interpreted as a hidden variable in the recognition process. In these subband models the observation vector o is considered to be made up of sub-observations o_j which correspond to the sub-bands. Subband models which use only the subbands themselves without recombining them, have first been proposed in (Dupont and Bourlard, 1996; Bourlard et al., 1996) and have recently been extended to include the so-called full combination approach in (Hagen et al., 1998; Morris, 1998; Morris et al., 2000). Both methods are usually formulated within the HMM/ANN frame-work and therefore calculate the posterior probabilities of observing a hidden state $s = i$ given a feature vector o rather than the posterior of observing o given $s = i$. The full combination approach calculates the posterior probability $p(s = i|o)$ in the following way

$$p(s = i|o) = \sum_j p(c_j|o)P(s = i|c_j, o) \tag{3.4}$$

Here, c_j represents both a combination of the basic subbands of o and the statement that these subbands are reliable. The posterior probability $P(s = i|c_j, o)$ is therefore usually calculated by using o_{c_j} which is the restriction of the full observation o to the subbands corresponding to c_j. In (3.4) the posteriors $P(s = i|c_j, o)$ are weighted by the posterior probability $p(c_j|o)$ that the subbands in c_j are reliable, given o. The sum over all these weighted posteriors finally yields the overall probability of observing $s = i$ given o. If the posteriors $p(c_j|o)$ are good predictors of the reliability of the subbands c_j, then the decomposition of $p(s = i|o)$ in (3.4) means that only the subbands of o which are useful for recognition will enter the recognition process. While the full combination approach is clearly the correct way to express $p(s = i|o)$ as a sum of the $P(s = i|c_j, o)$ it has the disadvantage that the number of possible combinations of d subbands is 2^d and therefore the number of parameters in (3.4) increases exponentially with the number of subbands. As pointed out in (Hagen et al., 1998), this explosion in the size of the parameter space can be reduced if the subbands are conditionally independent. In this case the posterior $P(s = i|o_{c_j})$ can be derived from the posteriors $P(s = i|o_j)$ where o_j is a single subband, i.e.

$$P(s = i|o_{c_j}) = \frac{P_{kj}}{\sum_{l=1}^{K} P_{lj}} \tag{3.5}$$

$$P_{kj} = \frac{\prod_{l \in c_j} P(s = i|o_l)}{P^{|c_j|-1}(s = i)} \tag{3.6}$$

The posteriors $p(c_j|o)$ in (3.4) are either assumed to be constant or they are modelled with neural networks. In order to train these neural networks a minimum squared error approach is used. This is, however, inconsistent with the EM training paradigm and can furthermore lead to suboptimal solutions due to the nature of error surface of the neural network parameters. An example which illustrates this problem will be given in section 5.3.

Subband models have been shown to be successful in the case of speech that is corrupted by band-limited noise (Morris et al., 2000). For more general noise conditions, however, the performance of the subband models has not surpassed the performance of a standard HMM/ANN system that uses the full observation o.

Subband models are conceptually equivalent to multi-stream models, the only difference being that multi-stream models use observations from different sources as opposed to splitting one source into several subsources. Multi-stream processing has, for instance, been used to combine acoustic and visual information like the movement of the lips (Dupont and Luettin, 1998) to increase recognition performance.

3.3 Mixtures of experts

The models in the last two sections implicitly combined or selected a number of different models to recognise speech. Mixture of experts (ME) models, represent a frame-work which addresses the problem of model selection or combination in the more general context of pattern recognition. The reason for mentioning ME models in this chapter is not only their relationship to the models in the previous sections but also the fact that softmax functions, which will play an important role in chapter 5, have been extensively used in this area.

Mixtures of experts systems, which have first been introduced in (Jacobs et al., 1991), combine the output of several expert networks with a gating network which determines the "reliability" of each expert and regulates its influence on the overall network output. The basic ME concept is visualised in figure 3.1. Here μ_i is the output of the i-th expert and g_i is, according

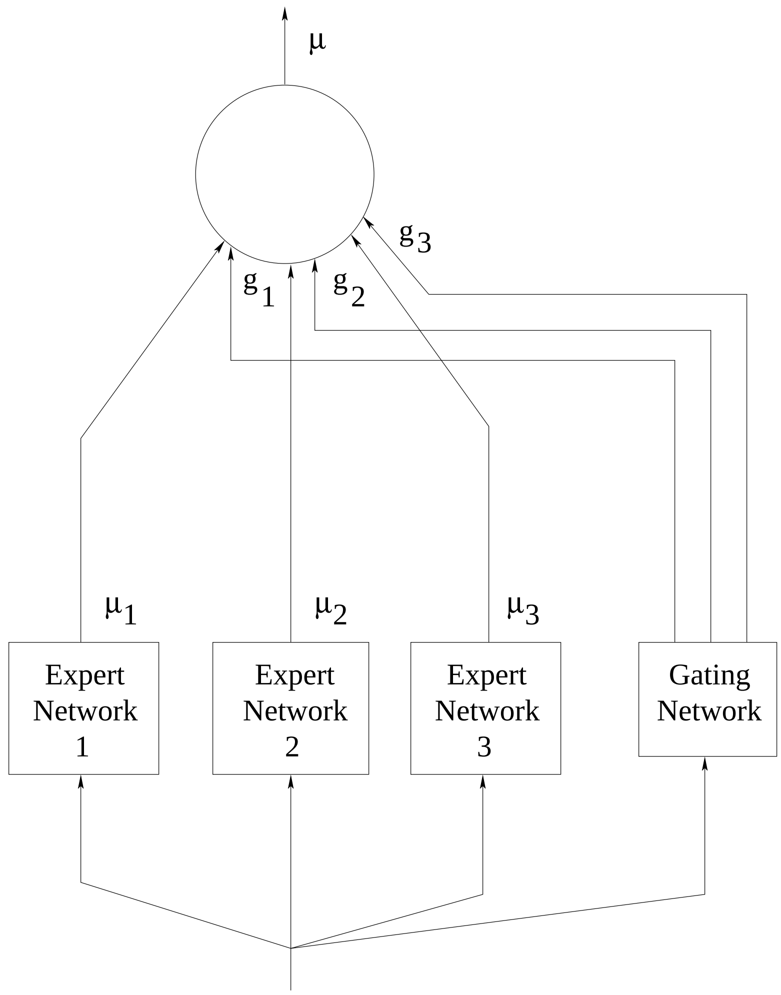


Figure 3.1 *Mixture of experts.*

to the gating network, the "reliability" of expert i given the input. The total output of the ME network is then given by

$$\mu(x) = \sum_j g_j(x)\mu_j(x) \tag{3.7}$$

where x is the system input.

The sub-band and multi-stream models in the last section as well as the hidden mode models in section 3.1 can be regarded as mixtures of experts. In the first case, the probability of the hidden mode $p(v|c,d)$ can be interpreted as a gating function that determines the influence of the mode specific experts $p(q|w,v)$ and $b(o|v,s)$ on the overall system output. In section 3.2 the experts are the models $p(s|c_j,o)$ that are used for recognising the combination of subbands or streams c_j and the probability of c_j being a good source of information $p(c_j|o)$ is the gating function.

In the ME systems in (Jacobs et al., 1991), the outputs of the gating network are calculated with the following formula

$$g_i(x) = \frac{e^{l_i(x)}}{\sum_j e^{l_j(x)}} \tag{3.8}$$

where $l_i(x)$ is the linear combination of the input x received by the output unit i of the gating network. The definition of the $g_i(x)$ (3.8) implies that they are strictly positive, smaller than 1 and that the sum over all the $g_i(x)$'s is equal to 1. The function that transforms the $l_i(x)$ into the gating outputs $g_i(x)$ (3.8) has been called a softmax function (Bridle, 1989), because it implements a "soft" decision boundary as its values range continuously from 0 to 1. Softmax functions also appear in the theory of generalised linear models where they are known as multinomial logit-functions (McCullagh and Nelder, 1989).

The ME models in (Jacobs et al., 1991) were trained by optimising the following error function

$$-\log \sum_x \sum_i g_i(x) e^{-0.5||y(x)-\mu_i(x)||^2} \tag{3.9}$$

where x is a training data point with associated output $y(x)$ and the first sum in (3.9) extends over all training data points x. It is argued in (Jacobs et al., 1991) that optimising this error function will adapt the best fitting expert fastest in the early stages of the training as compared to a simple mean squared error function on the overall network output which, supposedly, adapts the best fitting expert the slowest.

The performance of the ME model in figure 3.1 trained by optimising the error function (3.9) was evaluated in (Jacobs et al., 1991) on a speaker independent four-class vowel discrimination problem (Peterson and Barney, 1952). The experiments showed that the use of the gating network produced experts which focused on one particular decision problem in the classification task. The overall performance of the mixture of experts in (Jacobs et al., 1991) was found to be comparable with that of an MLP, however, the number of training iterations, using a gradient descent approach, was greatly reduced.

The ME architecture has also been combined with HMM's (Bengio and Frasconi, 1995). Here the ME was used to find the relationship between an input sequence and an output sequence, and consequently this type of HMM has been called an input-output HMM (IOHMM). The structure of this model is given in figure 3.2, where d_t and o_t are the input and output of the system at time t. Again, every expert and gating network processes the input d_t. The i-th gating network

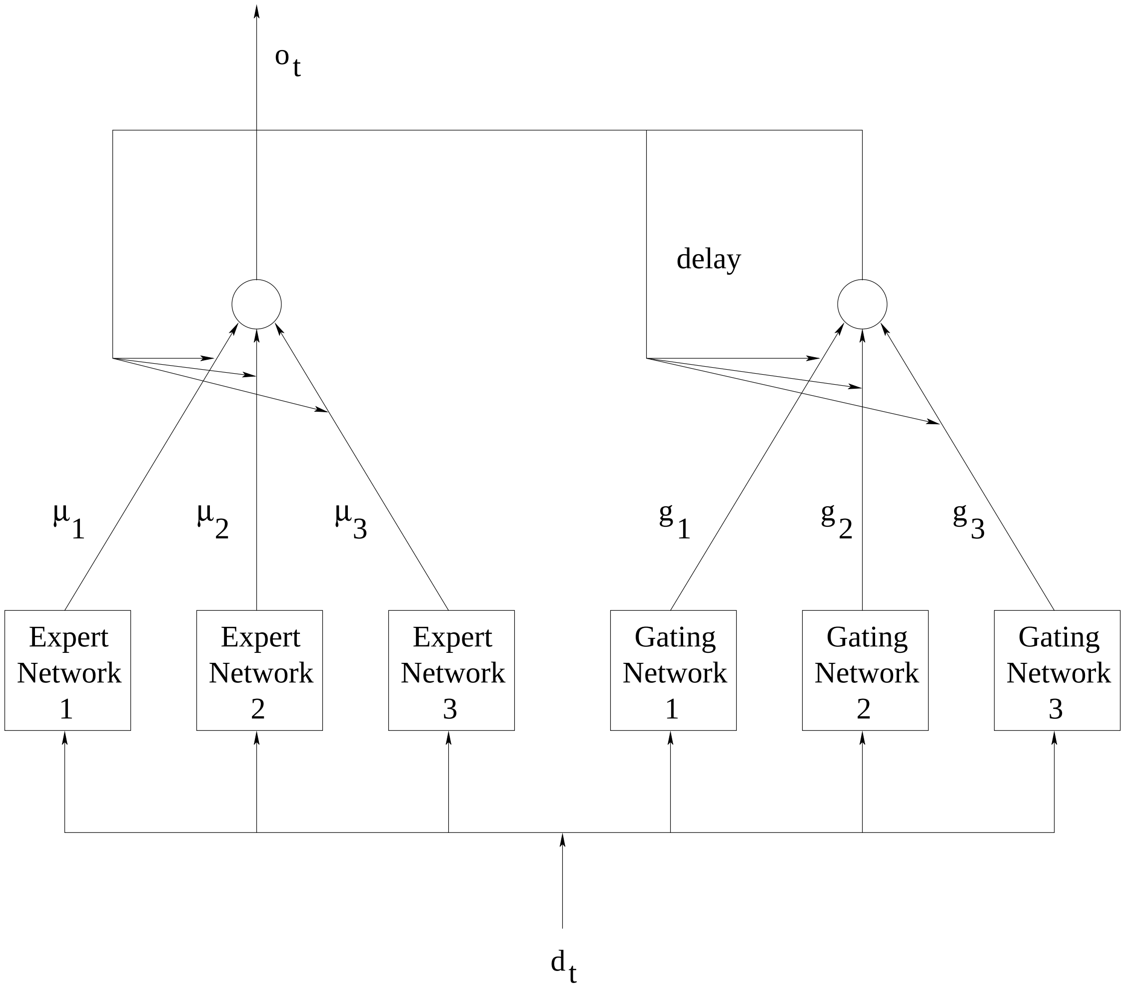


Figure 3.2 *The input-output HMM.*

in figure 3.2, which is modelled by an MLP, is meant to give the probability of observing the hidden states of the HMM, given that the process was in state i at time $t-1$. In contrast to figure 3.1, the g_i in figure 3.2 are therefore probability distributions over all the possible hidden states in the system. Since the sequence of states in the IOHMM is assumed to follow a first order Markov process the gating output needs to be fed back into the gates via the delay loop in figure 3.2. As in a standard HMM, the probability of observing state j at time t is therefore given by

$$\sum_i g_i(j) p_{t-1}(i) \tag{3.10}$$

where $p_{t-1}(i)$ is the probability of observing state i at time $t - 1$ and, applying a slight change in notation, $g_i(j)$ is now the probability of being in state j, given that the process was in state

i in the previous time instant. The IOHMM network can be easily modified to produce output probabilities rather than output vectors. In this case the output pdf $b(y|d_t, s_t)$ is dependent on the system output o_t. Typically, o_t will be used as the mean of the density if $b(y)$ is assumed to be Gaussian. Bengio and Frasconi outlined a training algorithm in (Bengio and Frasconi, 1995) and they tested the IOHMM architecture on the Tomita grammars (Tomita, 1982) which have been used to compare the accuracy of inference methods based on recurrent networks. These results showed improved generalisability as compared to results reported previously (Watrous and Kuhn, 1992). However, in some of the experiments the percentage of models that converged successfully to the correct solution was very small at 10% - 15%. This hints at a problem in the MLP gating networks which do not necessarily converge to the correct solution.

Another application of the ME frame-work to the modelling of time series has been proposed in (Ghahramani and Hinton, 1996). The switching state-space model in this article is also an extension of the HMM frame-work since it makes use of a number of sequences of continuous state vectors in addition to the discrete state variable of the HMM. Each sequence of continuous state vectors is processed by an expert network and the state of the discrete variable is used to switch between these outputs, hence the name.

Both the IOHMM and the switching state space model replace the optimisation of the rather unconventional error function (3.9) with the EM algorithm which can be applied in this case since both models contain hidden variables.

The gating network in the initial concept of a mixture of experts lacked the ability to model non-linear decision boundaries, because each output node was linearly connected to the input nodes. To overcome this restriction, various extensions to the original model have been proposed. As in (Bengio and Frasconi, 1995; Ghahramani and Hinton, 1996), Nowlan and Hinton (Nowlan and Hinton, 1991) replaced the linear combination of the system inputs for a node in the gating network with the output of an MLP. While this increases the number of possible decision boundaries it results in a less satisfactory training algorithm due to the existence of local minima in the MLP error surface.

Another extension to the initial gating concept was proposed in (Jordan and Jacobs, 1992). Here each node in the network was itself the output of the nodes connected to it from the previous layer weighted by the output of another gating network. This architecture is depicted in figure 3.3. The overall output of this network is therefore given by

$$\mu(x) = \sum_i g_i(x) \sum_j g_{ij}(x) \mu_{ij}(x) \tag{3.11}$$

where x is, again, the system input.

Networks of the type in figure 3.3 are the building blocks of the hierarchical mixtures of experts (HME's), where the overall output μ can again be fed into a hierarchical mixture of experts. In general, the branching factor does not have to be two and arbitrary connections between nodes in successive layers are possible.

Just as for an MLP, adding a gating network in a hidden layer increases the number of possible decision boundaries by moving from a linear decision boundary with a single layer to convex

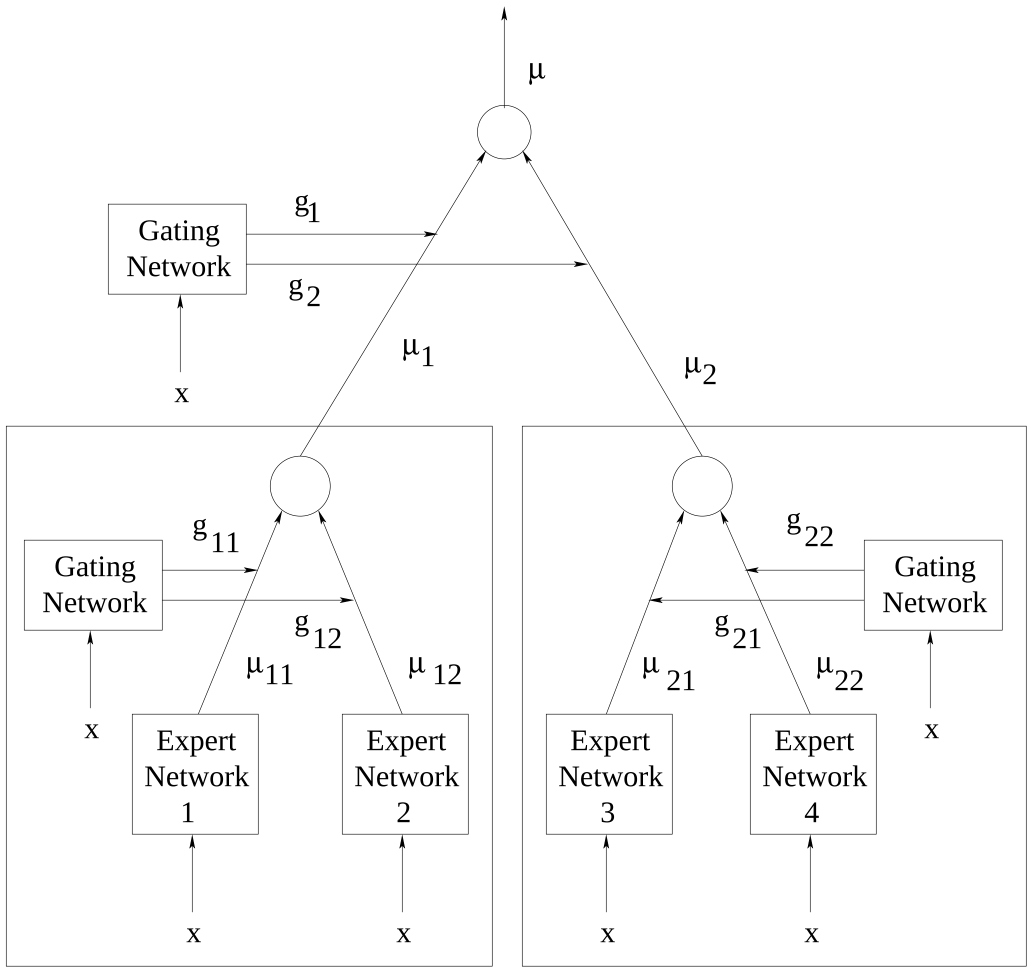


Figure 3.3 *Hierarchical mixture of experts.*

decision boundaries with one hidden layer, and so on. Although as in (Nowlan and Hinton, 1991), this architecture allows for arbitrarily complicated decision boundaries, it again suffers from the presence of local minima in the error surface (Waterhouse and Robinson, 1994). It will later be shown that it is not necessary to sacrifice global convergence in order to model non-linear decision boundaries. Using the concept of polynomial softmax functions, which moves the non-linear components of the model from the ME outputs to the ME inputs, it will be demonstrated in section 5 and appendix C that both the modelling of complex decision boundaries and robust convergence can be achieved at the same time.

The problem of training an HME with the EM algorithm was investigated in (Jordan and Jacobs, 1994; Jordan and Xu, 1995). Here the expert networks in the HME were considered to be single Gaussians whose means are linearly related to the system input. In (Jordan and Xu, 1995) Jordan and Xu derived convergence results for the EM algorithm applied to the training of an HME. The reestimation procedure for the gating networks in this paper is a generalised EM algorithm because instead of maximising the auxiliary function this method only guarantees an increase. This is achieved by applying a single step of the Newton algorithm with a fixed learning rate and is therefore similar to the one step Newton algorithm discussed in section 5.4. The first theorem in (Jordan and Xu, 1995) states that the difference between parameter sets in successive EM iterations can be derived by applying a positive definite linear transformation to the gradient of the log likelihood. A consequence of this result is that the search direction of the EM algorithm has a positive projection onto the gradient of the log probability of the training data which exhibits the relationship between EM and a gradient ascent approach. The second theorem in (Jordan and Xu, 1995) gives an upper bound on the difference between a local maximum of the log likelihood and the log likelihood of a parameter set that has been trained with one EM iteration, given the difference between the log likelihood of the local maximum and the untrained parameter set. These results involve a scaling factor which can attain arbitrarily high values and the result in (Jordan and Xu, 1995) is therefore of limited use from a practical point of view. However, it is worth pointing out, that their proof contains some remarks about the Hessian of softmax functions which will later be commented in appendix C.

It should be noted, that the results in (Jordan and Xu, 1995) only concern the convergence to local rather than global maxima of the log likelihood surface. This restriction is necessary because of the output pdf of the ME model for which the log likelihood surface can have multiple local maxima and is, in general, unbounded. Focusing exclusively on softmax functions rather than on the full ME model it will be shown in appendix C that convergence to a global optimum can be achieved in most practical situations. The convergence results in (Jordan and Xu, 1995), were later also formulated for Gaussian mixture models in (Xu and Jordan, 1996).

Hierarchical mixture of experts have also been applied in speech recognition in the context of HMM/ANN models (Zhao et al., 1995; Waterhouse, 1997). In (Zhao et al., 1995), Zhao et.al. use an HME in a similar manner as a phonetic decision tree. Here the HME implements a soft partition of the input space and for each region that is thus derived an expert is trained that calculates the probabilities of observing the hidden states of the HMM for features that lie in this region. The concept of a probabilistic decision tree using HME's was also discussed in (Jordan et al., 1997). Zhao et.al. report improvements on the Wall Street Journal database when combining their HME system with a conventional HMM. To improve robustness, in (Waterhouse, 1997), Waterhouse extended the HME training algorithm by combining it with the evidence frame-work (MacKay, 1992) and conducts recognition experiments on the Wall Street Journal database comparing HME's to recurrent networks (Robinson, 1994) and MLP's in the ABBOT speech recognition system. Hinton (Hinton and Brown, 2001) recently employed HME's to learn a number of small models that specialised in detecting a single feature in the speech signal.

3.4 Dynamic Bayesian Networks

The models discussed in this chapter so far exhibit a great deal of overlap and it is therefore not surprising that they can all be derived from a single unifying frame-work. This frame-work is the theory of Bayesian networks, also called belief networks, which has been applied to the modelling of time varying processes giving rise to the concept of the dynamic Bayesian network.

As already pointed out earlier, the factorisation of the joint probability distributions in section

3.1 is just one of many possibilities. Dynamic Bayesian networks which were developed in (Dean and Kanazawa, 1988; Zweig, 1998; Zweig and Russell, 1998a; Zweig and Russell, 1998b; Stephenson et al., 2000) discuss the factorisation of time dependent processes from a general point of view.

The motivation for studying Bayesian networks is that modelling a joint distribution of variables is often not feasible both in terms of the size of the resulting models and the time and training data that is needed to estimate such a model. It is therefore desirable to restrict the model to the dependencies between the variables that are truly relevant. Consider, for instance, the problem of modelling the joint probability of the following variables.

variable a the driver of a car is short sighted

variable b the driver of a car is drunk

variable c the driver of a car causes an accident

In this example, one can safely assume that variables a and b are independent and therefore the following holds

$$p(a, b) = p(a)p(b) \tag{3.12}$$

Consequently, the joint probability $p(a, b, c)$ can be factorised as follows

$$p(a, b, c) = p(c|a, b)p(a)p(b) \tag{3.13}$$

Assuming that the variables a and b can take three values, only 6 probabilities have to be estimated in (3.12) as compared to 9 probabilities in the unfactorised joint probability $p(a, b)$. This decrease in model complexity also speeds up the calculation of the probabilities which proves the points mentioned above.

Bayesian networks express the dependency between a set of variables x_i in the form of a directed graph, which for the previous example is given in figure 3.4 (a) at the end of this chapter. As can be seen from this figure, the variables are represented by nodes and the dependency of variable c on a and b is expressed by arrows which lead from b and c to a. In its most general form a Bayesian network can be any directed graph and the parents of a node are the variables that this node depends on. The probability of observing x_i given all the other variables in the model can therefore be written as follows

$$p(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = p(x_i|\text{Parents}(x_i)) \tag{3.14}$$

Using this convention, the dependencies between the variables in an HMM can therefore be realised by the graph in figure 3.4 (b). This figure also shows the dependencies between the variables in the IOHMM model 3.4 (c) from section 3.3 and the Bayesian network corresponding to the factorisation of the likelihood function in section 3.1, figure 3.4 (d). Because of their ability to model dependencies between model parameters on a very general level, Bayesian networks have also been used to overcome the independence assumption that is inherent in standard HMM's (Gravier et al., 1998; Bilmes, 1998; Bilmes, 1999).

If the graph of a Bayesian network has a tree structure the algorithms for inference are very similar to the forward-backward algorithm of HMM's. If the Bayesian network is not tree-structured, however, inference is NP-hard (Cooper, 1990; Dagum and Luby, 1993). This problem can be alleviated if the same Bayesian network is used repeatedly. In this case, one can transform the Bayesian network, via a change of variables, into an equivalent network (Lauritzen and Spiegelhalter, 1988; Jensen et al., 1990) which is tree-structured and is called a clique tree. This preprocessing step needs to be done only once and will therefore result in a more efficient model.

A dynamic Bayesian network is a special type of Bayesian network which models a time varying process. Here a Bayesian network is replicated for each time slice t and connections between the nodes in different time slices are added. If the first-order Markov property is satisfied, such a network can be completely described by two successive time slices and the connections between the variables in each of their Bayesian networks. The addition of connections between variables in different time slices will usually result in a non-tree structure of the generated network, even if the Bayesian networks in each time slice are tree-structured. In order to make the application of general Bayesian networks feasible for speech recognition, it is therefore necessary to apply the preprocessing step that was mentioned earlier. However, since utterances of different length result in different dynamic Bayesian networks their reusability is severely limited. This problem can be overcome to a certain extent by the splicing method which is introduced in (Zweig, 1998), but the NP complexity of inference in a general DBN remains at the very heart of the model.

Dynamic Bayesian networks have, for instance, been used in a speech recognition system to model an articulatory variable in the speech process (Stephenson et al., 2000) which is treated as an additional hidden variable.

3.5 Other related work

This section will review some of the approaches which are not directly related to the models in the last sections but which have also been explored in an attempt to deal with hidden dynamics in speech.

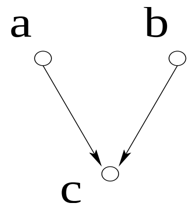
The models in the previous sections all tried to include information about hidden dynamic changes into the recognition process by choosing implicitly between a number of expert models. Another model which implements this soft model selection concept has been proposed in (Iyer et al., 1998). In this paper each HMM in the acoustic model set has two separate branches, one for slow speech and one for fast speech which intersect only at the start and end nodes of the HMM. This ensures, when using a Viterbi decoder, that only one of the speaking rate dependent models is chosen for the complete duration of the phone. The model in (Iyer et al., 1998) can be linked to the theory of trajectory models which has been developed in (Siu et al., 1998; Gish and Ng, 1993; Gish and Ng, 1996) and has been shown to give moderate improvements if states belonging to different branches are shared. The main difference between this type of model selection and the ones discussed in the previous section is the absence of a gating network. In

(Iyer et al., 1998), the decision which model is to be chosen is entirely dependent on the match between the features and the expert models themselves.

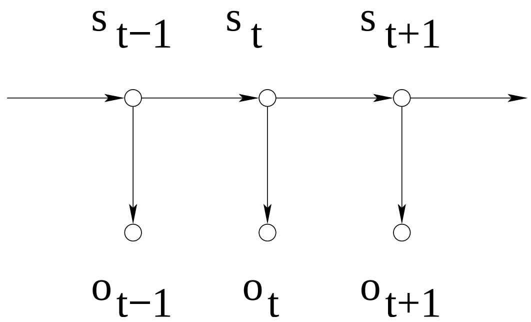
An extension to the model in (Iyer et al., 1998) has been proposed in (Tuerk and Young, 1999), where output pdf's were associated with the transitions between states in an effort to implicitly rescore the hypothesis derived with the model in (Iyer et al., 1998). Although the overall improvement was marginal a positive effect on spontaneous speech could be observed.

In contrast to the models, which implement an implicit, soft model selection, other schemes have been proposed which perform model selection explicitly by choosing what is believed to be the most appropriate model at each point in time for recognition. If speaking rate is the hidden dynamic of interest, this is typically done by using a speaking rate measure on which the selection of the model is based. If the measure indicates a high speaking rate, a fast model is chosen and a slow model is selected if the speaking rate measure is low. The changes to the recognition setup in response to the speaking rates usually affect both acoustic models and pronunciation probabilities. In (Zheng et al., 2000) and (Mirghafori et al., 1995b; Mirghafori et al., 1996), for instance, acoustic models are trained for fast and slow speech and in (Mirghafori et al., 1995b; Mirghafori et al., 1996) the exit transitions in the states of the HMM's are increased for fast speech. Zheng et.al., on the other hand, introduce skip topologies in (Zheng et al., 2000) for fast speech models which allows a phoneme to be skipped completely in the transcription of a word. In (Liu et al., 1998), Liu et.al. also investigate skip topologies and create special acoustic models for non-speech events like laughter. In addition, they use compound words, like “gonna” and “wanna”, in their recognition dictionary. In (Pfau and Ruske, 1998a), Pfau and Ruske address the problem of data sparsity in the training of speaking rate dependent models. This is motivated by the fact that training fast and slow acoustic models results in training sets for each model that are half the size of the complete training set and the speaking rate dependent models therefore do not always generalise well. Pfau and Ruske investigate different training schemes, i.e. maximum likelihood and maximum a posteriori estimation in addition to vocal tract length normalisation (Eide and Gish, 1996; Zhan and Westphal, 1997; Lee and Rose, 1996). They report improvements over their baseline systems for all training methods, VTLN giving the best improvements followed by MAP and ML.

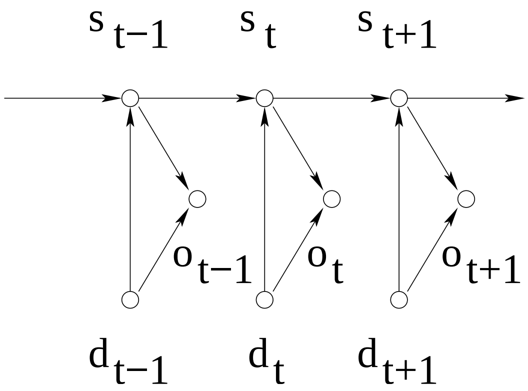
Another model that attempts to capture the dynamics of conversational speech has been proposed in (Bridle et al., 1998; Richards and Bridle, 1999; Picone et al., 1999). This model, which has been called a hidden dynamic model (HDM), is rather different from the ones mentioned so far, in that it neither implicitly nor explicitly performs model selection. Instead the HDM attempts to model the speech production process by smoothing a sequence of target vectors which are representative of the phonetic units in a hidden state space. This results in a trajectory which is then mapped into the acoustic signal space with an MLP. This approach is related to the work in (Blackburn and Young, 1995) which also included information about the speech production process into the recognition model. Since the HDM assumes constant target vectors for the duration of the phonetic units this model is also related to segmental models as discussed in (Gales and Young, 1993a; Gales and Young, 1993b).



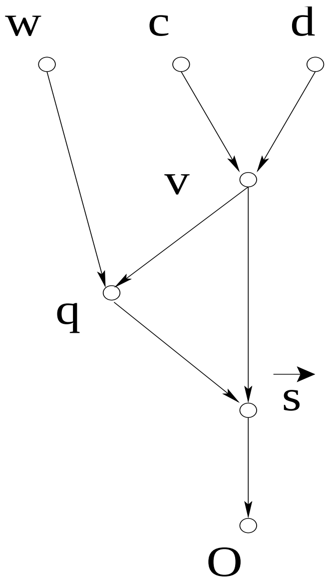
(a) Bayesian network of the factorisation in (3.13).



(b) The DBN structure of an HMM.



(c) The DBN structure of an IOHMM.



(d) The factorisation of the hidden mode model in (3.2) as a Bayesian network.

Figure 3.4 *Examples of Bayesian and dynamic Bayesian networks.*

4

Speaking rate measures

Some of the models discussed in the previous section made use of a second feature, in addition to the standard observations o, which was used to infer the state of the hidden mode. This chapter will review some of the features that have been proposed in connection with spontaneous speech. Since the main task, in this case, is to distinguish between fast and slow speech, these features are measures of speaking rate.

Section 4.1 discusses various measures that can be derived if a transcription of the speech signal is available. Section 4.2, on the other hand, introduces measures which can be directly calculated from the acoustic signal.

4.1 Transcription based measures

There are a number of speaking rate measures which can be derived from a transcription of a speech signal containing start and end times of words or phones. Most of these measures count certain phonetic units and compare them to the length of the segment in which they occurred. This relationship is usually expressed by the ratio between the number of phonetic units n and the duration of the segment, i.e.

$$\frac{n}{\text{duration of segment}} \tag{4.1}$$

If the duration is measured in seconds this gives the number of phonetic units per second. The phonetic units most commonly used in the literature are phones, syllables and words. Since the number of phones that occur in a syllable is limited as is the number of phones and syllables that occur in a word, it is clear that these three speaking rate measures are all correlated to a certain extent (Pfitzinger, 1998).

The measure in (4.1) not only depends on the type of phonetic units but also on the type of the segments. The segments usually correspond to the complete speech material of a single speaker, an utterances, words or phones. Some techniques also use windows around each time instant to define a concept of local speaking rate (Pfitzinger, 1996; Pfitzinger, 1998).

In (Mirghafori et al., 1996) Mirghafori et.al. proposed an ROS measure which differs somewhat from the definition in (4.1). Here the number of phones per second was replaced by a measure of the average phone based speaking rate over the segment. This has been called the mean rate (MR) measure and has been defined by

$$\frac{\sum_i \text{rate}_i}{n} \tag{4.2}$$

where the rate of each phone is defined by

$$\text{rate}_i = \frac{1}{\text{duration}_i} \tag{4.3}$$

In order to measure the speaking rate of a word, in (Zheng et al., 2000) a relative measure is proposed which relates the duration of a word to the durational statistics of the word which are derived from some training set. This measure is defined by

$$R_w(D) = \sum_{d=D+1}^{\infty} p_w(d) \tag{4.4}$$

where $p_w(d)$ is the probability that word w has a duration of d frames. The number $R_w(D)$ is therefore the probability that word w has a duration of more than D frames. Since the word based statistics $p_w(d)$ are difficult to obtain, Zheng et.al. suggest the use of the durational statistics of the phones that make up word w. Under the assumption that the durational statistics of these phones are independent the durational statistics of word w are then given by the convolution of the statistics of the phones. In (Zheng et al., 2000) this measure is used to decide which of a number of speaking rate dependent models should be chosen for recognition.

While all the above measures are equally sensible ROS measures their ultimate usefulness has to be determined by either applying them in a speech recogniser or by comparing them with what an individual perceives as speaking rate. In (Pfitzinger, 1998), Pfitzinger reports experiments that were performed to determine the relationship between perceptive speaking rate and local phone and syllable rate. For this purpose, test subjects were asked to rank a number of test stimuli according to their perceived speaking rate. The results showed that local syllable rate has a higher correlation with perceived speaking rate than does the phone rate. Combining both ROS measures linearly gave the best correlation.

One of the disadvantages of phone rate as an ROS measure is that it does not take the overall structure of the segment for which it is calculated into consideration. Sentences, for instance, which contain many unstressed vowels and unvoiced stops will have a much higher phone rate than sentences with many stressed vowels and fricatives, since the latter have on average a higher duration. To remove this dependency of the ROS measure on the surrounding context measures have been proposed that compare the observed phone duration to the phone duration predicted by a model (Anastasakos et al., 1995; Monkowski et al., 1995; Suaudeau and Andre-Obrecht, 1994; Jones and Woodland, 1993). Here parametric models, like Gamma or Gaussian densities can be used to describe the durational statistics of individual phones (Burhstein, 1995; Levinson, 1986).

4.2 Signal based measures

Since the purpose of speech recognition is to recognise the correct sequence of words in the absence of a manual transcription, the measures discussed in the last section cannot be directly used in a realistic speech recognition task. In (Mirghafori et al., 1996) it was suggested that an intermediate recognition pass could be used to generate a rough transcription of the speech signal from which the speaking rate could be inferred with the methods in the last section. However, it has been pointed out in the same paper that this approach has two disadvantages. The first is the high computational cost of a recognition pass and the second is that the recogniser output is not correct which is especially true for segments containing spontaneous speech. The speaking rate estimate for such segments will therefore be less reliable. The experiments in (Mirghafori et al., 1996) evaluated the correlation between phone rate and the mean of rates measure for manual and automatic transcription and found that phone rates have a substantially higher correlation. This implies that phone rate is less susceptible to recognition errors than the MR measure.

In order to avoid having to use a recogniser just for calculating the speaking rate, the group at ICSI later proposed the enrate measure (Morgan et al., 1997). This measure, whose name refers to energy rate, attempts to quantify the temporal variation of the energy envelope of the signal. The rational behind this approach is that regions where the signal has high amplitude occur more frequently in fast speech and the rate of the change in the spectral envelope should therefore be related to speaking rate.

The enrate measure is derived by first half-wave rectifying the signal, i.e. setting the parts where the signal amplitude is negative to zero and the subsequent application of a low-pass filter. This results in an approximation to the energy envelope which can be substantially downsampled due to its high degree of smoothness. Now the short term spectrum of this approximation is calculated and the first moment of the spectrum is determined which, finally, gives the enrate measure. If the value of enrate is high, this means that the short term spectrum of the energy envelope is concentrated at high frequencies and changes in the shape of the envelope therefore occur more often.

Although the correlation with phone or syllable rate from manually transcribed data has been reported to be relatively weak in (Morgan et al., 1997), recognition experiments on the WallStreet Journal task (Pallett et al., 1994) revealed small performance improvements due to the use of fast and slow models depending on the value of the enrate measure.

To improve correlation with manually determined phone rates, in (Morgan and Fossler-Lussier, 1998) the enrate measure was combined with other measures like peak counting on the original and a filter bank derived version of the signal, to produce the mrate measure. This resulted in an increase in the correlation coefficient from 0.41, which was previously reported for the enrate measure, to 0.67 for the mrate measure. In (Morgan and Fossler-Lussier, 1998) also classification experiments where performed on fast, medium and slow instances of phones which evaluated the usefulness of mrate as a discriminative feature. These experiments are similar to the ones carried out in (Tuerk and Young, 1999) and the experiments described in chapter

6.

In contrast to (Morgan et al., 1997) and (Morgan and Fossler-Lussier, 1998), the methods developed in (Verhasselt and Martens, 1996; Pfitzinger et al., 1996; Pfau and Ruske, 1998b) calculate an ROS measure via an estimate of the number of phonetic units present in a given segment. In (Verhasselt and Martens, 1996) an MLP is investigated which is trained to recognise phone boundaries. The output of this MLP is the probability for observing a boundary at a certain point in time. Summing over all the probabilities for all the frames therefore gives a probabilistic measure of the phone rate. This measure has also been applied in a number of recognition experiments where it was used to determine whether fast, medium and slow acoustic and duration models should be used to recognise an utterance. The experiments reported in (Verhasselt and Martens, 1996) showed small improvements in phone error rate on the TIMIT corpus and furthermore that the gains based on the true phone rate and the estimated phone rate were almost identical.

In (Pfau and Ruske, 1998b) an ROS measure is calculated via an estimate of the number of vowels in a segment. This estimate is derived from the modified loudness which is the half wave rectified difference between the loudness in low frequency bands (Bark 3 to Bark 15) and the loudness in high frequency bands (Bark 20 to Bark 22). The reason for using the modified loudness is that the energy of vowels is mainly concentrated in low frequency bands as opposed to consonants whose energy is concentrated in high frequency bands. For vowels the modified loudness is therefore positive and its maxima occur in the vowel centres. Low-pass filtering the modified loudness ensures a one-to-one relationship between vowels and maxima and counting the number of peaks in the smoothed modified loudness therefore gives an estimate of the number of vowels present in a segment. Dividing this number by the duration of the segment, as in (4.1), one can therefore obtain an estimate of the phone rate of a speech segment. This estimate has been also combined with the zero-crossing rate in (Pfau and Ruske, 1998b) which was reported to result in a performance improvement of the German Verbmobil database.

The method developed in (Pfitzinger et al., 1996) is similar to the one in (Pfau and Ruske, 1998b), although in (Pfitzinger et al., 1996) the speaking rate is calculated from an estimate of the number of syllables rather than from an estimate of the number of vowels. Since, however, the syllable nucleus is usually a vowel these measures ought to be strongly correlated. Although (Pfitzinger et al., 1996) refers to the modified loudness as a good source of information about the location of syllable nuclei, it is not used in the derivation of the estimate of the syllable rate. Instead the signal is passed through a band pass filter in (Pfitzinger et al., 1996) to remove frequencies that do not contribute to the information about the presence of syllables. Then the logarithmic short-term spectrum is calculated and low-pass filtered to increase smoothness. This results in an energy contour whose peaks are considered to be potential syllables. In a post-processing step the syllables are then determined by applying a threshold to the amplitude and averaging over a window to remove multiple peaks that correspond to the same syllable. The accuracy of this syllable estimate was evaluated in (Pfitzinger et al., 1996) by comparing its output to a syllable level transcription which was derived manually.

The signal based ROS measures discussed in this section attempt either directly or indirectly to estimate the phone or syllable rate of a speech segment and therefore suffer from the same problem that was mentioned at the end of section 4.1, namely the lack of contextual information. Since the identity of the phonetic unit has to be known if the expected duration is to be included in the calculation of an ROS measure, this means that relative phone rates cannot be calculated without the use of a recogniser and can therefore not be included into a purely signal based calculation of the ROS measure.

Chapter 6 will discuss a number of features which are also in some sense measures of speaking rate. There, however, a high correlation with phone rate or any other ROS measure derived from a manual transcription will not be an issue. Instead the usefulness of these features will be evaluated in classification experiments which attempt to distinguish between what are assumed to be different states of the hidden mode.

5

The state based Mixture of Experts HMM

Chapter 3 discussed some of the models that were proposed to deal with hidden dynamics in speech signals. Combining some of their features and avoiding some of their inherent disadvantages, this chapter will propose a new model for this task and will discuss its theoretical properties.

After reviewing some of the problems that were encountered in the models in chapter 3, section 5.1 sets out by introducing the basic structure of the state based mixture of experts HMM (SBME-HMM) and will compare the dependencies between its variables to that of similar models. This section will also, more formally, introduce the concept of softmax functions and specify their use in the SBME-HMM.

Section 5.2 will develop the parameter estimation theory of the SBME-HMM by applying the EM algorithm. Reestimation formulae for the different components of the system will be presented and the link between the ones derived for the softmax functions and the cross-entropy error will be investigated. Furthermore, it will be shown that a variant of the forward backward algorithm of the standard HMM can be developed for the SBME-HMM.

Section 5.3 presents a brief digression into the topic of modelling posterior probabilities that depend on some continuous input. Here, neural networks and Gaussian mixture models will be compared to softmax functions and the advantages and disadvantages of the different models will be examined.

Section 5.4 is concerned with details of the implementation of the SBME-HMM parameter estimation and highlights a computationally expensive but exact solution to the softmax parameter estimation problem as well as an approximate but computationally less demanding method.

Section 5.5 develops two different initialisation techniques that are based on mode specific labelling of the training data and the division of the indicating feature space into areas of equal volume.

Finally, section 5.6 discusses the use of the SBME-HMM in recognition and proposes three different model topologies that are appropriate for this task.

5.1 Basics properties of the SBME-HMM

Before developing a new model in this section it is worth remembering some of the problems of the models in chapter 3 which one would like to avoid.

The hidden mode model in section 3.1, for instance, had the disadvantage that the features c and d had to be determined for each word hypothesis $\vec{w}$ separately. While this is unavoidable for a feature like c that is based on the language model, it is less obvious why this should be necessary for an acoustic feature like d. Since the main focus of the model that will be proposed in this chapter is to describe acoustic variability it would be advantageous if the feature d could be calculated directly from the acoustics.

The subband models in section 3.2 proposed a neural network as a model for the posterior probabilities that a certain combination of subbands of the acoustic signal was reliable. Here, the objective of the training algorithm was to minimise the mean squared error (MSE) of the model for the estimated posterior probabilities. This approach has the disadvantage that MSE optimisation is not consistent with the ML frame-work. Furthermore, neural networks that are trained to optimise this criterion can often result in suboptimal solutions because the MSE error surface contains local minima. The other gating networks in chapter 3 were also shown to either exhibit local minima in their error surfaces, as the HME, or to be not general enough to model arbitrary decision boundaries, as the softmax function with linear inputs. To avoid these problems one would like to find a model for the posterior probabilities of the hidden mode that is general enough to allow for complex decision boundaries but which, at the same time, can also be guaranteed to have local minima free error surfaces.

The theory of Dynamic Bayesian networks, as introduced in section 3.4, represented the unifying frame-work for modelling the correlation between a set of variables in a dynamics process. It was mentioned there that in its most general form DBN's suffer from an NP-hard inference problem. In order to yield a tractable reestimation and recognition theory it is mandatory to avoid such high computational cost and a model that is closer to the standard HMM is therefore desirable. It would be especially advantageous if a variant of the forward-backward algorithm could be used to estimate the model likelihoods.

Summarising the previous discussion one arrives at the following constraints that should be satisfied by a new hidden mode model.

- The acoustic features d can be calculated without referring to the word hypothesis $\vec{w}$.
- All the model parameters can be trained within the EM paradigm and in such a way that the maximisation step in the EM algorithm has a unique solution. Furthermore, the models for the mode probabilities should be capable of dealing with arbitrary decision boundaries.
- The model is a natural extension of the HMM frame-work and a variant of the forward-backward algorithm can be used to estimate the model likelihoods.

The first point above can be satisfied by allowing state changes of the hidden mode to occur on a frame-by-frame basis. This removes the necessity to consider the word hypothesis $\vec{w}$ when calculating the features d.

It will be shown in section 5.2 and appendix C that for a special type of gating function the uniqueness of solutions in the parameter optimisation problem can be established under relatively general assumptions. And since the other model components are Gaussian mixture models and transition probabilities this will guarantee that the maximisation step in the EM algorithm has exactly one global maximum.

The third point in the list above can be satisfied by restricting the dependency between the variables of the model to result in a tree-structure of the graph. This guarantees an efficient inference algorithm and therefore an efficient expectation step in the EM algorithm.

Like the model in section 3.1, the model that is proposed here uses, in addition to the ordinary observation sequence O, an observation sequence D which consists of features that are correlated with the states of the hidden mode v. Since the hidden mode can change its state every frame, the sequences O and D have the same length and the features d_t do not have to be derived with regard to the word hypothesis $\vec{w}$. Based on the additional information contained in d_t about the state of the hidden variable v the joint likelihood of the observation sequences O and D and the sequences of hidden variables $\vec{s}$ and $\vec{v}$ will be calculated as follows

$$L(O, D, \vec{s}, \vec{v}) = a_{s_1} b(o_1|s_1, v_1) \prod_{t=2}^{T} a_{s_{t-1}, s_t} b(o_t|s_t, v_t) p(v_t|d_t, s_t) b(d_t|s_t) \tag{5.1}$$

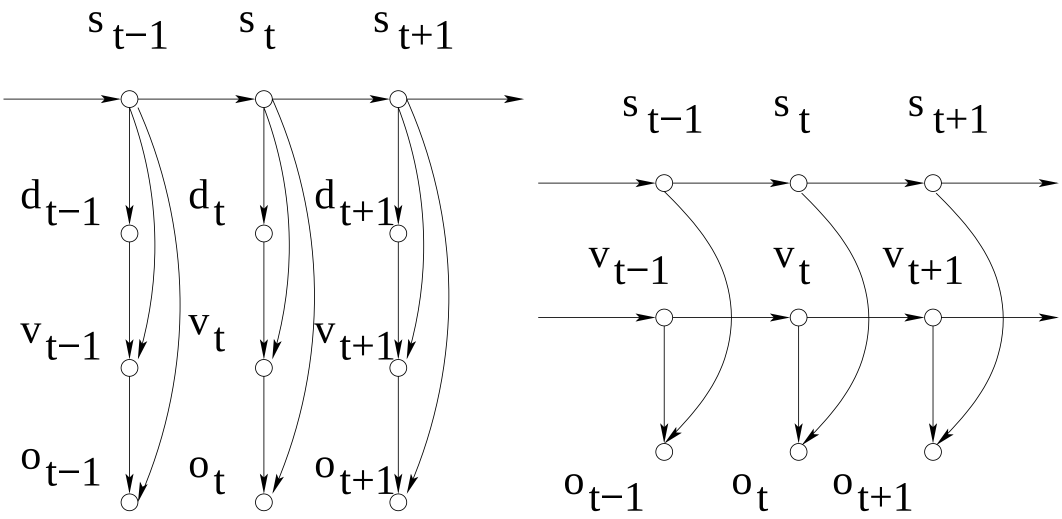
Here, $a_{i,j}$ are the usual transition probabilities for moving from state $s = i$ to state $s = j$ and $b(o|s, v)$ is the output probability of observation o and state s, this time, dependent on the state of the hidden variable v. The posterior probability of observing a particular state of the hidden variable v is denoted by $p(v|d, s)$ and $b(d|s)$ is the output probability of feature d given state s. Since d gives an indication as to which state the hidden variable v is in, this feature will be called an indicating feature.

The model in (5.1) implements a particular factorisation of the joint likelihood of observing o, d and v given s, i.e.

$$b(o, d, v|s) = b(o|s, v) p(v|d, s) b(d|s) \tag{5.2}$$

This factorisation is only valid if $b(o|s, v, d) = b(o|s, v)$ or, in other words, if the dependency of o on d can be completely modelled by the hidden variable v. The hidden mode probability $p(v|d, s)$ can be interpreted as the output of a gating network and (5.2) therefore implements an ME model for each hidden state s. For this reason the model in (5.1) will be called a state based mixture of experts HMM (SBME-HMM).

The dependencies between the different variables in this model are illustrated in the DBN structure in figure 5.1 (a). There is a certain resemblance between this DBN and the one corresponding to an IOHMM as depicted in figure 3.4 (c). This graphic shows, that in an IOHMM the input d_t is used to infer the state of the variable s_t and the hidden state at time t and the input d_t is used to infer o_t. The DBN in figure 5.1 (a), on the other hand, shows that in an SBME-HMM the feature d_t is used to infer the state of v_t and that only v_t is used to infer o_t. The part of the SBME-HMM below the state s_t is therefore like the DBN of an IOHMM where v_t in the SBME-HMM corresponds to s_t in the IOHMM and the connection between adjacent v_t has



(a) DBN for SBME-HMM (b) DBN for factorial HMM

Figure 5.1 *DBN's representing the conditional dependencies in an SBME-HMM and a factorial HMM.*

bee removed as well as the connection between d_t and o_t. These last two variables are assumed to be independent in an SBME-HMM once the state of v_t is known.

The SBME-HMM DBN structure also resembles the DBN of a factorial HMM (Ghahramani and Jordan, 1997; Logan and Moreno, 1998) which is shown, for a special case, in figure 5.1 (b). In general, there can be more than two hidden variables at each time instant t. The main difference in this case is that the v_t are dependent on their predecessor in a factorial HMM and that the dependency of o_t on s_t is modelled directly and not via d_t and the hidden variable v_t. However, the dependency of v_t on its predecessor can, in principle, also be included into the SBME-HMM structure.

Before proceeding any further, it is necessary to answer the question why the particular factorisation of the joint likelihood in (5.2) should be better than any other factorisation. For example, one could also factorise $b(o, d, v|s)$ in the following form

$$b(o, d, v|s) = b(o|s, v)b(d|s, v)p(v|s) \tag{5.3}$$

which has the same modelling power as the factorisation in (5.2). There are two main reasons for choosing (5.2) over (5.3). First, the main information that is contained in the joint likelihood $b(d, v)$ relates to the distinction between the states of the hidden variable v and is therefore the posterior probability $p(v|d, s)$. The output probability $b(d|s)$ on the other hand will add little discriminatory information about the state s since feature o is the prime source of information in this case. It will therefore be possible to share $b(d|s)$ over many states of s and therefore feature d will not give an important contribution in the recognition process. This shows that the main focus rests on the posterior $p(v|d, s)$ and justifies in itself the factorisation in (5.2).

However, one might argue that the hidden mode probabilities can also be estimated by using (5.3) and applying Bayes formula subsequently. And indeed the model in (5.3) has the advantage that it represents just a special case of a standard HMM and therefore all the usual algorithms can be applied to estimate the model parameters. This has, however, the disadvantage that the training paradigm of such a model is not discriminatory in nature and can therefore result in suboptimal solutions for the posterior $p(v|d,s)$. This will be shown in section 5.3.2.

In (5.2), the posterior $p(v|d,s)$ is defined for every possible state of the variable v and is a function of feature d. Since this feature will be continuous in the following, the posterior $p(v|d,s)$ can be modelled by a set of K different functions in the continuous variable d, where K is the number of discrete states of v. In order to satisfy the restrictions of a probability distribution, these functions have to take values between 0 and 1 and their sum has to be 1 for each feature d. Softmax functions, which have already been briefly mentioned in section 3.3, recommend themselves for this task. For a particular state of s and the K states of v they are given by

$$S_{v,s}(d) = \begin{cases} \frac{e^{q_{v,s}(d)}}{1+\sum_{k=1}^{K-1} e^{q_{k,s}(d)}} & : \quad 1 \leq v \leq K-1 \\ \frac{1}{1+\sum_{k=1}^{K-1} e^{q_{k,s}(d)}} & : \quad v = K \end{cases} \tag{5.4}$$

and $p(v|d,s)$ will therefore be modelled by

$$p(v|d,s) = S_{v,s}(d) \tag{5.5}$$

In equation (5.4) the $q_{v,s}(d)$ are linear combinations of functions $\phi^l_{v,s}(d)$, i.e.

$$q_{v,s}(d) = \sum_{l=0}^{L_j} c_l^{v,s} \phi^l_{v,s}(d) \tag{5.6}$$

Furthermore the $c_l^{v,s}$ are independent of each other, and the $\phi^l_{v,s}(d)$ are independent of all the softmax parameters $c_l^{v,s}$. For polynomial exponents the $\phi^l_{v,s}$ are given by

$$\phi^l_{v,s}(d) = d^l \tag{5.7}$$

and the $q_{v,s}(d)$ are therefore

$$q_{v,s}(d) = \sum_{l=0}^{L_j} c_l^{v,s} d^l \tag{5.8}$$

Here $L_j + 1$ is the number of parameters in the polynomial and l can be a multi-index if the dimension of d is greater than one. Since only polynomial exponents $q_{v,s}(d)$ will be considered in this chapter, $\phi^l_{v,s}(d)$ will always be here denoted by d^l. It should, however, be borne in mind that the results developed later can be applied to more general functions $\phi_{v,s}(d)$ as well. The softmax parameters $c_l^{v,s}$ for one particular state $s = i$ will be denoted by

$$\mathbf{c}_i = \{c_l^{v,i} : l = 0, \dots, L_v \qquad v = 1, \dots, K-1\} \tag{5.9}$$

Note that the definition of the softmax functions in (5.4) differs slightly from the definition given in section 3.3. In (5.4) both the nominator and denominator have been divided by one of the exponentials in the sum of the denominator. This is necessary to ensure a unique solution in the parameter estimation problem.

Using (5.4) directly to calculate the softmax functions can result in numerical instability because both the nominator and denominator in (5.4) can have very high values. In order to decrease the influence of numerical errors it is therefore sensible to divide both the denominator and nominator in (5.4) by the nominator, which results in the following set of equations

$$S_{v,s}(d) = \begin{cases} \frac{1}{1+e^{-q_{v,s}(d)} + \sum_{k=1,k\neq v}^{K-1} e^{q_{k,s}(d)-q_{v,s}(d)}} & : \quad 1 \leq v \leq K-1 \\ \frac{1}{1+\sum_{k=1}^{K-1} e^{q_{k,s}(d)}} & : \quad v = K \end{cases} \tag{5.10}$$

It is interesting to note that the softmax functions in (5.4) have a form similar to the hypothesis posteriors in the MMI criterion (2.42) which was briefly introduced in section 2.8. For this reason, the training of the SBME-HMM can also be regarded as a within-state discriminative training. In contrast to (2.42), however, the exponents of the softmax functions depend linearly on the model parameters which, as will be seen later, is an important prerequisite for a satisfactory reestimation theory.

5.2 Parameter Estimation

The first step in the implementation of the EM algorithm is the derivation of the auxiliary function. For the SBME-HMM in the last section this function is given by

$$Q(\lambda, \bar{\lambda}) = \sum_{\vec{s},\vec{v}} L_{\lambda}(O, D, \vec{s}, \vec{v}) \log L_{\bar{\lambda}}(O, D, \vec{s}, \vec{v}) \tag{5.11}$$

Here the sum has to be extended over all possible sequences of states of the hidden variables s and v. In order to maximise the auxiliary function with respect to the model parameters, one has to calculate its partial derivatives and find a set of parameters where they vanish. There are four different types of parameters in an SBME-HMM which are associated with the following objects.

- transitions between states s
- output pdf's $b(o|s,v)$ for feature o
- output pdf's $b(d|s)$ for feature d
- posterior probabilities $p(v|d,s)$ of observing the states of the hidden variable v

In the following, $\omega_{i,k}$ and δ_i will denote an arbitrary parameter of $b(o|i,k)$ and $b(d|i)$, respectively, $a_{i,j}$ and $c_l^{k,i}$ will, as usual, denote transition probabilities and softmax parameters. With this notation the derivatives of the auxiliary function (5.11) are given by

$$\frac{\partial}{\partial a_{i,j}} Q(\lambda, \bar{\lambda}) = \sum_{t=1}^{T-1} L_\lambda(O, D, s_t = i, s_{t+1} = j) \frac{\partial}{\partial a_{i,j}} \log a_{i,j} \tag{5.12}$$

$$\frac{\partial}{\partial \omega_{i,k}} Q(\lambda, \bar{\lambda}) = \sum_{t=1}^{T} L_\lambda(O, D, s_t = i, v_t = k) \frac{\partial}{\partial \omega_{i,k}} \log b(o_t|i,k) \tag{5.13}$$

$$\frac{\partial}{\partial \delta_i} Q(\lambda, \bar{\lambda}) = \sum_{t=1}^{T} L_\lambda(O, D, s_t = i) \frac{\partial}{\partial \delta_i} \log b(d_t|i) \tag{5.14}$$

$$\frac{\partial}{\partial c_l^{k,i}} Q(\lambda, \bar{\lambda}) = \sum_{t=1}^{T} \sum_{k=1}^{K} L_\lambda(O, D, s_t = i, v_t = k) \frac{\partial}{\partial c_l^{k,i}} \log p(k|d_t, i) \tag{5.15}$$

In order to simplify notation, the following definitions will be used

$$\gamma_t(i) = L(O, D, s_t = i) / \sum_i L(O, D, s_t = i) \tag{5.16}$$

$$\gamma_t(i,k) = L(O, D, s_t = i, v_t = k) / \sum_{i,k} L(O, D, s_t = i, s_t = k) \tag{5.17}$$

$$\gamma_t(k|i) = L(O, D, v_t = k|s_t = i) / \sum_k L(O, D, v_t = k|s_t = i) \tag{5.18}$$

The derivatives in (5.12) and (5.14) are the same as for the standard HMM and the reestimation formulae in both cases are therefore identical. Equation (5.13) differs from the corresponding derivative for the standard HMM in that the state likelihood $\gamma_t(i)$ has been replaced by the joint state likelihood $\gamma_t(i,k)$. Exchanging these likelihoods in the reestimation formulae for the standard HMM therefore gives the reestimation formulae for the parameters of the output pdf $b(o|s,v)$. If $b(o|i,k)$ is, for example, modelled by a single Gaussian the mean and variance of this Gaussian are estimated as follows

$$\mu_{i,k} = \frac{\sum_{t=1}^{T} \gamma_t(i,k) o_t}{\sum_{t=1}^{T} \gamma_t(i,k)} \tag{5.19}$$

$$\Sigma_{i,k} = \frac{\sum_{t=1}^{T} \gamma_t(i,k)(\mu_{i,k} - o_t)(\mu_{i,k} - o_t)'}{\sum_{t=1}^{T} \gamma_t(i,k)} \tag{5.20}$$

So far, the derivations of the reestimation procedures have been straight forward. The main complication in the estimation of the model parameters arises due to the existence of the posteriors $p(v|d,s)$. First the derivative in (5.15) will be written in a form which reveals its information theoretic background, i.e.

$$\begin{aligned} \frac{\partial}{\partial c_l^{k,i}} Q(\lambda, \bar{\lambda}) &= \sum_{t=1}^{T} \gamma_t(i) \sum_{k=1}^{K} \gamma_t(k|i) \frac{\partial}{\partial c_l^{k,i}} \log p(k|d_t, i) \\ &= -\sum_{t=1}^{T} \gamma_t(i) \sum_{k=1}^{K} \gamma_t(k|i) \frac{\partial}{\partial c_l^{k,i}} \log \frac{\gamma_t(k|i)}{p(k|d_t, i)} \end{aligned} \tag{5.21}$$

The latter equation holds because $\gamma_t(k|i)$ is independent of $c_l^{k,i}$. Ignoring the derivative $\frac{\partial}{\partial c_l^{k,i}}$, the sum over k in (5.21) is the Kullback-Leibler (KL) distance between the estimated probabilities

$\gamma_t(k|i)$ and the model probabilities $p(k|d_t, i)$. Therefore, the expression in (5.21) is the partial derivative of the negative cross-entropy error function (Bishop, 1995) that is weighted by the state probabilities $\gamma_t(i)$. To emphasis the dependency on the softmax parameters $\mathbf{c}_i$, this error function will be denoted by

$$E(D, \gamma, \mathbf{c}_i) = \sum_{t=1}^{T} \gamma_t(i) \sum_{k=1}^{K} \gamma_t(k|i) \log \frac{\gamma_t(k|i)}{p(k|d_t, i)} \tag{5.22}$$

Due to the minus sign in (5.21), maximising the auxiliary function with respect to the parameters of the posteriors $p(v|d, s)$ amounts to minimising the weighted cross-entropy error (5.22) between the estimated and the model posteriors. Substituting the definition of the softmax functions for the $p(v|d, s)$ and taking the derivative with respect to one of the softmax parameters $c_l^{v,s}$ requires the derivatives of the logarithm of the softmax functions which are given by

$$\begin{aligned} \frac{\partial}{\partial c_l^{v,s}} \log S_{k,s}(d) &= \frac{\partial}{\partial c_l^{v,s}} \left(q_{k,s}(d) - \log \left(1 + \sum_{k=1}^{K-1} e^{q_{k,s}(d)} \right) \right) \\ &= \begin{cases} d^l - S_{k,s}(d) d^l & : \quad v = k \\ \\ -S_{v,s}(d) d^l & : \quad v \neq k \end{cases} \end{aligned} \tag{5.23}$$

Substituting this into equation (5.21) therefore yields

$$\frac{\partial}{\partial c_l^{v,s}} Q(\lambda, \bar{\lambda}) = \sum_{t=1}^{T} \gamma_t(i) \sum_{k=1}^{K} \left(\gamma_t(k|i) - S_{k,s}(d_t) \right) d_t^l \tag{5.24}$$

Since the S_j depend non-linearly on the parameters $c_l^{v,s}$, setting equation (5.24) to zero results in a set of non-linear equations in the $c_l^{v,s}$ from which the softmax parameters cannot be derived in closed form. In order to solve these equations, an iterative scheme like the Newton-Raphson algorithm will have to be employed.

A priori, the solution of the equations $\frac{\partial}{\partial c_l^{v,s}} Q(\lambda, \bar{\lambda}) = 0$ only guarantees that the corresponding set of parameters is associated with a local extremum or a saddle point of the auxiliary function. To ensure that this point is a local maximum, it is necessary to evaluate the second order derivatives of the auxiliary function. This will be done in appendix C where it will also be shown that the Hessian of the cross-entropy error function with respect to the softmax parameters is always a positive definite matrix. This ensures that the cross-entropy error function is convex which is equivalent to the auxiliary function being concave. As an immediate consequence of the concavity of $Q(\lambda, \bar{\lambda})$ one can rule out the existence of suboptimal local extrema and, with a proper training algorithm, the softmax functions will therefore always converge to the correct solution no matter how they are initialised. The relevant statements, asserting the existence of unique solutions, together with their proofs can be found in appendix C.

In order to arrive at a practicable reestimation procedure, the γ's which appeared in this section's formulae will have to be calculated efficiently. Since the following equations hold

$$\gamma_t(i) = \sum_{k=1}^{K} \gamma_t(i,k) \tag{5.25}$$
$$\gamma_t(k|i) = \frac{\gamma_t(i,k)}{\gamma_t(i)} \tag{5.26}$$

it is only necessary to calculate $\gamma_t(i,k)$. This can be done with a modified forward-backward algorithm which takes the existence of the additional hidden variable v into account. Here the forward and backward sequences are defined as follows

$$\alpha_t(i,k) = L(O_1^t, s_t = i, v_t = k) \tag{5.27}$$
$$\beta_t(i,k) = L(O_{t+1}^T | s_t = i, v_t = k) \tag{5.28}$$

and consequently $\gamma_t(i,k)$ is the product of α and β, divided by the sum of these products i.e.

$$\gamma_t(i,k) = \alpha_t(i,k)\beta_t(i,k) / \sum_{i,k} \alpha_t(i,k)\beta_t(i,k) \tag{5.29}$$

The values of the forward and backward sequences can now be recursively calculated via the following expressions

$$\alpha_{t+1}(j,l) = \sum_{i,k} \alpha_t(i,k) a_{i,j} b(o_{t+1}|j,l) p(l|d_{t+1},j) b(d_{t+1}|j) \tag{5.30}$$
$$\beta_{t+1}(j,l) = \sum_{j,l} \beta_{t+1}(j,l) a_{i,j} b(o_{t+1}|j,l) p(l|d_{t+1},j) b(d_{t+1}|j) \tag{5.31}$$

Using this method to derive the forward-backward sequences and hence the joint state likelihoods results in an efficient algorithm which can be carried out with low computational cost.

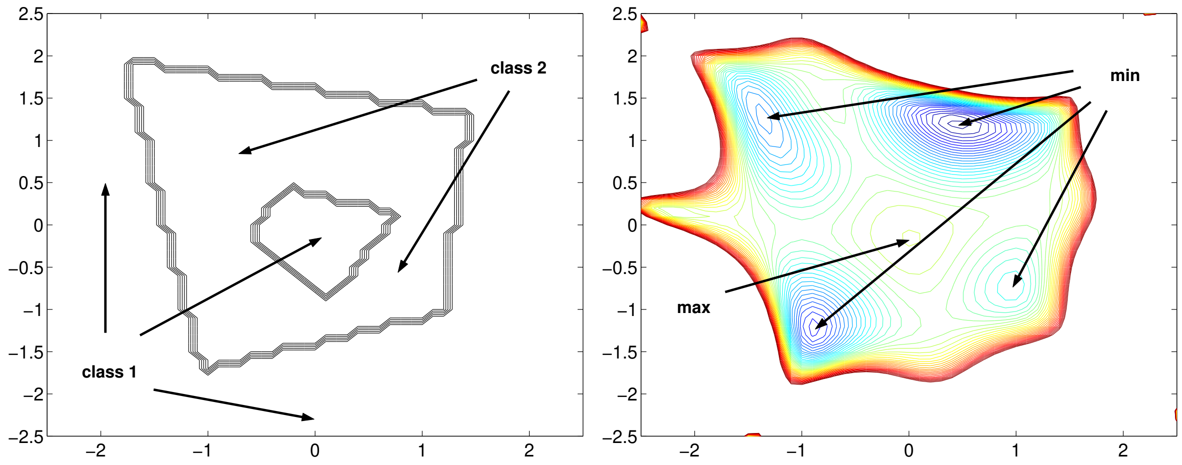
5.3 Excursion: Modelling posterior probabilities

It has already been pointed out that there are a number of ways to model posterior probabilities. This section will try to show why polynomial softmax functions should be chosen for this task by comparing their performance on a number of artificially generated data to that of more commonly used techniques.

5.3.1 Neural networks

Due to their flexible structure, neural networks are a very versatile model. In principle, neural networks can approximate arbitrary functions as long as their topology and activation units are chosen properly. This flexibility can also be exploited to model posterior probabilities of class membership given a continuous variable. Each of the outputs of such a network is then interpreted as the probability of belonging to one of the classes. In order to give a proper probability distribution the network outputs have to sum to one for each input. To satisfy this requirement, the activation units of the output layer are usually chosen to be softmax functions.

Figure 5.2(a) shows an example of a two-class problem which is normally used to motivate the introduction of neural networks with two or more hidden layers. Here, the plane is divided



(a) Two-class problem (b) Estimated 8th degree polynomial

Figure 5.2 *Exponential polynomial of softmax functions trained on two-dimensional two-class problem.*

into three regions. The innermost and outermost region belong to class 1 while the middle region belongs to class 2. The boundaries that separate the regions each consist of 4 non-rectangular straight lines. Training two softmax functions with an eighth degree polynomial on data generated from these two classes gives the polynomial whose contour plot is shown in figure 5.2(b). The contour plot of the corresponding softmax function is visually indistinguishable from the graph in figure 5.2(a) and has therefore been omitted. One can see that the polynomial in figure 5.2(b) results in a good approximation of the class posteriors because one of the contour lines in this figure closely resembles the decision boundary of the classification problem in figure 5.2(a). This contour line corresponds to the set of points where the polynomial vanishes and the value of the softmax functions on these points is therefore 0.5. The local minima and maxima depicted in figure 5.2(b) correspond to values of the softmax functions that are either close to 0 or 1. In this example, the polynomial was initialised, without referring to the training data, as follows

$$q_{init}(d_1, d_2) = d_1 + d_2 \tag{5.32}$$

This initialisation corresponds to a hyperplane which passes through $(0, 0, 0)$ and whose normal vector is $(1, 1, -1)$. As a result, the contour plot of this function consists of a set of parallel straight lines. Compared to the estimated polynomial, the initial guess is therefore far away from the final solution which indicates the great robustness of the polynomial parameter estimation problem.

Figure 5.3, on the other hand, shows the problems that can occur if one chooses a neural network with sigmoids as activation functions to model the class posteriors. In this example a

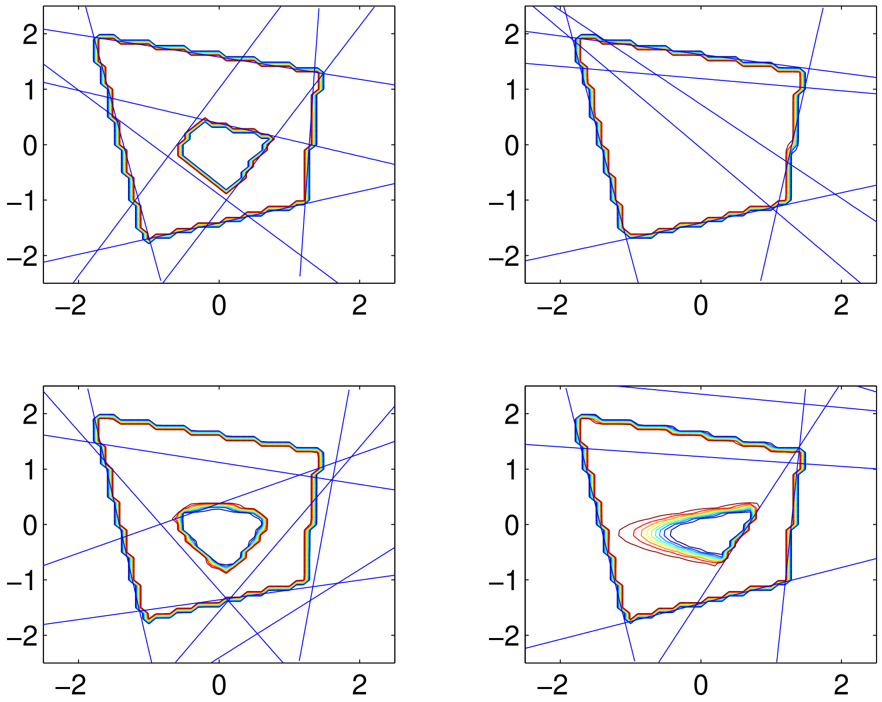


Figure 5.3 *Suboptimal approximations of the class posteriors in figure 5.2(a) by a neural network with two hidden layers. The blue lines correspond to the weights and biases of the input layer while the remaining lines show the approximated decision boundary.*

neural network with 8 units in the first, 2 units in the second and 2 units in the output layer was used to approximate the class posteriors of the classification problem in figure 5.2(a). In addition, each of the network layers had a bias unit and the complete network was trained on the same training data as in the previous example. The network was initialised 100 times using the Nguyen-Widrow initialisation (Nguyen and Widrow, 1990) which is a random initialisation technique that attempts to distribute the network weights evenly over the whole network structure. Figure 5.3 shows some of the solutions that were obtained for this model by optimising the MSE criterion with the Levenberg-Marquardt algorithm (Hagan and Menhaj, 1994). The blue lines that are shown in this figure correspond to the weights that connect the input layer with the first hidden layer. For a perfect approximation these lines should coincide with the decision boundary of the classification problem. One can see from figure 5.3 that this was only achieved in the example in the top left graph of this figure. All other approximations show suboptimal solutions which correspond to local minima of the MSE error function. While the approximation in the bottom left corner is reasonably good, the graphs on the right hand side of figure 5.3 exhibit considerable flaws. For the 100 differently initialised neural networks the model converged in 62 cases to an approximation similar to the ones on the left hand side of figure 5.3, only 5 of which were associated with a global minimum as in the top left graph. Of the remaining models, 18 converged to an approximation similar to the one in the top right graph in figure 5.3 and 12 models converged to a posterior which was constant over the training interval. The remaining 8 models converged to more exotic suboptimal solutions, as in the bottom right corner of figure 5.3, where parts of the outer or inner decision boundary were not modelled properly.

The failure of the neural network to converge to a proper solution might be acceptable in a system which uses a small number of models because in this case the convergence of each model can be checked by manual inspection. For a system with many models, however, the failure to converge poses a serious problem. In a LVCSR system, for instance, where an SBME-HMM will use 1000's of posterior probabilities $p(v|d,s)$ a more reliable training procedure is very important.

5.3.2 Mixture models

As mentioned in section 5.1, the SBME-HMM implements just one of many possible factorisations of the joint output probability $b(o,d,v|s)$. An alternative factorisation was also briefly discussed which was given by

$$b(o,d,v|s) = b(o|s,v)b(d|s,v)p(v|s) \tag{5.33}$$

It was claimed in section 5.1 that the factorisation of the SBME-HMM was superior to (5.33) due to its discriminative nature. This claim will now be substantiated with an example.

Since, for recognition purposes, one is mainly interested in the posteriors $p(v|d,s)$ the quality of a model derived by training (5.33) has to be evaluated by comparing the model posterior to the true posterior. The model posterior $p(v|d,s)$ can be calculated from the components in (5.33) by applying Bayes formula, i.e.

$$p(v|d,s) = \frac{b(d|s,v)p(v|s)}{\sum_v b(d|s,v)p(v|s)} \tag{5.34}$$

To illustrate the two different approaches to estimating the probabilities $p(v|d,s)$, a simple two-dimensional classification problem has been generated in figure 5.4. Here the data points are separated into a red and a blue class and each of the four red and blue circles in the plane was generated by sampling a Gaussian distribution with unit variance. In order to derive the posterior with the model in (5.33) one first has to learn the Gaussian mixtures from which the training data in figure 5.4 were derived.

Figure 5.5 illustrates the fact that training a Gaussian mixture model with the EM algorithm might result in a suboptimal solution. Here the models were initialised by disturbing the overall mean of the training data by a small amount in the d_1 and d_2 directions. The graphs in the upper half of figure 5.5 show that in this case the mixture components converge to Gaussians whose means lie between the centres of the red and blue circles in the plane. The circles themselves are modelled by the intersections of the Gaussians. As a result, the approximating mixture models, which can be seen in the lower half of figure 5.5 are relatively far from the true mixture. After applying Bayes formula to derive the posterior $p(v|d,s)$ the approximation to the true posterior is therefore relatively poor which can be seen from figure 5.6. The graph in the top left corner of figure 5.6 shows the true posterior, while the top right graph shows the approximation derived

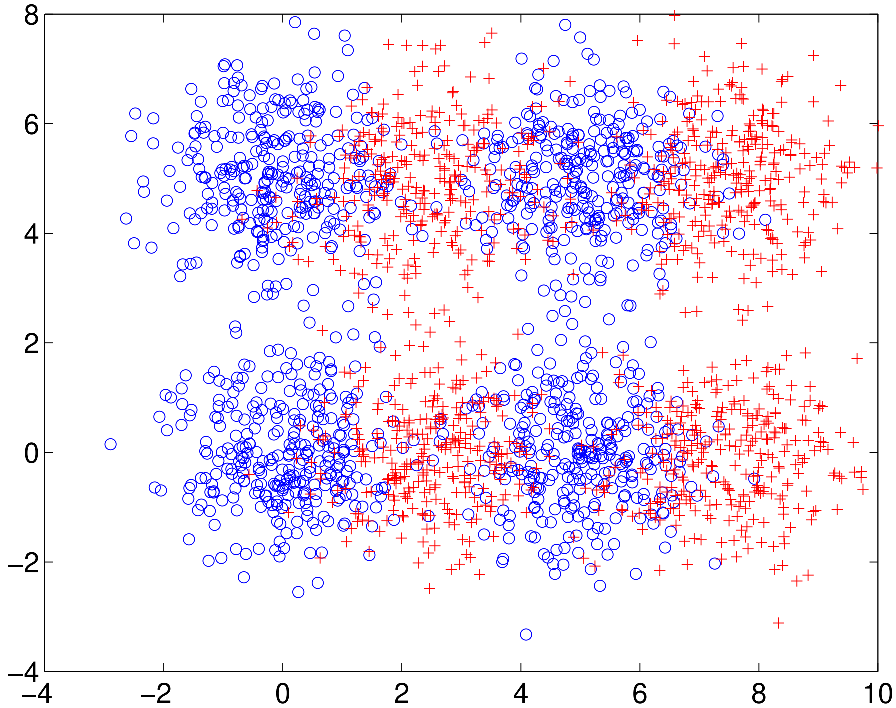


Figure 5.4 *Training data for a two-dimensional two-class problem.*

by training the model in (5.33) with EM and subsequent application of Bayes formula. As a comparison the lower two graphs in this figure show approximations of the posterior that were derived by training softmax functions with polynomials of degree two and three, respectively. The left lower graph shows that already for a polynomial of degree two the approximation is relatively good. For the approximation with a polynomial of degree three, in the bottom right graph in figure 5.6, the distinction between the two classes becomes sharper since the slope of the posterior becomes steeper.

The posterior in the right upper graph in figure 5.6, on the other hand, only distinguishes well between the two classes in areas of the plane were there were little training data. In areas where the training data is concentrated the distinction is relatively blurred which is the result of the artifacts that appear between the centres of the Gaussians. The good quality of the approximation by softmax functions in figure 5.6 also shows that the number of parameters that are necessary to model the posterior properly is much smaller for softmax functions than it is for GMM's. The Gaussian mixture models in this example used 5 parameters for each Gaussian, 2 for the mean and 3 for a full covariance matrix, and the total number of parameters in this model was therefore 40. The softmax model with a polynomial of degree two, on the other hand, used only 6 parameters and the softmax functions with a polynomial of degree 3 used 10 parameters.

The suboptimal solution for the GMM encountered in this example illustrates the same problem that is inherent in the training of neural networks – the existence of local minima in the error surface. This phenomenon is not restricted to Gaussian mixture models but will occur in

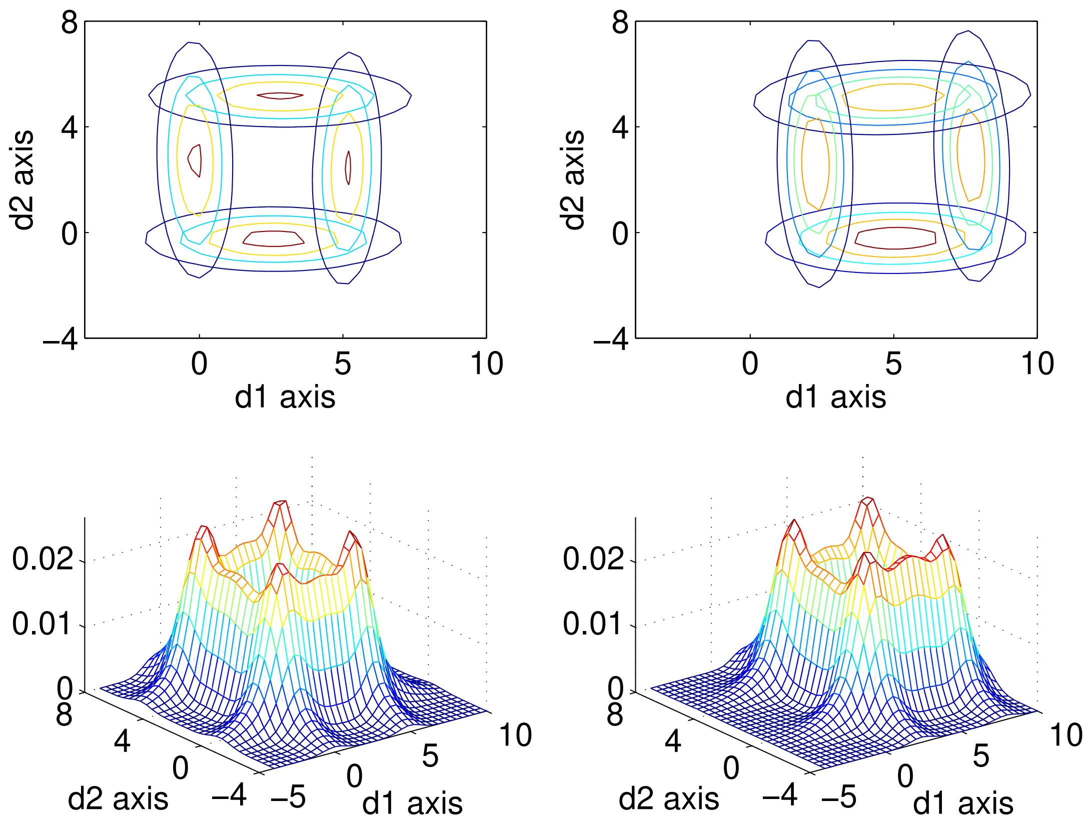


Figure 5.5 *Suboptimal approximation of the training data by Gaussian mixture models.*

mixtures with other components as well. Further details relating to the EM training of mixture models can be found in (Redner and Walker, 1984) and (Xu and Jordan, 1996). The work in (Xu and Jordan, 1996) is, essentially, an application of the results in (Jordan and Xu, 1995) to the special case where the gating networks are constant. In (Redner and Walker, 1984), the authors remark that the training data likelihood for a mixture model will in general be unbounded. This can be seen by considering the example of a Gaussian mixture with 2 components. If one of the mixtures and the mixture weights are kept constant then the likelihood of the training data under the model can be arbitrarily increased if the mean of the second mixture component is identical to one of the training data points and its variance tends to zero. This shows that the points at which the likelihood of the model goes to infinity correspond to invalid model parameters. The parameters of a mixture model can therefore not be found by searching for a global maximum of the training data likelihood under the model. In (Redner and Walker, 1984), the ML parameter estimation problem is therefore reformulated. Softmax functions clearly avoid these problems since their error surface is always bounded.

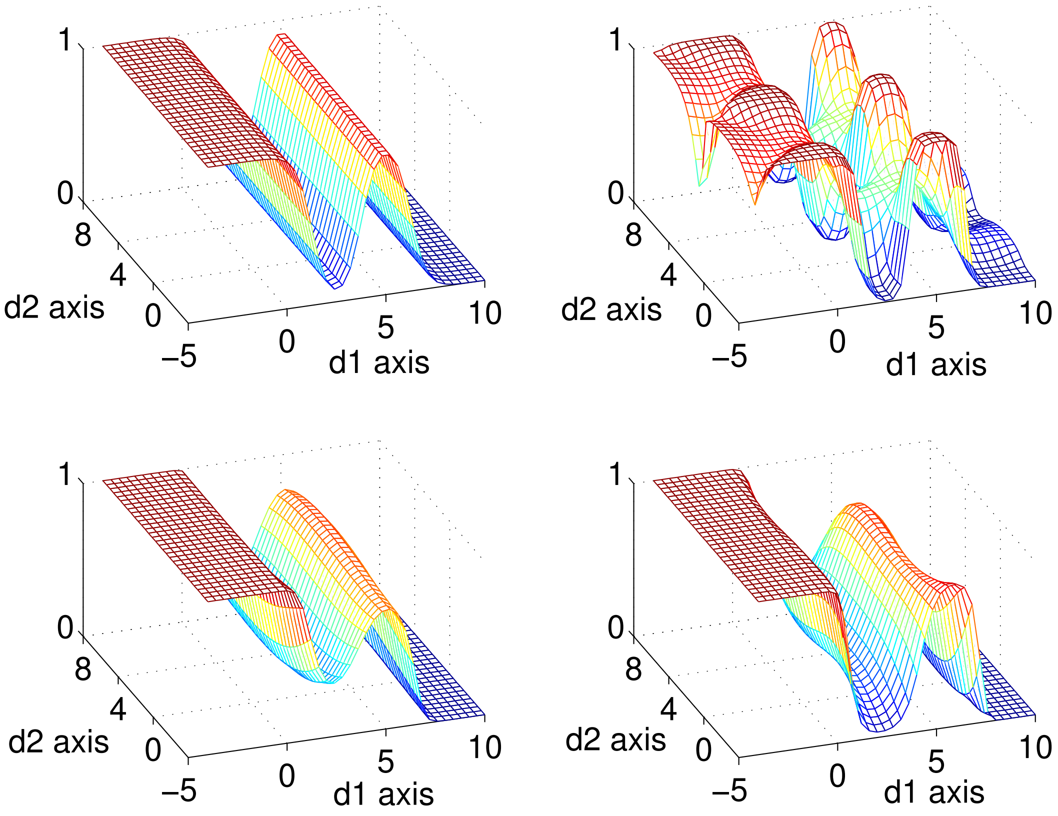


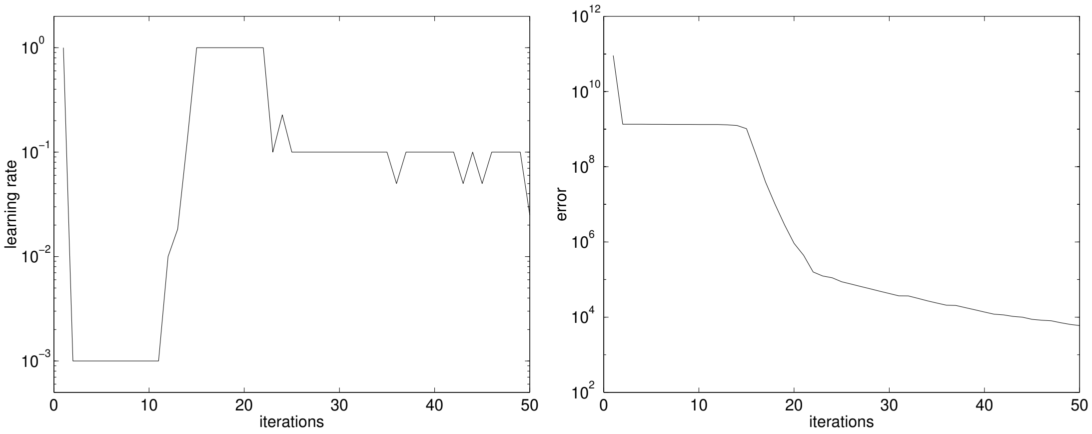
Figure 5.6 *True class posterior and its approximations derived by applying Bayes formula to the GMM's in figure 5.5 and by training softmax functions. The left upper graph shows the true posterior while the right upper graph gives the approximation resulting from Bayes formula. The lower left and right graph show the approximation of the true posterior by a softmax function with second and third degree polynomial, respectively.*

5.4 Implementation details

Since setting the partial derivatives (5.24) to zero does not lead to closed form reestimation formulae for the softmax parameters, it is necessary to employ an iterative scheme to find an approximate solution. The method that will be used in this thesis is the Newton algorithm with line search and back-tracking which is described in detail in (Press et al., 1997). This is a variant of the iteratively reweighted least squared (IRLS) algorithm which is used in connection with generalised linear models (McCullagh and Nelder, 1989; Jordan and Jacobs, 1994). One iteration of the Newton algorithm with line search is given by

$$\mathbf{c}_i^{n+1} = \mathbf{c}_i^n - \rho(\mathbf{c}_i^n)\nabla^2 E(D,\gamma,\mathbf{c}_i^n)^{-1}\nabla E(D,\gamma,\mathbf{c}_i^n) \tag{5.35}$$

where $\mathbf{c}_i^n$ are the softmax parameters after the n-th iteration and $\nabla^2 E(D,\gamma,\mathbf{c}_i^n)$ and $\nabla E(D,\gamma,\mathbf{c}_i^n)$ are the Hessian and gradient of the error function $E(D,\gamma,\mathbf{c})$. The components of the gradient are given by equation (5.24) and the Hessian can be derived from equations (C.17) and (C.18) in appendix C. The scalar $\rho(\mathbf{c}_i^n) \leq 1$ is the positive parameter that determines how far one moves in the Newton direction $\nabla^2 E(D,\gamma,\mathbf{c}_i^n)^{-1}\nabla E(D,\gamma,\mathbf{c}_i^n)$. The parameter $\rho(\mathbf{c}_i^n)$ is sometimes referred to as the learning rate of the algorithm. It is necessary to allow $\rho(\mathbf{c}_i^n)$ to be strictly smaller than one because taking a full Newton step far away from the solution can lead to oscillatory behaviour in the training algorithm. To illustrate the relationship between learning rate and error, both values have been plotted for the first 50 iterations in the training of the 8th degree polynomial in figure 5.2. As can be seen, a full step in the Newton direction in the first iteration



(a) Learning rate. (b) Cross-entropy error.

Figure 5.7 *Learning rate and cross-entropy error for the first 50 iterations in the training of the polynomial in figure 5.2.*

reduces the error by about 2 orders of magnitude. A decrease in the learning rate in subsequent iterations means that the error decreases very slowly but high values of $\rho(\mathbf{c}_i^n)$ in later iterations, again, lead to fast convergence. This variation in the learning rate has been observed in all situations where the initial parameters were far away from the correct solution.

Equation (5.35) shows that each Newton iteration requires the evaluation of the Hessian and the gradient of $E(D, \gamma, \mathbf{c}_n)$ for each new parameter set $\mathbf{c}_n$. One can see from the derivatives of $E(D, \gamma, \mathbf{c})$ in (5.24), that these evaluations rely on the presence of the complete training sequence of the d_t's and their corresponding $\gamma_t(i,k)$'s. In order to carry out more than one Newton iteration (5.35), it is therefore necessary to store both the sequences d_t and $\gamma_t(i,k)$ for each i and k. Alternatively, if one restricts the Newton algorithm to just one iteration, the statistics of the gradient and Hessian of $E(D, \gamma, \mathbf{c})$ can be directly accumulated from the training data, and the sequences d_t and γ_t can be discarded. However, in this case the learning rate has to be fixed a priori because a line-search with back-tracking would need to evaluate the gradient of the error function repeatedly which again makes use of the complete training data. The two different training schemes can be summarised as follows.

One step Newton algorithm Use the forward-backward algorithm to calculate the state likelihoods $\gamma_t(i,k)$ and accumulate the statistics of the gradient and Hessian of $E(D, \gamma, \mathbf{c}_i)$ for all the parameter sets $\mathbf{c}_i$ by using equation (5.24) and equations (C.17) and (C.18) in appendix C. Choose a learning rate $\rho(\mathbf{c}_i)$ for each parameter set $\mathbf{c}_i$ and apply the Newton iteration (5.35) with the accumulated values.

Full Newton algorithm Use the forward-backward algorithm to calculate the state likelihoods $\gamma_t(i,k)$ and buffer the sequences $\gamma_t(i,k)$ and d_t for each combination of states i,k. Calculate the gradient and Hessian of $E(D,\gamma,\mathbf{c}_i)$ repeatedly from these buffers and apply Newton iteration (5.35) until convergence.

The advantage of the first scheme is that it uses relatively little storage since only the gradient and Hessian of $E(D,\gamma,\mathbf{c}_i)$ have to be stored for each state $s=i$. The application of just one Newton step (5.35), on the other hand, will result in slow convergence of the model and will require many forward-backward evaluations even though the parameters of the other model components might already be stationary. In addition, the learning rate has to be chosen by hand because evaluations of the error function to find the optimal learning rate are not possible in this case.

The disadvantage of the second method is that it uses considerably more storage since it has to store all the likelihoods $\gamma_t(i,k)$ and the training data d_t for each time instant and for all states $s=i$ and $v=k$. However, since the complete data are present during the model update the Newton iterations (5.35) can be applied until the model parameters converge.

In this work, the second method has been implemented. In order to restrict the size of the data buffers and to speed up the calculation of error values, gradients and Hessians, these buffers can be restricted to a certain size, which has been done in some of the experiments described in chapter 7. In addition, only data points which had a likelihood $\gamma_t(i)$ beyond a certain threshold were stored in the buffer and for each of these the $\gamma_t(i,k)$ were stored in likelihood buffers for each of the states of variable v. Since there are usually several training utterances the likelihoods $\gamma_t(i,k)$ are normalised by the overall likelihood of the training utterance. Figure 5.8 shows an example of such buffers that contain the d_t's and normalised $\gamma_t(i,k)$'s. In this example the feature d is one dimensional, the hidden variable v has 3 states and the maximal buffer size has been restricted to 2000 data points. Experiments in chapter 7 will show that it is, in fact, possible to limit the size of the buffers for such low dimensional features without observing performance degradation due to undertrained system parameters. For higher dimensional features, however, the application of a likelihood threshold is more appropriate.

The normalised likelihood of the first state $\gamma_t(i,1)$ is plotted in the top graph of figure 5.8 against the corresponding feature d_t. These data are represented by blue circles. The red line in this graph shows a softmax function with linear exponents that has been trained on these data. The middle and bottom graph in figure 5.8 contain the same information for the second and third state of the hidden variable v, respectively. Figure 5.8 shows that the normalised likelihoods $\gamma_t(i,k)$ are mainly concentrated around 0 and 1 and that the training algorithm converges to a set of softmax functions that model the distribution of the training data reasonably well.

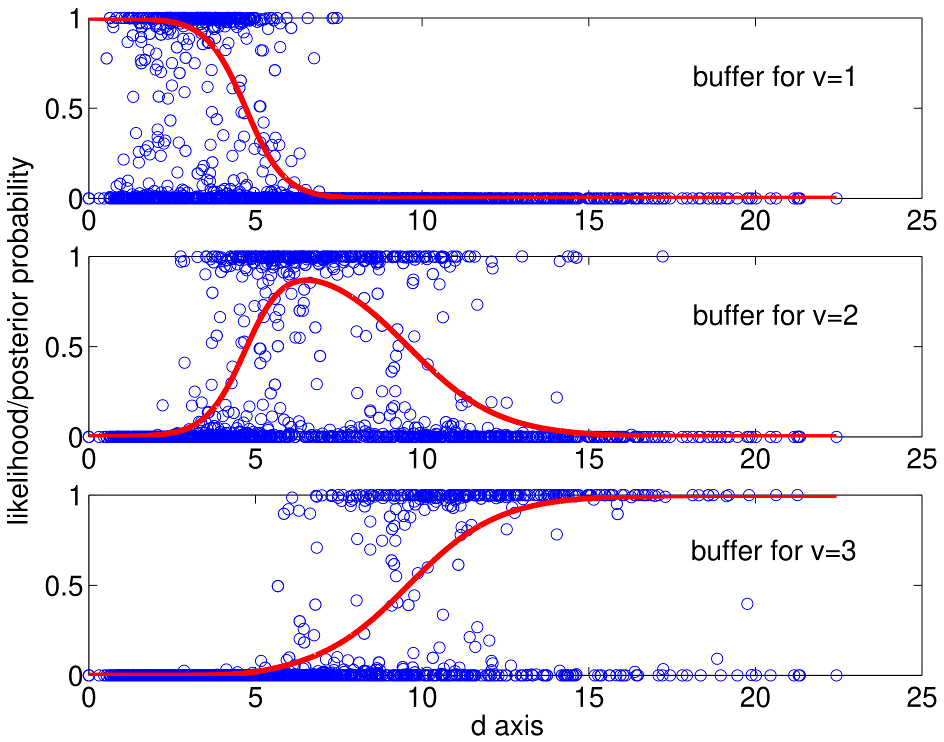


Figure 5.8 *Example of likelihood buffers and approximating softmax functions. The hidden variable v has 3 states and the feature d is one-dimensional. The contents of the buffers are visualised by the circles in this figure and the approximating softmax functions, which have linear exponents, are indicated by the continuous lines.*

5.5 Initialisation

Initialisation of the model parameters can be carried out in a number of ways. Two different approaches will be investigated in this thesis. Both initialise some parts of the model by estimating their parameters on the training data, while the remaining parts of the model are either set to predefined values or to values that are estimated from the trained components.

5.5.1 Relabelled training initialisation

The initialisation method discussed in this section describes a method to estimate the output pdf's $b(o|v,s)$, starting from the output pdf's $b(o|v,s)$ of a standard HMM, by training the latter on a particular transcription of the training data. The $b(o|v,s)$ will subsequently be used to estimate the $b(d|s)$ output pdf's. This method will be called relabelled training initialisation (RTI).

A standard HMM is usually trained on a phonetic transcription of the training data. If, in addition to the sequence of phonetic labels, also the sequence of states of the hidden variable v is known then this information can be used to train v-state dependent HMM's. This can be done by adding the state of v to the phonetic label of the training data transcription. This process is illustrated in figure 5.1. Here the word "often" has been transcribed phonetically with a sequence

sil-ao+f	ao-f+ax	f-ax+n	ax-n+ae
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
$v = 1$	$v = 2$	$v = 3$	$v = 2$
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
sil-ao+f_v1	ao-f+ax_v2	f-ax+n_v3	ax-n+ae_v2

Table 5.1 *Adding v-state information to a phonetic transcription of the word "often".*

of triphones. The second row in figure 5.1 shows which of the states of v is present during each phonetic unit and the last row shows the resulting transcription which adds information about the v state to the phonetic labels. The acoustic models of the v-state dependent labels in figure 5.1 can be initialised with the acoustic models of the corresponding v-state independent labels. Training these models on the v-state dependent transcription will result in estimates for the $b(o|v, s)$.

The process depicted in figure 5.1 makes two assumptions. The first is that the state of the variable v does not change within a phonetic unit. The second assumes the existence of a method to explicitly determine the state of v at each point in time. Both assumptions are only valid to a certain degree which explains why the RTI method should only be used as an initialisation scheme.

Given the output probabilities $b(o|v, s)$ for the different states of v it is sometimes possible to find good estimates for the output probability $b(d|s)$. This depends on the relationship between o and d. In section 6, for instance, d will be derived by a linear transform from o and by calculating the squared Euclidean norm of some subvectors of o. In both cases a reasonable estimate of what $b(d|s)$ should be can be derived. These will be discussed in the following.

If d is the squared Euclidean length of a higher dimensional vector which is distributed according to a Gaussian distribution with mean μ and covariance matrix $\Sigma = \text{diag}(\sigma^2)$ then according to the theory of non-central χ^2 functions in (Kendall and Stuart, 1973a), it is possible to approximate $b(d|s)$ with a Gamma distribution. The parameters of this approximation are given by

$$\eta = \frac{n+\lambda}{2\sigma^2(n+2\lambda)} \tag{5.36}$$

$$\nu = \frac{(n+\lambda)^2}{2(n+2\lambda)} \tag{5.37}$$

where λ is the so-called displacement parameter which is defined as by

$$\lambda = \mu\Sigma^{-1}\mu' \tag{5.38}$$

Figure 5.9 illustrates the quality of this approximation. Here the top graph shows the empirically determined distribution of the squared Euclidean norm, which is represented by the dashed line and the approximation with a Gamma density which is represented by the solid line. In this experiment a 13 dimensional Gaussian was chosen with a mean of length about 1 and

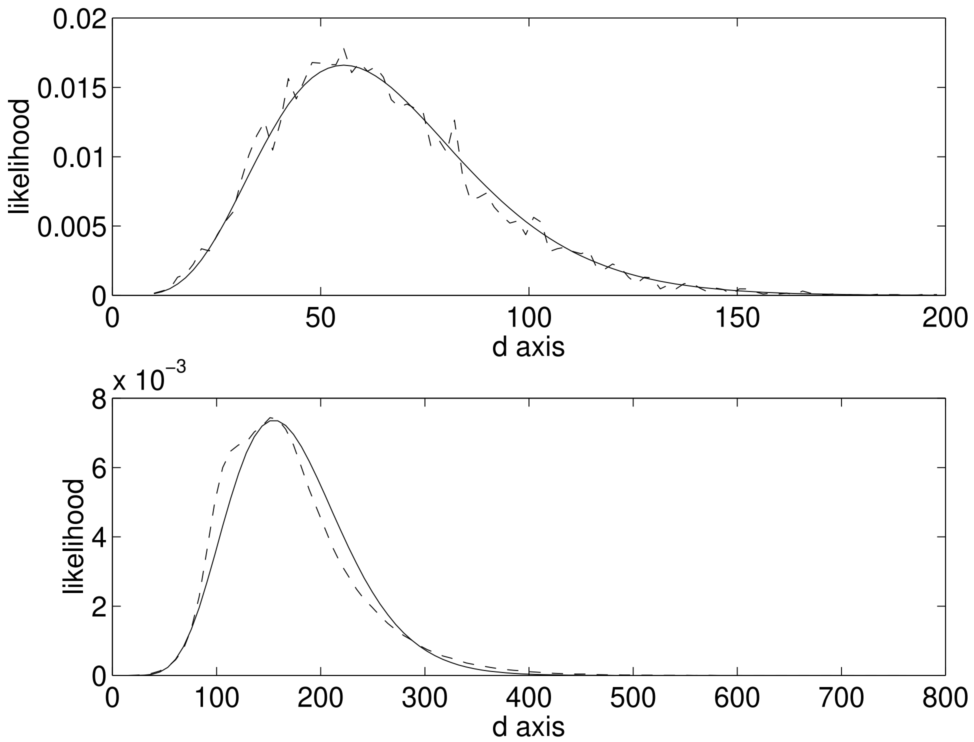


Figure 5.9 *Approximation of non-central χ^2 function by Gamma density.*

variance $\sigma^2 = 5$. This Gaussian was sampled and the squared Euclidean length of the samples was calculated to derive the dashed distribution. As can be seen, the Gamma density that was estimated with (5.36) and (5.37) based on the Gaussian model parameters fits the empirically derived distribution very well. The reason why the dashed distribution looks rather "spiky" is due to the relatively small sample size of the Gaussian distribution in this example. For larger sample sizes the dashed distribution becomes much smoother and the approximating Gamma and the true distribution become visually almost indistinguishable.

The theory developed above can now be used to initialise $b(d|s)$ in the following way. Suppose the $b(o|v,s)$ are known for the K different states of v. Then, assuming that the v states are not biased, i.e. $p(v = k|s) = 1/K$, the overall distribution of o given s is given by

$$\sum_{k=1}^{K} \frac{1}{K} b(o|v = k, s) \tag{5.39}$$

If the $b(o|v = k, s)$ are GMM's, the distribution in (5.39) is again a GMM. In order to apply (5.36) and (5.37) it is therefore necessary to replace the GMM distribution with a single Gaussian with diagonal covariance $\Sigma = diag(\sigma^2)$. While this might result in a poor approximation of the true distribution of o the effect on the resulting Gamma densities is less dramatic. This is, for example, illustrated by the graph at the bottom of figure 5.9. Here a 13 dimensional GMM with 12 mixture components and different variances was first approximated by a single 13 dimensional Gaussian with diagonal covariance for which the corresponding Gamma density was calculated. The true distribution in this graph is the dashed line which was again derived by sampling the 13 dimensional GMM and calculating the squared Euclidean norms of the samples. As can be seen from the bottom graph in figure 5.9 the approximation estimated from the GMM parameters is relatively accurate.

Next, a simple method that replaces a Gaussian mixture model with a single Gaussian will be introduced. Since all the GMM's considered here have diagonal covariances it is sufficient to study the problem for a one dimensional situation where the GMM is given by

$$\sum_{i=1}^{N} w_i \mathcal{N}(o, \mu_i, \Sigma_i) \tag{5.40}$$

Here w_i is the weight of the i-th one dimensional Gaussian $\mathcal{N}(o, \mu_i, \Sigma_i)$ which is given by

$$\mathcal{N}(o, \mu_i, \Sigma_i) = \frac{1}{\sqrt{2\pi\Sigma_i}} e^{-\frac{(x-\mu_i)^2}{2\Sigma_i}} \tag{5.41}$$

First the mean of the GMM distribution (5.40) can be derived as follows

$$\int \sum_{i=1}^{N} w_i \mathcal{N}(o, \mu_i, \Sigma_i) \mathbf{d}o = \sum_{i=1}^{N} w_i \mu_i \tag{5.42}$$

which is the weighted sum of the means μ_i of the components of the GMM and the second order moment of (5.40) is given by

$$\int o^2 \sum_{i=1}^{N} w_i \mathcal{N}(o, \mu_i, \Sigma_i) \mathbf{d}o = \sum_{i=1}^{N} w_i (\mu_i^2 + \Sigma_i) \tag{5.43}$$

The total variance of the GMM distribution can now be derived by subtracting the squared overall mean (5.42) from the second order moment (5.43). Replacing the GMM with a single Gaussian with mean and variance that have been calculated in this way therefore gives the same overall mean and variance. In order to ensure that all the individual variances in a diagonal higher dimensional covariance matrix are the same these have been replaced by their averages.

If the feature d is derived from o via a dimensionality reducing linear transform M, estimating $b(d|s)$ from the $b(o|s, v)$ under the assumption of unbiased class distributions $p(v = k|s) = 1/K$ is much simpler, since the Gaussian mixture that models the transformed vectors d can be exactly calculated from the Gaussian mixture of the original vector o. The weights of the components stay the same and the Gaussian mixture components of the original o have to be replaced by the Gaussians of the transformation d which for the i-th mixture component are given by

$$\frac{1}{\sqrt{2\pi^n |M\Sigma_i M'|}} e^{-\frac{1}{2}(d-M\mu_i)'(M\Sigma_i M')^{-1}(d-M\mu_i)} \tag{5.44}$$

where n is the dimension of d. If necessary, this transformed mixture model can be further reduced to a single Gaussian by the same methods as above.

5.5.2 Median split initialisation

It will be seen in section 6 that the method discussed in the previous section suffers from the problem that the division of the training data into v-state dependent classes can result in an uneven distribution of the training data. In particular, one of the classes might be considerably smaller than the others and the models trained for this v-state label are therefore less reliable. The method developed in this section tries to overcome this problem by splitting the feature space of d into areas of equal size. This method will be called the median split initialisation (MSI).

In the following, it will be assumed that d is a one-dimensional feature. In this situation one can find $K-1$ points $m_1, \dots, m_{K-1}$ on the d axis such that

$$\int_{-\infty}^{m_i} b(d|s)\mathbf{d}d = \frac{i}{K} \tag{5.45}$$

In other words, it is possible to divide the d axis into K intervals such that the partial integral of $b(d|s)$ over each interval $[m_i, m_{i+1}]$ is the same. If $K = 2$ the point that divides the d axis into intervals of equal volume is called the median of $b(d|s)$. A method to practically derive the points $m_1, \dots, m_{K-1}$ for an output probability $b(d|s)$ which is a Gamma or Gaussian distribution, will be discussed at the end of this section.

To ensure that the training data are divided evenly between the different states of v one would ideally need a set of K posterior probabilities which are equal to 1 on one of the intervals $[m_i, m_{i+1}]$ and 0 elsewhere. In addition, the union of the intervals where the posteriors do not vanish should be the complete d axis. A similar but slightly less complicated restriction is the requirement that two of the K softmax functions intersect at each of the points $m_1, \dots, m_{K_1}$ with a value that is close to 0.5. For the case $K = 2$ this results in the following equation.

$$S_{1,s}(m_1) = S_{2,s}(m_1) \tag{5.46}$$

where m_1 is the median of $b(d|s)$. In terms of the exponential polynomial of the softmax functions $S_{v,s}(d)$ this means that

$$q_1(m_1) = 0 \tag{5.47}$$

has to hold, which can be satisfied by a polynomial of arbitrary degree. It makes sense, however, to place an additional restriction on the softmax functions. For instance, one might require $S_{1,s}(d)$ to attain a value c which is close to one at a point m_2 which is far away from the median, i.e. $S_{1,s}(m_2) = c$. This can, again, be interpreted as a restriction on $q_1(m_2)$, i.e.

$$q_1(m_2) = \log\left(\frac{c}{1-c}\right) \tag{5.48}$$

In order to satisfy both (5.47) and (5.48) the polynomial $q_1(d)$ has to have at least degree one. For a set of three softmax functions the requirement that two of them intersect at each of the points m_1, m_2 with a certain value c leads to the following equations

$$S_{1,s}(m_1) = S_{2,s}(m_1) = c \tag{5.49}$$

$$S_{2,s}(m_2) = S_{3,s}(m_2) = c \tag{5.50}$$

Equation (5.49) implies that $q_1(m_1) = q_2(m_2)$ has to hold. Substituting this back into (5.49) therefore leads to the following equation

$$q_1(m_1) = q_2(m_1) = \log\left(\frac{c}{1-2c}\right) \tag{5.51}$$

Similarly, equation (5.50) results in the following restrictions

$$q_1(m_2) = \log\left(\frac{1-2c}{c}\right) \tag{5.52}$$

$$q_2(m_2) = 0 \tag{5.53}$$

Equations (5.51), (5.52) and (5.53) represent an interpolation problem which can be solved for linear polynomials $q_k(d)$. Figure 5.10 gives an example of this complete initialisation procedure. The upper graph in figure 5.10 shows the output probability $b(d|s)$ which, in this case, is a

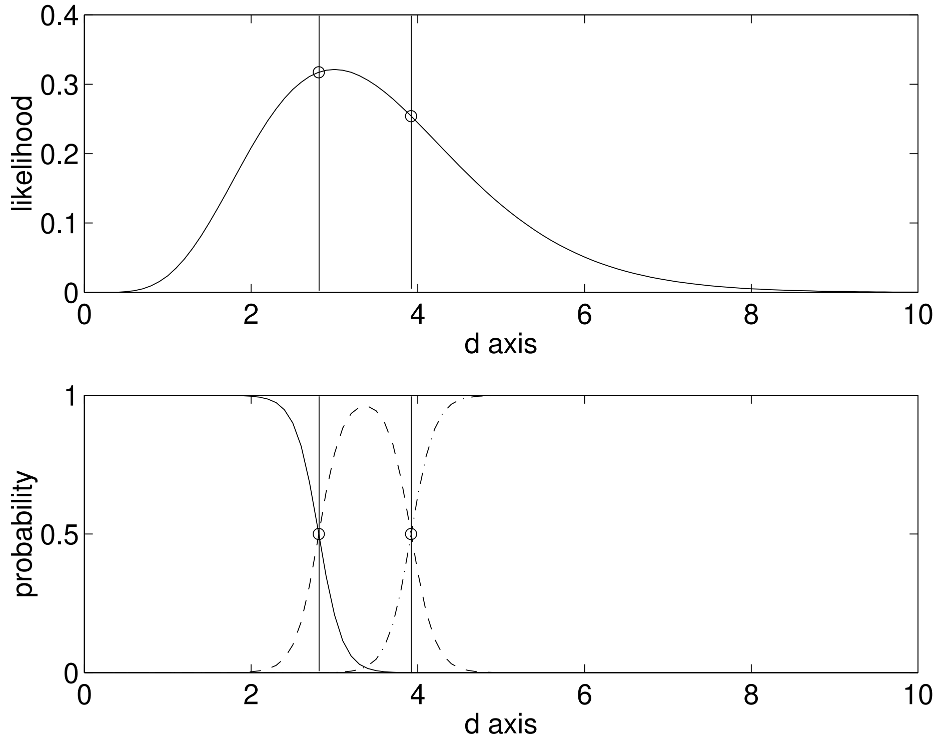


Figure 5.10 *Median split initialisation example. The upper graph shows $b(d|s)$ and the points at which the integral of $b(d|s)$ is $1/3$ and $2/3$, respectively. In the lower graph the softmax functions have been initialised by solving (5.51), (5.52) and (5.53) with $c = 0.4999$. The softmax functions are represented by the solid, dashed and dashed-dotted lines.*

Gamma distribution. The first step in the initialisation method is the determination of the points m_1 and m_2 which satisfy (5.45). These points and their corresponding likelihoods are indicated by the straight lines and the circles, respectively. The lower graph in figure 5.10 shows the softmax functions that were obtained by solving the interpolation problem given by the equations (5.51-5.53). Here the value c was set to 0.4999 which is the value of two of the softmax functions at each of the points m_1, m_2. The distance of c from 0.5 affects the slope of the softmax functions. If one chooses, for instance, a value of $c = 0.49$ the slopes are less steep and the maximum of the softmax function in the middle is closer to 0.5. Since one would ideally like the posteriors to be step functions which assume the values 0 and 1 it is therefore necessary to choose c reasonably close to 0.5.

For a hidden variable v with more than 3 states finding appropriate softmax functions is more complicated. Here, in addition to, (5.49) and (5.50) the following equations have to be satisfied

$$S_{k,s}(m_k) = S_{k+1,s}(m_k) = c \tag{5.54}$$

for all $k = 3, \ldots, K-1$. Unfortunately, this results in a set of non-linear equations in the polynomial parameters for which there is no closed form solution. However, by introducing the additional restriction that all the values $S_{l,s}(m_k)$ are equal for $l \neq k, k+1$ it is possible to reduce the problem to the same form as for $K = 3$. For $K > 3$ this results in an interpolation problem with $K-1$ different points $m_1, \ldots, m_{K-1}$ and therefore requires polynomials $q_k(d)$ of degree $K-2$. Just to give a flavour of what the interpolation constraints look like, here they are derived for $K = 4$.

$$S_{1,s}(m_1) = S_{2,s}(m_1) = c \qquad S_{3,s}(m_1) = S_{4,s}(m_1) \tag{5.55}$$

$$S_{2,s}(m_2) = S_{3,s}(m_2) = c \qquad S_{1,s}(m_2) = S_{4,s}(m_2) \tag{5.56}$$

$$S_{3,s}(m_3) = S_{4,s}(m_3) = c \qquad S_{1,s}(m_3) = S_{2,s}(m_3) \tag{5.57}$$

These equations are the equivalent to (5.49) and (5.50) and they result in the following interpolation problem for the exponential polynomials $q_k(d)$.

$$q_1(m_1) = q_2(m_1) = \log\frac{2c}{1-2c} \qquad q_3(m_1) = 0 \tag{5.58}$$

$$q_2(m_2) = q_3(m_2) = \log\frac{2c}{1-2c} \qquad q_1(m_2) = 0 \tag{5.59}$$

$$q_1(m_3) = q_2(m_3) = \log\frac{1-2c}{2c} \qquad q_3(m_3) = 0 \tag{5.60}$$

This shows that for $K = 4$ the exponential polynomials q_k have to have at least degree two to satisfy the above requirements.

Note that the initialisation method developed in this section relies on the fact that d is one dimensional since the first step in the median-split initialisation scheme is the search for the median. There is no equivalent in the higher dimensional case. In two dimensions and for a Gaussian with diagonal covariance matrix, for instance, every straight line that passes through the mean of the Gaussian separates the feature space into areas of equal volume.

5.5.2.1 Estimating the median

The derivation of the points $m_1, \ldots, m_{K-1}$ requires the calculation of the partial integrals $\int_{m_i}^{m_{i+1}} b(d|s)\mathbf{d}d$. If $b(d|s)$ is modelled by a Gamma distribution these integrals can be derived from the incomplete Gamma function which is defined as follows.

$$P(\nu, x) = \frac{1}{\Gamma(\nu)} \int_0^x e^{-t} t^{\nu-1} \mathbf{d}t \tag{5.61}$$

This is the partial integral of a Gamma density with $\eta = 1$. For $\eta \neq 1$ the partial integrals of $b(d|s)$ are given by

$$\frac{\eta^{\nu}}{\Gamma(\nu)} \int_0^x e^{-\eta t} t^{\nu - 1} \mathbf{d}t = \frac{1}{\Gamma(\nu)} \int_0^{\eta x} e^{-t} t^{\nu-1} \mathbf{d}t = P(\nu, \eta x) \tag{5.62}$$

According to (Abramovitz and Stegun, 1965), the incomplete Gamma function (5.61) can be calculated with the help of the following exponential series.

$$P(\nu, x) = \frac{x^{\nu}}{\Gamma(\nu)} \sum_{n=0}^{\infty} \frac{(-x)^n}{(\nu + n) n!} \tag{5.63}$$

For the purpose of this work it was found to be sufficient to use the first 50 terms of the series (5.63) to obtain a good estimate of $P(\nu, x)$. This approximation to the incomplete Gamma function can now be used in a Newton search to find the appropriate points. One only has to remember that the derivative of the incomplete Gamma function is the Gamma function itself.

For Gaussians the median coincides with mean and therefore nothing has to be calculated for a variable v with 2 states. For variables with more than 2 states the incomplete integrals of the Gaussians have to be calculated which can, again, be efficiently derived from the so-called error function erf(x) which is given by

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \mathbf{d}t \tag{5.64}$$

From this the incomplete integral of a Gaussian with mean μ and variance σ can be derived by the following expression

$$\frac{1}{2}\left(1 + \text{erf}\left(\frac{x - \mu}{\sigma\sqrt{2}}\right)\right) \tag{5.65}$$

According to (Abramovitz and Stegun, 1965), the error function can be approximated for positive values up to an error smaller than 1.5×10^{-7} by the following formula

$$\text{erf}(x) = 1 - (a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5) e^{-x^2} \tag{5.66}$$

where $t = 1/(1 + px)$, with $p = 0.3275911$ and the coefficients of the polynomial in (5.66) are given by

$$\begin{aligned} a_1 &= 0.254829592 & a_2 &= -0.284496736 \\ a_3 &= 1.421413741 & a_4 &= -1.453152027 \\ a_5 &= 1.061405429 \end{aligned}$$

Again this approximation has been used in a Newton search to find the points which divide the space of feature d into areas of equal volume. As previously, one has to remember that the derivative of the incomplete integral is the integrand which, in this case, is the Gaussian.

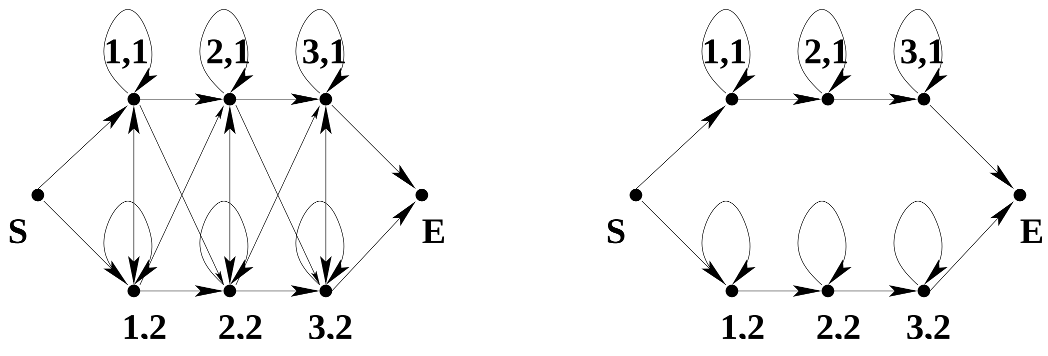
5.6 Recognition

The SBME-HMM can be used in a number of ways for recognition. The particular implementation depends on the treatment of the hidden variable v. If the state of v is not used in the recognition process, one can sum over all v-states to obtain an output probability of o and d that depends only on s, i.e.

$$b(o,d|s) = \sum_{k=1}^{K} b(o,d,v=k|s) \tag{5.67}$$

This will be called the factorised topology.

The easiest way to include information about the state of v into the recognition process is by adding this information to the state of s. In this way, one arrives at combined s,v-states which are represented by a pair of numbers (i,k) where the first number i refers to the state of s while the second number k refers to the state of v. The output pdf's of the s,v-states are then the probability densities $b(o|v,s)$. In order to restrict the search space of the s,v-states it is necessary to define a topology on the pairs of state indices, which amounts to defining transition probabilities $a_{(i,k)(j,l)}$. Two such topologies are illustrated in figure 5.11 for a variable v with two different states. As can be seen, from looking at the s-state indices, the topologies in figure



(a) Full topology. (b) Parallel topology.

Figure 5.11 *Model topologies that explicitly use the state of v in the recognition process.*

5.11 have been derived from the standard 3-state left-to-right HMM in figure 2.4. The transition probabilities in these topologies are time-dependent as they depend on the value of feature d. For the topology in figure 5.11(a), for instance, the transition probabilities are given by

$$a_{(i,k)(j,l)}(t) = a_{i,j}\, p(l|d_t, j) \tag{5.68}$$

These are true probabilities because taking the sum over all the $a_{(i,k)(j,l)}(t)$ with i, k and t fixed gives 1. Since transitions between all the states in this models are allowed this topology will be referee to as the full topology. The transition probabilities for the topology in figure 5.11(b), on the other hand, are given by

$$a_{(i,k)(j,l)}(t) = \begin{cases} a_{i,j}p(l|d_t,j) & : \quad k = l \\ \\ 0 & : \quad k \neq l \end{cases} \tag{5.69}$$

This does not result in a true probability since the sum over all $a_{(i,k)(j,l)}(t)$ with fixed i, k and t is, in this case, equal to $p(k|d_t,j)$. It still makes sense to study the topology in figure 5.11(b), however, because the most likely state sequence in the parallel model will be the most likely state sequence in the full model with the additional restriction that the state of v does not change within a phonetic unit. The topology in figure 5.11(b) is therefore useful for hidden variables v that are assumed to change slowly over time. This topology will be called the parallel topology. It is the same that was used in the trajectory models in (Iyer et al., 1998) which were mentioned in section 3.5.

The auxiliary function for the full topology is the same as for the factorised topology since, in this case, summing over all sequences of s,v-states is equivalent to summing over all sequences of s-states and all sequences of v-states. The transition probabilities $a_{i,j}$ and the v-state probabilities $p(l|d_t,j)$ that have been trained for a factorised topology can therefore be directly substituted into (5.68) to give the transition probabilities between s,v-states in the full topology model. However, for the parallel topology using the parameters that were trained for a factorised topology is suboptimal. Although these parameters could be trained for a parallel topology, as well, this was not done here. Instead, as for the full topology, the parameters that were trained for a factorised topology were used in (5.69) to obtain the s,v-state transition probabilities.

6

Indicating Features

The model that was proposed in the previous chapter makes use of a feature d, called an indicating feature, that is meant to contain information relating to the state of the hidden variable v. In order to derive an efficient model it is therefore necessary to find an indicating feature that distinguishes well between different v-states. This chapter discusses various methods to derive such a feature and compares them in a number of classification experiments.

In contrast to the signal based speaking rate measures that were discussed in chapter 4 a high correlation with ROS measures based on a manual transcription, like phone or syllable rate, is not an issue. Instead of deriving the indicating feature directly from the acoustic signal, the information contained in the standard feature vector o that relates to temporal changes in the speech process will be harnessed for the task at hand. Such information is usually present in the form of first or higher order derivatives and it should therefore be possible to calculate the indicating feature by applying a suitable transform to o.

Section 6.1 gives a brief introduction to two of the most common linear dimensionality reduction schemes, namely principle component analysis (PCA) and linear discriminant analysis (LDA). In addition, a non-linear dimensionality reduction scheme will also be presented which does not require the estimation of a transform.

Section 6.2 will evaluate the performance of the indicating features derived with the different dimensionality reduction schemes from section 6.1 in a number of classification experiments. First section 6.2.1 will address the problem of defining classes that are correlated to what are assumed to be states of the hidden variable v. These methods will be subsequently applied to define 2 and 3 class problems for all the phones in the phone set in appendix A. The results of the experiments for the different dimensionality reduction schemes on the 2 class tasks will be reported in section 6.2.2, whereas section 6.2.3 discusses the 3 class classification results. In both cases phone dependent and phone independent LDA and PCA transforms are estimated which gives rise to model dependent and independent indicating features.

Finally, section 6.3 reviews the experimental work done in this chapter and highlights the key results.

6.1 Dimensionality Reduction

It was already mentioned in the introduction to this chapter that the standard feature vector o itself will be used to derive the indicating feature d. Since d will be employed to distinguish between different states of the hidden variable v, this approach is similar to the one in subband and multi-stream models, as discussed in section 3.2, where feature o is also used to infer the best combination of subbands or streams for recognition. Rather than using o directly, however, here functions $f(o) = d$ will be investigated that extract all the information in o relevant to the state of v and put this information into feature d.

Ideally, the function $f(o) = d$ should reduce the dimensionality of o as much as possible and at the same time be easy to calculate. These two objectives motivate the attempt to find a suitable indicating feature d by reducing the dimension of o with an appropriate linear map. Two such linear dimensionality reduction schemes will be investigated in this section. In addition, a simple non-linear dimensionality reduction scheme will also be discussed.

6.1.1 Principal component analysis

Principal component analysis (PCA) (Kendall and Stuart, 1973b), which is also known as the Karhunen-Loeve transform (Papoulis, 1984), attempts to decorrelate the components of a feature vector. Here $O = \{o_1, \ldots, o_T\}$ is a sequence of vectors with mean m and the covariance matrix of these data is therefore given by

$$\Sigma = \sum_{t=1}^{T} (o_t - m)(o_t - m)' \tag{6.1}$$

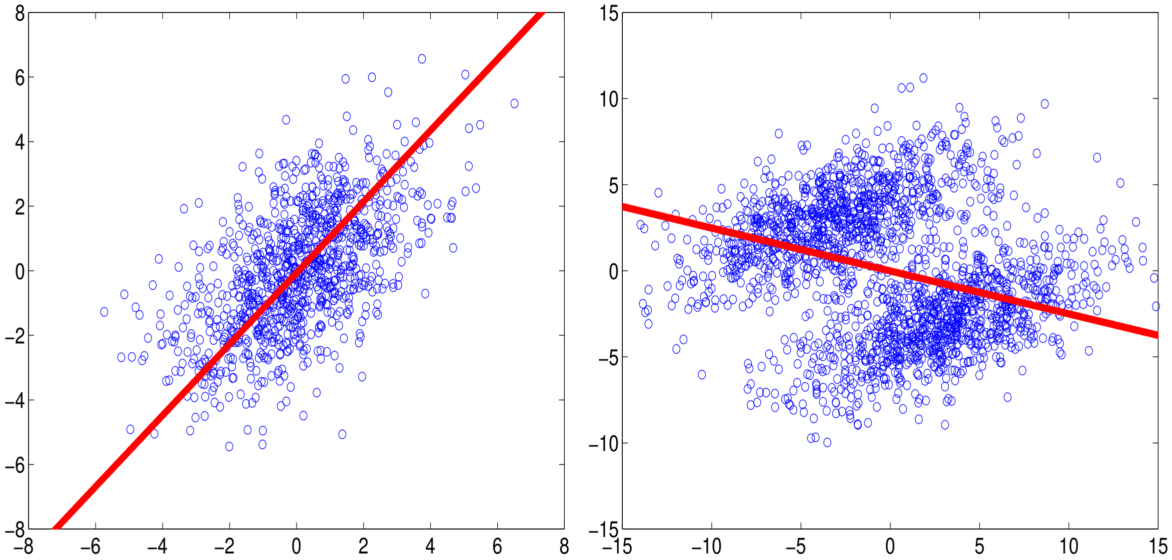
The index t above is not necessarily a time index but can be any counter on the vectors in O. Similarly o_t does not have to be a feature vector in a speech recognition system but can be an arbitrary vector. Since here PCA will be primarily applied in the context of speech recognition, however, a more general notation will not be introduced. In order to decorrelate the components of o one has to find a transformation P such that the transformed sequence $PO = \{Po_1, \ldots, Po_T\}$ has diagonal covariance. In terms of the covariance matrix of O (6.1) this means that P has to satisfy the following equation

$$P\Sigma P' = \text{diag}(\sigma_1^2, \ldots, \sigma_n^2) \tag{6.2}$$

where $\text{diag}(\sigma_1^2, \ldots, \sigma_n^2)$ is a diagonal matrix with positive diagonal elements $\sigma_1^2, \ldots, \sigma_n^2$ and the σ_i are assumed to be ordered according to their absolute value, i.e. $\sigma_1^2 > \sigma_2^2, \ldots, \sigma_n^2$. From (6.2) one can see that the rows of P have to be the eigenvectors of Σ. Here the eigenvector of Σ that is associated with the highest eigenvalue is the direction in the feature space in which the data O has the highest variance. Likewise, the eigenvector associated with the second highest eigenvalue σ_2^2 is the direction in the orthogonal complement of the first eigenvector along which the data O have the highest variance. Similar interpretations apply to the higher eigenvalues of Σ. If one wants to reduce the dimension of o as much as possible while at the same time retaining most of the information of the complete sequence O, it is therefore sensible to project

o onto the eigenvectors of Σ with the highest eigenvalues. This amounts to removing rows with high indices from the PCA transform and directions in the vector space of o where variation is small.

Figure 6.1 shows the location of the eigenvector of Σ corresponding to the highest eigenvalue for two two-dimensional data sets. Here the o_t are indicated by circles. Figure 6.1(a) shows an



(a) Direction of highest variation for a single class.

(b) Direction of highest variation for two classes.

Figure 6.1 *One-dimensional PCA transforms of a single class and two classes. The PCA transforms are the orthogonal projections onto the straight lines. Each class in 6.1(b) corresponds to one of the ellipsoids in this figure.*

example where the o_t were sampled from a single Gaussian with mean $(0, 0)$. Projecting the data set onto the line in figure 6.1(a) will minimise the loss of information because the data set is concentrated around this line.

While such an orthogonal projection might be effective for a data set which has been sampled from a single distribution this method can lead to undesirable effects in the case of multi-class problems. This is, for instance, illustrated in figure 6.1(b) where the data O have been sampled from two distributions with the same variance but with means at $(-3, 3)$ and $(3, -3)$. Since the aim is to distinguish well between the two classes, a projection into one dimension should result in two well separated distributions. Figure 6.1(b) clearly shows that an orthogonal projection of O onto the straight line in this figure will not result in a good separation of the data. In this example, the eigenvector that corresponds to the highest eigenvalue of the covariance matrix Σ is therefore not ideal. This problem of the PCA dimensionality reduction scheme is due to the fact that the derivation of the PCA transform does not take class membership into consideration. Another method that does so will be discussed in the next section.

6.1.2 Linear Discriminant Analysis

Linear discriminant analysis (LDA) (Duda and Hart, 1973; Fukunaga, 1990) differs from PCA in that it explicitly refers to the class membership of the data points o_t in the training set. Here the objective is to find a linear transform P of O such that the discrimination between the transformed data PO is maximal between different classes. This problem is formalised with the help of the between-class and within-class scatter matrices which will be denoted by S_B and S_W, respectively. For a set of K classes $X_1, \dots, X_K$ these matrices are defined by

$$S_W = \sum_{i=1}^{K} S_i \tag{6.3}$$

$$S_i = \sum_{o_t \in X_i} (o_t - m_i)(o_t - m_i)' \tag{6.4}$$

and

$$S_B = \sum_{i=1}^{K} n_i (m_i - m)(m_i - m)' \tag{6.5}$$

where m_i is the estimated mean of class X_i and m is the overall mean of the training data. With S_B and S_W the separability of the classes $X_1, \dots, X_K$ can now be measured by the quotient of the determinants of S_B and S_W, i.e.

$$\frac{|S_B|}{|S_W|} \tag{6.6}$$

This measure is motivated by the fact that the determinant of a matrix is the product of its eigenvalues. Since S_B and S_W are the sums of covariance matrices, whose eigenvalues are the variances of certain distributions, the determinants of the within and between class scatter matrices can be interpreted as measures of the variance within and across classes, respectively. A large value of (6.6) therefore ensures that the means of the classes $X_1, \dots, X_K$ are well separated while at the same time the data within each class are close to the class mean. A projection that discriminates well between classes will therefore maximise the quotient (6.6). Transforming the data O with P results in the following within class and between class scatter matrices $\tilde{S}_B$ and $\tilde{S}_W$, respectively.

$$\tilde{S}_B = P' S_B P \tag{6.7}$$

$$\tilde{S}_W = P' S_W P \tag{6.8}$$

The LDA transform is therefore the transform P which maximises the following expression.

$$\frac{|P' S_B P|}{|P' S_W P|} \tag{6.9}$$

It can be shown (Wilks, 1962) that the maximisation of (6.9) can be achieved by solving the following generalised eigenvalue problem

$$S_B p_i = \lambda_i S_W p_i \tag{6.10}$$

The columns of the LDA transform P are then the p_i in (6.10) which correspond to the highest non-zero generalised eigenvalues λ_i. Since S_B is the sum of K matrices of rank one or less and only $K - 1$ means m_i and m are linearly independent, S_B is of rank $K - 1$ or less. Therefore there are at most $K - 1$ non-zero λ's and P is a projection into a space of dimension at most $K - 1$. Figure 6.2 gives an example of an LDA transform which has been calculated for the same training data as in figure 6.1(b). Here the straight line represents the direction in the

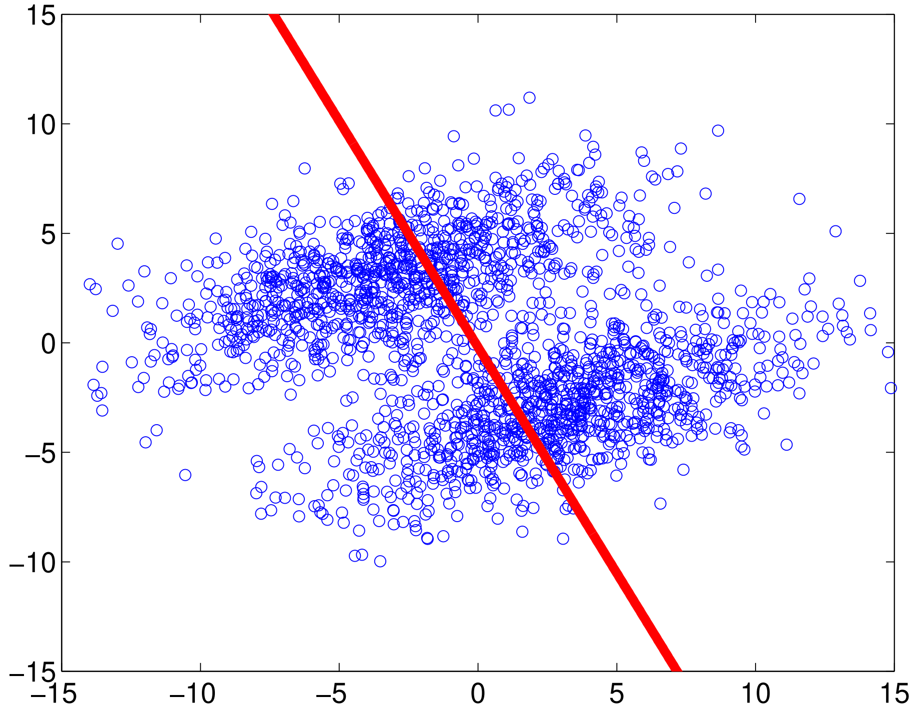


Figure 6.2 *LDA transform of two classes. The LDA transform is the orthogonal projection onto the straight line.*

feature space on which the training data are projected orthogonally. As can be seen the LDA transform results in two one-dimensional distributions that are much better separated than the distributions of the one-dimensional PCA transform in figure 6.1(b).

It is interesting to note that LDA transforms have also been used in speech recognition to project the features o into a more discriminative feature space (Haeb-Umbach and Ney, 1992). A variant of LDA which is called heteroscedastic discriminative analysis (HDA) has also been investigated (Schukat-Talamazzini et al., 1995; Gopinath, 1998; Saon et al., 2000). This method estimates the feature space transformation by optimising an objective function different from the expression in (6.9). In contrast to (6.9), optimisation of the HDA objective function can be interpreted as a constrained maximum likelihood estimation problem (Saon et al., 2000) and is therefore also related to the concept of semi-tied covariances (Gales, 1999). Although deriving an indicating feature with an HDA trained transform will not be investigated further in this work, it is interesting to note that, in principle, such a transform could be trained within the ML paradigm with the same methods that were developed for the HDA case.

6.1.3 Squared Euclidean Norm

In addition to the two linear dimensionality reduction schemes mentioned in section 6.1.2 and 6.1.1 a non-linear method will also be investigated in this chapter. This method simply calculates the feature d by taking the squared Euclidean norm of subvectors of o. The squared Euclidean norm of the first or second order derivatives of feature o, for instance, should be a good source of information relating to speaking rate because the length of these vectors can be interpreted as a measure for the speed and acceleration of the sequence O. This measure also has the advantage that it can be easily calculated without estimating a linear projection on the training data.

6.2 Classification Experiments

This section describes classification experiments that are intended to evaluate the usefulness of the different dimensionality reduction methods discussed in the previous section. All the experiments were carried out on the training set of the 1997 Broadcast News task. The feature vector o consisted of 13 PLP coefficients together with their first and second order derivatives which gives a feature vector o with 39 dimensions in total. In the following, the first and second order derivatives in o will be denoted by Δ and $\Delta\Delta$, respectively. The PLP coefficients themselves will be called static components of o.

6.2.1 Defining Classes

In order to evaluate the quality of the indicating feature d, it is necessary to construct training and test sets which consist of features d_t and what is presumed to be the state of v at time t. The state of v will be defined with the help of a forced alignment on the training data of the 1997 Broadcast News task, which will be used to find phone boundaries. Based on this information the durational statistics of each phone will be collected, which will then be used to split the occurrences of each phone into classes of approximately equal size. This procedure is illustrated by the diagram in figure 6.3. Here the result of collecting the durational statistics is indicated by a list of durations which are measured in frames and the number of times phone “ah” appeared in the training data with this duration. Phone “ah” is just one example. These statistics are collected for each phone in the 45 strong phone set in appendix A. The result of defining the classes for phone “ah”, is illustrated in the bottom left hand corner of figure 6.3. Here every instance of the phone that has a duration between one and four frames belongs to class one which is a class of fast phones, while a phone that has a duration of 5 or 6 frames belongs to class 2 which contains slightly slower phones, etc.

An example of the durational statistics of a phone, in this case the phone “oy”, is given in figure 6.4 which confirms the assumption in (Levinson, 1986) that the probability distribution of phone durations can be modelled by a Gamma density. Given the statistics in figure 6.4, the task of defining K v-states of approximately equal size can now be realised by finding durations $d_1, \dots, d_{K-1}$ such that the sum of the occurrences of the phone that have durations between two successive d_i's are approximately equal. This can be formulated more precisely by requiring

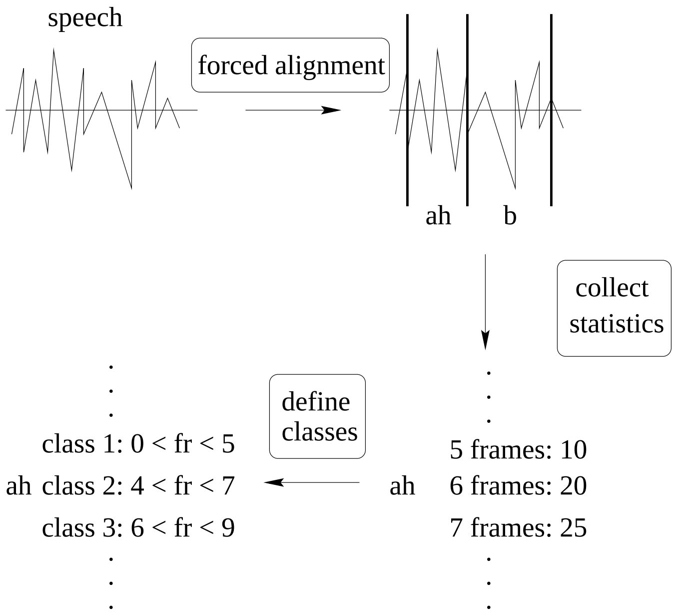


Figure 6.3 *Derivation of v-state labels.*

the d_i to satisfy

$$\sum_{d_i \leq d \leq d_{i+1}} p(d) \approx const. \tag{6.11}$$

where $p(d)$ is the number of occurrences of the phone for duration d. This is similar to finding the median in the MSI method in section 5.5.2. Here, however, the variable d is discrete.

The simplest approach to finding d_i's that satisfy (6.11) is to divide the total number of occurrences of the phone by the number of classes K which gives the number of elements in each class. Then one can move along the duration axis in the durational statistics of the phone, such as in figure 6.4, and define a boundary d_i whenever the sum of the phone occurrences reaches the class size. In figure 6.4, for example, the total number of occurrences of phone "oy" in the forced alignment was 3162. The class size for a variable v with two states is therefore 1581. Summing over the occurrences starting from duration 1 this class size is reached for a duration of $d_1 = 13$. This boundary splits the instances of phone "oy" into one class of size 1599 and another one of size 1563.

In order to ensure relatively equal class sizes across all phones, it is necessary to define the class boundaries d_i for each phone individually, since the durational statistics can be rather different for different phones. Consonants, for instance, exhibit much less variation in their duration than vowels and also tend to be shorter on average.

Given the class of the phone, one now has to associate a feature with it. In principle, this could be any feature within the phone boundaries. To avoid problems as a result of the imprecise estimation of phone boundaries in the forced alignment, only the feature d_t in the middle of the phone boundaries was used. The resulting set of labelled data points (d_t, v) was then divided into training and test sets by assigning 20% of the data points for each phone to the test set. This gave a training set with 1 374 863 data points and a test set with 343 691 data points. The

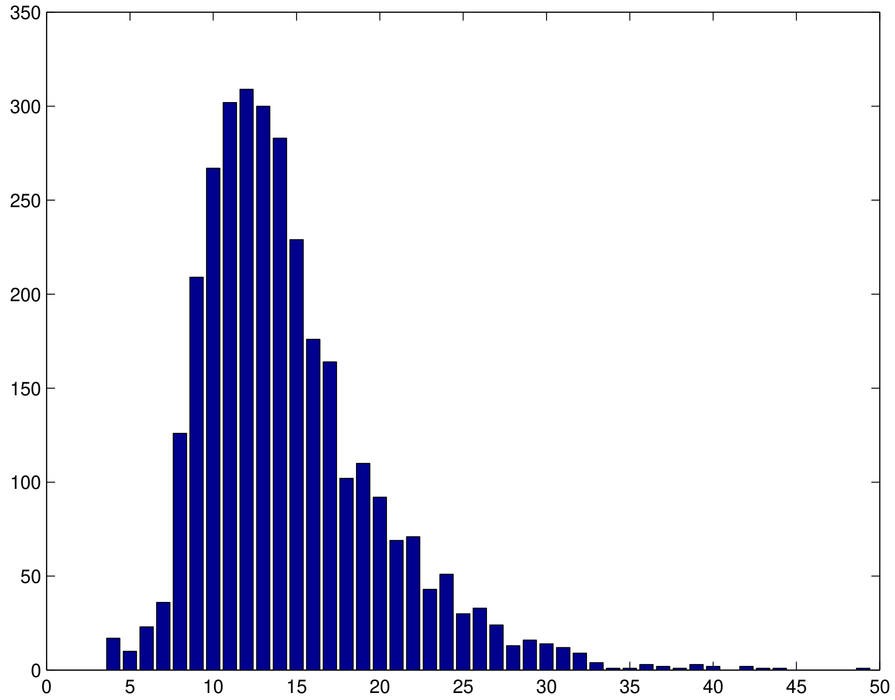


Figure 6.4 *Statistics of duration of phoneme “oy”.*

phone dependent training sets varied between 292 and 115 347 data points while the phone dependent test sets varied between 72 and 28 836 data points. The exact statistics of these sets can be found in appendix A.

The quality of the feature d was evaluated by building classifiers for each of the phones separately and applying them to the test set of each phone. This gives a performance metric for each of the phones which will be combined to obtain an overall performance metric. One can, for instance, derive an overall metric by simply calculating the mean over the classifier performances of all phones. However, this does not take into consideration that the classification tasks can be of different complexity. To illustrate this problem, it is helpful to study again the durational statistics in figure 6.4. To split the occurrences of phone “oy” into 3 classes, one first has to calculate the desired class size which, in this case, is $3162/3 = 1054$. Moving along the duration axis one can see that this class size is reached for $d_1 = 12$ and $d_2 = 17$. The corresponding class sizes are 1299, 1152 and 711, respectively, which is a much more biased distribution than in the 2 class case. The 3 class classification problem for phone “oy” is therefore simpler than a 3 class problem with equal class sizes. To account for the difference in complexity, the overall performance metric was calculated as the weighted sum of the individual performances where each weight reflects the difficulty of the task, i.e.

$$\text{perf}_{\text{total}} = \sum_{h \in H} w_h \text{perf}(h) \tag{6.12}$$

where h represents a phone in the phone set H. Dividing the size of the classes by the total number of elements in the test set of each phone results in a probability distribution for which the entropy $E(h)$ can be calculated. Since the entropy is maximal for an equal distribution, the entropy $E(h)$ was used in the derivation of the weights w_h, which were defined by

$$w_h = \frac{E(h)}{\sum_{h' \in H} E(h')} \tag{6.13}$$

This definition ensures that the weights w_h sum to one. The performance metric for the individual phone classifiers was chosen to be the percentage of correctly classified data points. The overall performance metric is therefore an average correct classification rate.

Given test and training sets of pairs (d_t, v) of features d_t and states of v, the phone classifiers were built by estimating a single Gaussian with full covariance matrix on each of the states of v. The Gaussians for all of the classes were then used in the classification experiments to calculate the likelihood of each test data point under all the Gaussians and the data point was assigned to the class with the highest likelihood. It is, of course, possible to build better classifiers by using, for instance, polynomial softmax functions or GMM's to model the individual distributions. Since the main emphasis in these experiments was on the development of a useful indicating feature rather than a good classifier these other possibilities were avoided to give the simplest testing scenario.

6.2.2 Two-class classification experiments

This section discusses experimental results that were derived for the 3 different dimensionality reduction schemes introduced in section 6.1. Here the LDA and PCA transforms were both calculated for each phone separately and for all phones combined. The first method results in a model dependent indicating feature d, because the map $f(o) = d$ with which d is calculated depends on the phone, while the latter method gives a model independent feature. In addition, PCA transforms were calculated for different parts of the feature vector o, i.e. for the complete feature, the static components and the first and second order derivatives. These PCA transforms were then used to reduce the dimension of the respective part of feature o from one up to the original dimension of this feature part. In contrast to the PCA transforms, LDA transforms were only calculated for the full feature vector. This was done because restricting the LDA input to a subvector will, in general, decrease the objective function (6.9) and therefore result in less well separated classes. If, on the other hand, a subvector of o gave the best separation this would be reflected in an LDA transform of the full feature vector whose coefficients are zero for certain indices.

6.2.2.1 Phone independent dimensionality reduction

Table 6.1 gives the results for a two class problem where the dimension of o was reduced with the PCA transform. The first three columns contain the classification results for a PCA transform of the full feature vector from one dimension up to 39 dimensions. One can clearly see the monotonic increase in performance which is also illustrated in figure 6.6(a). This figure furthermore shows that there are 3 different regions over which the increase in performance has a similar shape. These regions correspond to the static components and the first and second order derivatives of feature o. This behaviour is a result of the fact that the static components

full feature vector						static		Δ		$\Delta\Delta$	
dim	% cor	dim	% cor	dim	% cor	dim	% cor	dim	% cor	dim	% cor
1	53.60	14	65.60	27	71.36	1	53.67	1	55.62	1	58.86
2	57.46	15	66.30	28	72.00	2	57.46	2	57.99	2	63.00
3	59.09	16	66.92	29	73.36	3	59.18	3	59.78	3	64.49
4	60.51	17	67.61	30	73.93	4	60.57	4	61.52	4	65.63
5	61.04	18	67.76	31	74.03	5	61.08	5	62.01	5	65.86
6	61.39	19	68.05	32	74.26	6	61.44	6	62.27	6	66.13
7	61.69	20	68.14	33	74.50	7	61.77	7	62.87	7	66.86
8	62.31	21	68.44	34	74.90	8	62.38	8	63.39	8	67.24
9	62.71	22	68.47	35	74.97	9	62.67	9	63.66	9	67.58
10	62.87	23	68.49	36	75.14	10	62.88	10	64.02	10	67.74
11	63.15	24	68.56	37	75.20	11	63.14	11	64.16	11	67.82
12	63.33	25	68.63	38	75.23	12	63.35	12	64.28	12	68.00
13	65.26	26	71.02	39	75.20	13	63.51	13	64.54	13	68.10

Table 6.1 *Two-class classification experiments with phone independent PCA transforms of the full feature vector, its static components and first (Δ) and second order ($\Delta\Delta$) derivatives.*

of o have the highest variance followed by the variance of the first and second order derivatives. The first rows of the PCA transform will therefore always project into the space of the static components while the rows in the middle and at the end of the PCA transform project into the vector-spaces spanned by the first and second order derivatives, respectively. This means that the PCA transform for feature o is essentially block-diagonal which is also illustrated by figure 6.5. The fourth, fifth and sixth columns in table 6.1 give the classification results that were obtained by restricting the vector o prior to calculating the PCA transform to its static components, first and second order derivatives, respectively. These transforms have been subsequently applied to the appropriate components of o to obtain the results in this table. These, again, show a steady increase in classification performance with increasing dimension of the PCA transform.a

Table 6.2, on the other hand, contains the results of the classification experiments using the squared Euclidean length and the LDA transform. Again, only one LDA transform was calculated for all the phones. In addition to the overall classification performance, table 6.2 also shows the classification results for vowels and consonants separately. It is interesting to note that the classifier performance in this table was higher on vowels than on consonants for Δ, $\Delta\Delta$ and LDA parameters. Since vowels are the main source of durational variation, this is a desirable effect. Especially for the squared Euclidean length of the second order derivatives, the performance on vowels was relatively high at 62.52%. However, a comparatively poor performance on consonants of 56.72% compensates for this advantage to give the overall performance of 59.46%. Also for PCA transforms this difference in performance on vowels and consonants was observed. However, to increase the readability these numbers were left out of table 6.1.

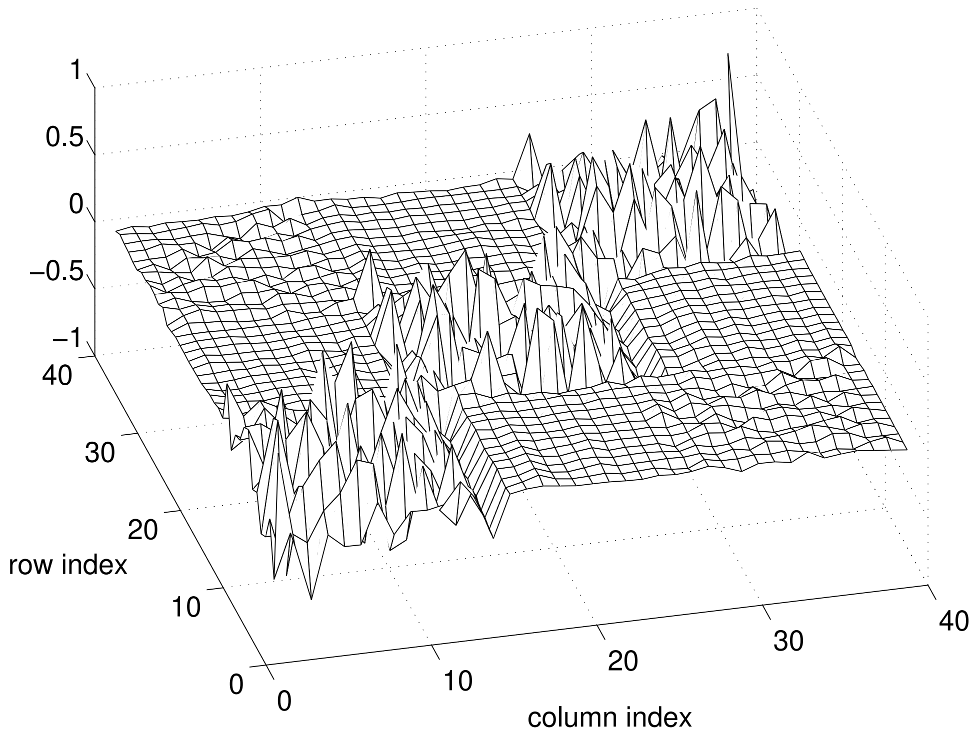


Figure 6.5 *Phone independent PCA transform.*

Figure 6.6(b) compares the various dimensionality reduction schemes for dimensions 1 to 13. In contrast to table 6.6(a), here the scale of the y axis which gives the classification performance is increased to show the differences between the methods more clearly. Since the squared Euclidean length and the LDA transform for a two class problem have a constant dimension of one they are indicated by straight lines. One can see from figure 6.6(b) that the best classification performance for more than one dimension is obtained by applying the PCA transform to the second order derivatives of o. Only in one dimension does the squared length of the second order derivative give a better classifier. The LDA transform results in a better performance than a one-dimensional PCA projection of the full vector, static and first order derivatives of o but is worse than the one-dimensional PCA projection of the second order derivatives. Figure 6.6(b) also shows that projecting the static components or the full feature vector onto the first 13 dimensions with a PCA transform gives essentially the same results. This is again a consequence of the block structure of the PCA transform illustrated in figure 6.5 which means that the PCA transform of the static components alone is essentially the first block in the PCA transform of the full feature vector.

Summarising the experiments with phone independent transforms, one can see that the one-dimensional projection with the best performance is the squared Euclidean length of the second order derivatives, which does especially well on vowels, while the PCA projection of the second order derivatives is the scheme with the best overall performance. Interestingly, calculating a single LDA transform for all the phones gives rather poor performance.

	squared length				LDA
	full	static	Δ	$\Delta\Delta$	
consonants	54.71	55.16	55.64	56.71	56.77
vowels	52.52	52.92	59.05	62.52	57.35
overall	53.67	54.11	57.25	59.46	57.05

Table 6.2 *Two-class classification experiments with phone independent LDA transform and squared Euclidean norm of the full feature vector, its static components and first (Δ) and second order ($\Delta\Delta$) derivatives. This table gives the average percentage of correctly classified data points.*

6.2.2.2 Phone dependent dimensionality reduction

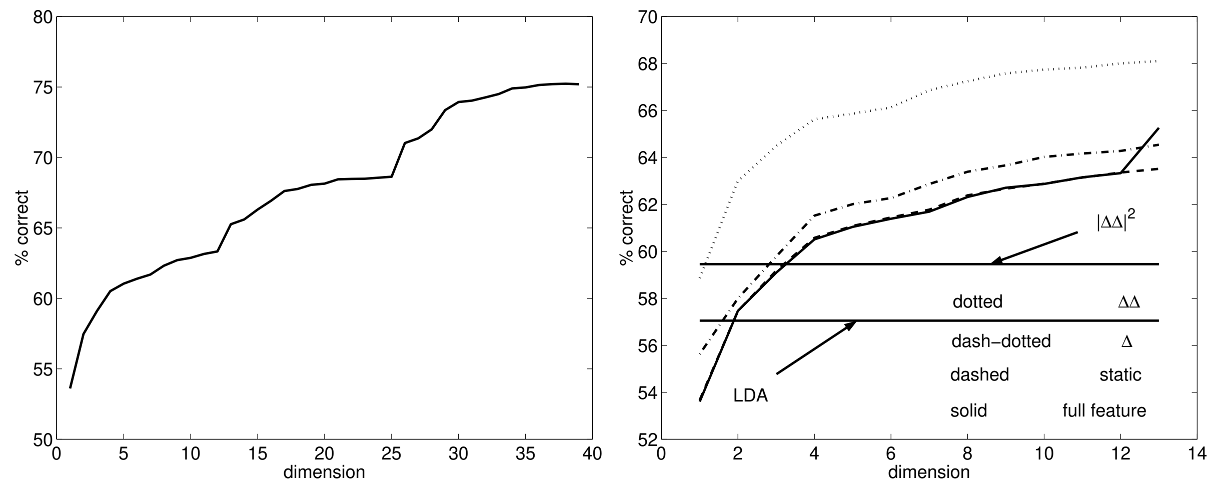
Table 6.3 contains the results for the classification experiments where PCA transforms were calculated for each phone individually. Figure 6.7(a) shows the effect of using phone dependent

full feature vector						static		Δ		$\Delta\Delta$	
dim	% cor	dim	% cor	dim	% cor	dim	% cor	dim	% cor	dim	% cor
1	57.38	14	66.13	27	73.30	1	57.36	1	55.55	1	61.01
2	59.14	15	66.86	28	73.88	2	59.12	2	58.65	2	63.95
3	60.68	16	67.56	29	74.13	3	60.38	3	60.60	3	64.96
4	61.49	17	67.82	30	74.32	4	61.44	4	61.68	4	65.94
5	61.95	18	68.20	31	74.52	5	61.88	5	62.25	5	66.37
6	62.21	19	68.37	32	74.70	6	62.12	6	62.63	6	66.95
7	62.48	20	68.68	33	74.86	7	62.37	7	63.04	7	67.24
8	62.68	21	68.81	34	75.08	8	62.56	8	63.56	8	67.71
9	62.90	22	68.84	35	75.11	9	62.71	9	63.82	9	67.98
10	63.16	23	68.94	36	75.21	10	63.02	10	64.27	10	68.31
11	63.40	24	69.37	37	75.26	11	63.17	11	64.45	11	68.59
12	64.00	25	71.11	38	75.24	12	63.38	12	64.61	12	68.77
13	65.06	26	72.47	39	75.20	13	63.51	13	64.54	13	68.78

Table 6.3 *Two-class classification experiments with phone dependent PCA transforms of the full feature vector, its static components and first (Δ) and second order ($\Delta\Delta$) derivatives.*

transforms in comparison to phone independent transforms. The dashed line in this figure is therefore the same curve as in figure 6.6(a). One can see from figure 6.7(a) that using phone dependent transforms gives better classification performance for almost all dimensions. It is therefore possible to reduce the dimensionality of d even further than with a phone independent transform, while still retaining good classification performance. As can be seen from table 6.3 and figure 6.7(a), the differences for phone dependent and phone independent classification results vanish for the maximal dimensionality of d.

The classification results for the first 13 dimensions of all the phone dependent dimensional-

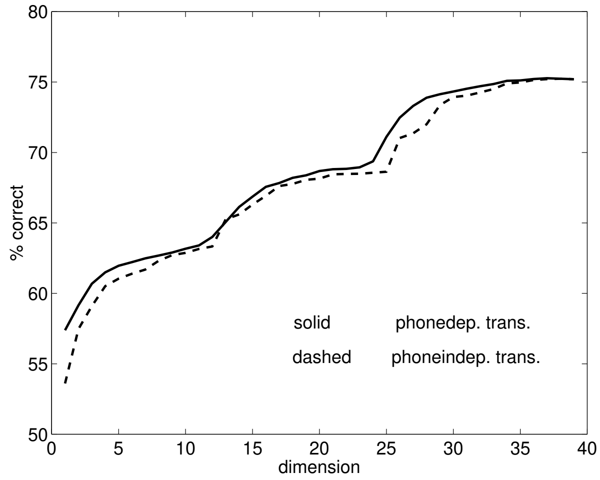


(a) Classification performance of a phone independent PCA transform applied to the full 39 dimensional feature vector.

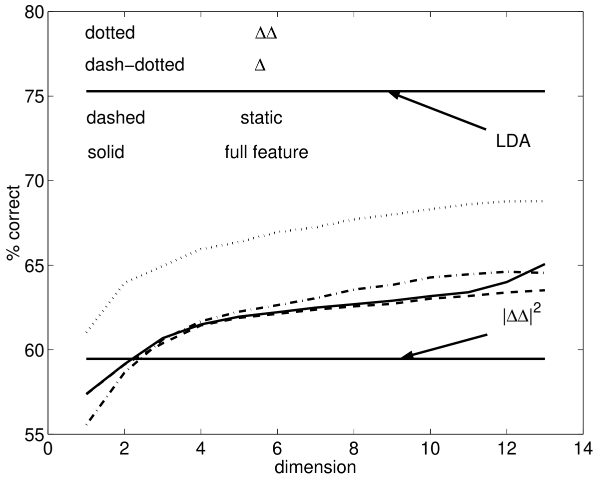
(b) Classification performance of the various phone independent dimensionality reduction schemes for dimensions 1 to 13.

Figure 6.6 *Relationship between the two-class classification performance of the phone independent dimensionality reduction schemes and the dimension of the resulting feature. Since $|\Delta\Delta|^2$ and the LDA transform have a constant dimension of 1 they are indicated by straight lines in figure 6.6(b). The other curves in this figure correspond to PCA transforms of the full feature vector and its various components.*

ity reduction schemes are summarised in figure 6.7(b) which is the equivalent to figure 6.6(b). Again, transforming the second order derivatives of feature vector o gives the best results of all the PCA transforms. In this case, the one dimensional projection is even better than the squared Euclidean length of the second order derivatives which has the same performance as before since it is a phone independent dimensionality reduction method. However, the biggest difference in performance between phone dependent and phone independent transforms can be observed for the LDA transform. While the classification performance for a phone independent LDA transform was 57.05% in the previous experiments this number increases to 75.29% for phone dependent LDA transforms which is the same result as for the full 39 dimensional feature vector o. This shows that it is possible to reduce the dimension of the standard PLP feature o to one without losing discriminative information for the distinction between fast and slow classes of phones. This result also illustrates that, in the classification problems considered, LDA transforms are less robust than PCA transforms to a change from a phone dependent to a phone independent feature extraction paradigm.



(a) Comparison between the classification performance of a phone dependent and a phone independent PCA transform applied to the full feature vector.



(b) Classification performance of the various phone dependent dimensionality reduction schemes for dimensions 1 to 13.

Figure 6.7 *Relationship between the two-class classification performance of the phone dependent dimensionality reduction schemes and the dimension of the resulting feature. Since $|\Delta\Delta|^2$ and the LDA transform have a constant dimension of 1 they are indicated by straight lines in figure 6.7(b). The other curves in this figure correspond to PCA transforms of the full feature vector and its various components.*

6.2.3 Three-class classification results

This section presents classification experiments with 3 classes using, as in the previous section, PCA and LDA transforms of the 39 dimensional PLP feature vector and the squared length of some of its subvectors. The classes in these experiments were again derived with the method described in section 6.2.1 for each of the phones in the phone set in appendix A.

6.2.3.1 Phone independent dimensionality reduction

The first set of experiments used phone independent dimensionality reduction schemes to obtain the indicating feature. Since the calculation of the PCA transforms is independent of the class membership of the training data the phone independent as well as the phone dependent PCA transforms in this section are the same as in section 6.2.2.

The results of the 3 class classification experiments for phone independent PCA transforms applied to the various components of the 39 dimensional PLP feature are summarised in table 6.4. This table has the same structure as tables 6.1 and 6.3, which means that it gives the percent of correctly classified frames for 1 to 39 dimensional PCA transforms applied to the full 39 dimensional feature vector and 1 to 13 dimensional PCA transforms applied to static components as well as the first and second order derivatives of the PLP feature o. Compared to table 6.1, the numbers in table 6.4 are considerably smaller due to the higher complexity of the classification task. As in the previous section, the relationship between performance and

full feature vector						static		Δ		ΔΔ	
dim	% cor	dim	% cor	dim	% cor	dim	% cor	dim	% cor	dim	% cor
1	37.06	14	49.31	27	55.70	1	37.46	1	38.04	1	41.51
2	40.48	15	50.17	28	56.70	2	40.72	2	41.06	2	46.71
3	42.09	16	50.79	29	58.34	3	42.34	3	42.92	3	48.72
4	43.67	17	51.46	30	59.14	4	43.92	4	44.54	4	50.13
5	44.04	18	51.75	31	59.44	5	44.37	5	44.82	5	50.58
6	44.60	19	52.02	32	59.71	6	44.82	6	45.46	6	50.83
7	44.92	20	52.10	33	59.91	7	45.25	7	46.07	7	51.96
8	45.55	21	52.37	34	60.29	8	45.84	8	46.60	8	52.43
9	45.85	22	52.39	35	60.46	9	46.22	9	46.86	9	52.72
10	46.15	23	52.57	36	60.62	10	46.47	10	47.43	10	52.94
11	46.54	24	52.62	37	60.70	11	46.90	11	47.53	11	52.94
12	46.98	25	52.66	38	60.86	12	47.37	12	47.68	12	53.65
13	48.93	26	55.42	39	60.85	13	47.61	13	48.04	13	53.75

Table 6.4 *Three-class classification experiments with phone independent PCA transforms of the full feature vector, its static components and first (Δ) and second order (ΔΔ) derivatives.*

dimensionality of the PCA projection applied to the full feature vector exhibits a distinctive step-shaped form as can be seen in figure 6.8(a) and is monotonic with increasing dimension of the projection. This is also the case for PCA transforms specific to the static components, first and second order derivatives of o.

Table 6.5 gives the classification performance for the squared Euclidean length of various subvectors of feature o as well as for the phone independent LDA transforms. As previously the overall performance and the performance on vowels and consonants separately are both included in this table. Since the classification problem deals with three different classes the dimension of the LDA transforms is two. This, of course, also holds for the phone dependent LDA transforms which will be investigated later.

Table 6.5 shows that the performance of the squared Euclidean norm exhibits similar trends as in the case of the 2 class problems in table 6.2. Again the performance on vowels is better

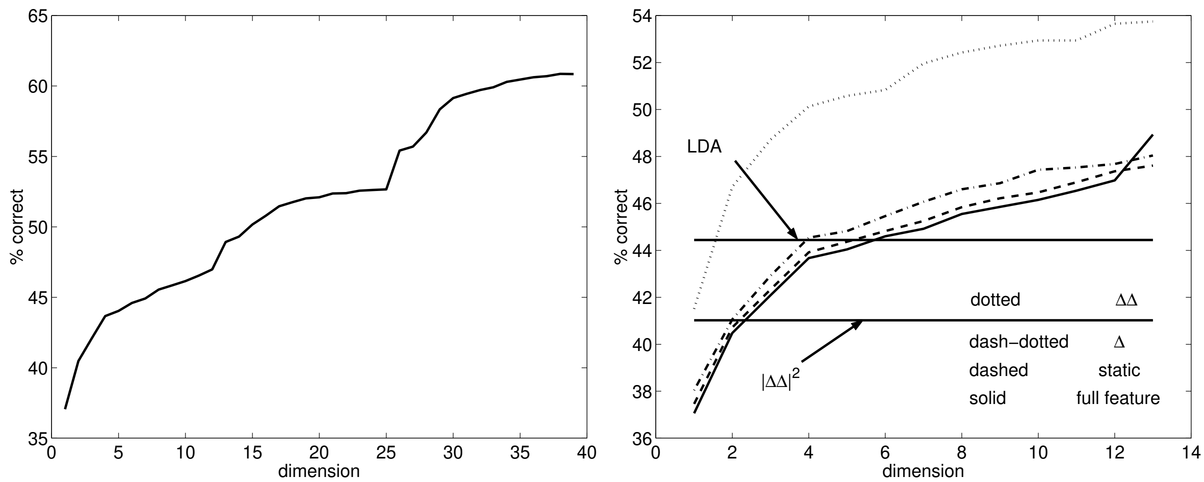
	squared length				LDA
	full	static	Δ	ΔΔ	
consonants	38.01	38.70	37.46	38.23	44.20
vowels	36.46	37.37	39.72	44.10	44.70
overall	37.27	38.07	38.53	41.02	44.44

Table 6.5 *Three-class classification experiments with phone independent LDA transform and squared Euclidean norm of the full feature vector, its static components and first (Δ) and second order (ΔΔ) derivatives. This table gives the average percentage of correctly classified data points.*

than on consonants for the squared length of the first and second order derivatives. This time, however, the performance of the model independent LDA transform is better than the squared length of the second order derivatives. This performance improvement can be attributed to the higher dimension of the LDA transform which, as mentioned above is, in this case, two.

It is interesting to note that, although the overall classification performance of the LDA derived indicating features is considerably higher at 44.44% correctly classified frames as compared to 41.02% for the squared length of the second order derivatives, the difference on vowels is much smaller at only 0.6% absolute. The main difference in performance occurs on the consonants where the squared Euclidean length gives considerably worse results by about 6% absolute.

The performance of the LDA, PCA and squared length dimensionality reduction methods for the first 13 PCA dimensions is compared in figure 6.8(b). Again, since the squared Euclidean length of the second order derivatives has a constant dimension of one as well as the now two dimensional 3 class LDA transforms, the results for these two dimensionality reduction schemes is indicated by straight lines which run parallel to the dimension axis in figure 6.8(b). As in figure 6.6(b) in section 6.2.2.2, one can see in figure 6.8(b) that the performance of the PCA transform that was calculated on the static features only results in more or less the same performance as the PCA transform of the full feature on its first 13 dimensions. Also the difference in performance between the first order derivative PCA transforms and the static PCA transforms is about the same.



(a) Classification performance of a phone independent PCA transform applied to the full 39 dimensional feature vector.

(b) Classification performance of the various phone independent dimensionality reduction schemes for dimensions 1 to 13.

Figure 6.8 *Relationship between the three-class classification performance of the phone independent dimensionality reduction schemes and the dimension of the resulting feature. Since $|\Delta\Delta|^2$ and the LDA transform have a constant dimension of 1 and 2, respectively, they are indicated by straight lines in figure 6.8(b). The other curves in this figure correspond to PCA transforms of the full feature vector and its various components.*

Figure 6.8(b) shows that the best phone independent reduction scheme for a 3 class problem is the PCA transform of the second order derivatives. In contrast to the 2 class case, here the one dimensional PCA projection of the second order derivatives gives the best one dimensional feature with 41.51% correctly classified frames on average, as compared to 41.02% for the squared length of the second order derivative. The two dimensional PCA projection is also superior to the phone independent 3 class LDA transform. Here the PCA transform classifies on average 46.71% of the frames correctly, while the LDA transforms leads to a correct classification in 44.2% of the cases. This shows therefore that PCA transforms yield the best phone independent dimensionality reduction scheme in the case of classification problems with 3 classes.

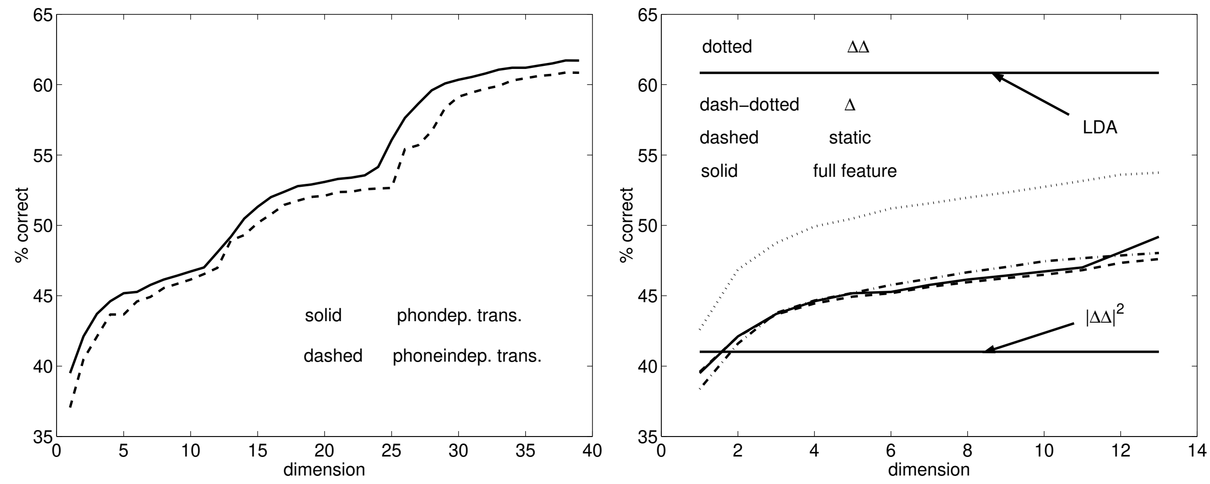
6.2.3.2 Phone dependent dimensionality reduction

The experiments in this section applied phone dependent transforms to reduce the dimension of feature o and evaluated their performance on the same 3 class problems as in the previous section. As mentioned before, the PCA transforms are the same as in the 2 class experiments in section 6.2.2 since the calculation of these transforms does not take class membership into consideration. The results for the PCA transforms with varying dimension applied to different components of the feature vector o are summarised in table 6.6.

full feature vector						static		Δ		$\Delta\Delta$	
dim	% cor	dim	% cor	dim	% cor	dim	% cor	dim	% cor	dim	% cor
1	39.50	14	50.48	27	58.65	1	39.59	1	38.37	1	42.59
2	42.11	15	51.32	28	59.60	2	42.10	2	41.60	2	46.86
3	43.71	16	52.01	29	60.09	3	43.67	3	43.78	3	48.75
4	44.61	17	52.39	30	60.35	4	44.45	4	44.66	4	49.92
5	45.18	18	52.79	31	60.55	5	44.94	5	45.20	5	50.49
6	45.27	19	52.90	32	60.79	6	45.18	6	45.77	6	51.20
7	45.77	20	53.09	33	61.07	7	45.63	7	46.21	7	51.56
8	46.15	21	53.30	34	61.20	8	45.96	8	46.67	8	51.97
9	46.43	22	53.39	35	61.20	9	46.25	9	47.04	9	52.33
10	46.73	23	53.56	36	61.36	10	46.49	10	47.45	10	52.74
11	47.01	24	54.15	37	61.51	11	46.82	11	47.67	11	53.16
12	48.09	25	56.06	38	61.72	12	47.34	12	47.86	12	53.60
13	49.19	26	57.64	39	61.71	13	47.61	13	48.04	13	53.75

Table 6.6 *Three-class classification experiments with phone dependent PCA transforms of the full feature vector, its static components and first (Δ) and second order ($\Delta\Delta$) derivatives.*

Again, the results for phone dependent and phone independent PCA transforms are compared in figure 6.9(a) which shows that, similar to the last section, phone independent PCA transforms reach the same level of classification performance as phone independent PCA transforms already at lower dimensions of the indicating feature. The differences in classification performance between these two approaches are approximately the same for both the 2 and 3 class problems.



(a) Comparison between the classification performance of a phone dependent and a phone independent PCA transform applied to the full feature vector.

(b) Classification performance of the various phone dependent dimensionality reduction schemes for dimensions 1 to 13.

Figure 6.9 *Relationship between the three-class classification performance of the phone dependent dimensionality reduction schemes and the dimension of the resulting feature. Since $|\Delta\Delta|^2$ and the LDA transform have a constant dimension of 1 and 2, respectively, they are indicated by straight lines in figure 6.9(b). The other curves in this figure correspond to PCA transforms of the full feature vector and its various components.*

As previously, figure 6.9(b) compares the classification performance for all the phone dependent dimensionality reduction schemes for the first 13 dimensions. Since both the squared length of the second order derivatives and the 3 class phone dependent LDA transforms have fixed dimension their performance is indicated by straight lines parallel to the dimension axis. Figure 6.9(b) shows, that as for the two class problems, phone dependent LDA outperforms all the other methods giving a percentage of correctly classified frames of 60.84%. This is almost as good as using the full feature vector and is a considerably better result than the one obtained by using the 2 dimensional PCA projection of the second order derivatives which gives an average error rate of 46.86%. It is interesting to note, that the increase in classification performance in going from the two dimensional phone independent PCA transforms to the two dimensional phone dependent PCA transforms only gives a small improvement of 0.15% absolute. It can also be seen from figure 6.9(b) that the squared Euclidean length of the second order derivatives performs worse than the one dimensional phone dependent PCA transform of the second order derivatives. Since the squared length of the second order derivative is an inherently model independent dimensionality reduction scheme the lower straight line in figure 6.9(b) is located at the same level as in figure 6.8(b).

6.3 Summary

This chapter investigated the performance of different dimensionality reduction schemes with the aim to reduce the full 39 dimensional PLP feature vector to a low dimensional indicating feature that still retains most of the discriminatory information which relates to the distinction between fast and slow classes of phones. For this purpose, LDA and PCA transforms where calculated for all the phones simultaneously which gave rise to phone independent dimensionality reduction schemes and for all the phones individually which resulted in phone dependent dimensionality reduction schemes. In addition to these linear methods, the squared Euclidean length of the full feature vector as well as of some of its subcomponents was used as an indicating feature. Since the latter does not depend on any transforms that need to be estimated it is inherently a phone independent indicating feature.

The usefulness of these different schemes was evaluated by performing classification experiments on two and three class problems which were constructed by splitting the instances of phones into classes of approximately equal size depending on their durational statistics. The classifiers were ML classifiers which used a single Gaussian with full covariance for each class. It should be pointed out that the results derived in this way using PCA transforms are specific to the particular feature vector used here since PCA transforms are not invariant under rescaling.

The experiments showed that for two classes the squared length of the second order derivative of feature o gave the best phone independent one dimensional classifier while the PCA transform of the second order derivatives of feature o gave the best one and two dimensional phone independent classifier for the 3 class problems. In both cases phone dependent LDA transforms gave the best phone dependent classification with almost no loss in performance as compared to the full feature vector o. For this reason, phone dependent LDA transforms will be used to derive the indicating feature in the recognition experiments in the next chapter. In addition, the squared length of the second order derivatives will also be used. The difference in performance between SBME-HMM's with these two types of indicating features will also highlight the influence on the model performance that results from choosing a model dependent or a model independent indicating feature.

7

Recognition Experiments

This section describes a number of recognition experiments that were carried out with the SBME-HMM whose theory was developed in chapter 5. The models in these experiments used the topologies presented in section 5.6 and were initialised with the methods in sections 5.5.1 and 5.5.2. The phone dependent LDA transforms which were estimated in the 2 class and 3 class experiments in sections 6.2.2 and 6.2.3 were used in the derivation of the indicating feature d and the squared Euclidean length of the second order derivatives was also studied in the experiments. The output pdf's $b(d|s)$ for these indicating features were therefore modelled by single Gaussians and Gamma densities, respectively, which can be motivated by the same arguments as in section 5.5.1. There it was shown that the squared Euclidean length of a random variable that is distributed according to a Gaussian mixture model can be approximated by a Gamma density while the linear transform of such a random variable is again distributed according to a Gaussian mixture. For convenience, the squared Euclidean norm of the second order derivative will be abbreviated by SLSD in the following.

Section 7.1 briefly outlines the type of evaluation metric that will later be applied in recognition experiments while section 7.2 discusses a significance test which is commonly used to determine whether differences in the output of various recognition systems are due to chance.

Section 7.3 gives a description of the data base on which the recognition experiments will be performed and section 7.4 discusses the environment in which the experiments were conducted.

Section 7.5 gives the results of experiments with a binary hidden variable v. After describing different baseline systems in section 7.5.1, section 7.5.2 evaluates the performance of an SBME-HMM which has been initialised with the RTI method. Section 7.5.3, on the other hand, presents results for a system initialised with MSI.

This is followed in section 7.6 by a discussion of experiments with a hidden variable v with 3 states. Again RTI and MSI initialised models are investigated in sections 7.6.1 and 7.6.2 and the computational cost of the full Newton algorithm with line search and backtracking to update the softmax parameters without a restriction on the size of the training buffers is analysed.

To conclude, section 7.7 summarises the results presented in this chapter.

7.1 Evaluation metric

The metric that is most commonly used to evaluate the performance of speech recognition systems is the word error rate. This number is calculated by summing over all the errors in the recogniser output and dividing by the total number of words that are in the reference transcription. Three different types of errors are identified whose definitions are given below.

Insertions An insertion occurs if the hypothesis contains a number of words between two correctly recognised words which does not correspond to a word in the reference transcription.

Deletions A deletion is the opposite of an insertion and occurs if the hypothesis does not contain a word that appears in the reference transcription.

Substitutions This error occurs if a word of the transcription is replaced by another word in the hypothesis.

These types of errors are visualised in figure 7.1, where the first line gives the correct word sequence and the lines below contain the word hypotheses of a speech recogniser. Abbreviating

cor:	this is speech		
hyp:	this is <u>a</u> speech	→	insertion
hyp:	this _ speech	→	deletion
hyp:	this is <u>peach</u>	→	substitution

Figure 7.1 *Types of recognition errors.*

the number of insertions, deletions and substitutions by I, D and S, respectively, the word error rate is given by the following expression

$$100 \times \frac{I+S+D}{N} \tag{7.1}$$

where N is the total number of words in the reference transcription. It is interesting to note that the expression (7.1) can become larger than 100% if the number of insertions is high.

7.2 Significance tests

In order to determine whether the differences in performance between various recognition systems are due to chance, statistical significance tests will be performed with the matched pair sentence segment word error (MPSSWE) test. This test has been proposed in (Gillick and Cox, 1989) and has been implemented as part of the NIST scoring package (Pallett et al., 1990). The MPSSWE test looks for segments in the hypothesis of two recognition systems whose errors can be assumed to be statistically independent. If N_1^i and N_2^i are the errors of system 1 and 2 on segment i then, because of the choice of the segments, the $N_1^i - N_2^i$ are independent identically distributed samples of a distribution which, in this case, is assumed to be a single Gaussian. The objective of the significance test is now to determine whether the null-hypothesis, that the mean of this distribution is zero, can be accepted. If this is the case then the differences in performance of the two systems will not be considered to be statistically significant. If the assumption that the $N_1^i - N_2^i$ are samples of a Gaussian with zero mean is correct, then it is well known (Lass and Gottlieb, 1971; Kendall and Stuart, 1973c) that the following value is distributed according to a Student-t distribution

$$T = \frac{\bar{m}}{\frac{S}{\sqrt{n}}} \tag{7.2}$$

where $\bar{m}$ and S are the estimated mean and the unbiased variance estimate of the samples $N_1^i - N_2^i$, respectively, i.e.

$$\bar{m} = \frac{\sum_{i=1}^{n} N_1^i - N_2^i}{n} \tag{7.3}$$

$$S = \frac{\sum_{i=1}^{n}(\bar{m} - (N_1^i - N_2^i))^2}{(n-1)} \tag{7.4}$$

where n is the total number of samples. Since the Student-t distribution converges to a Gaussian with zero mean and unit variance for $n \to \infty$, the Student-t distribution can be replaced by the normalised Gaussian for large n. Now the probability of observing the value T (7.2) under the null hypothesis is calculated as follows

$$2\int_{|T|}^{\infty} e^{-x^2/2}\mathbf{d}x \tag{7.5}$$

This is a two-tailed test since the absolute value of T rather than T itself is considered in (7.5). If the probability in (7.5) lies below a certain threshold the null-hypothesis will be rejected and the differences between the two recognition systems will be considered to be statistically significant. A typical value below which differences are considered to be significant is 0.05. This is, for instance, the confidence level which is used by NIST in the DARPA evaluations.

7.3 The 1997 Broadcast News task

The recognition experiments in the following sections were conducted on the 1997 Broadcast News task (Pallett et al., 1998) which consists of non-overlapping training, test and development test sets. The training set is made up of approximately 100 hours of manually transcribed news broadcasts and the test set is a collection of 158 variable length excerpts from a number of American TV and radio broadcasts which have been concatenated into a single file. The test set has a total duration of about 3 hours and contains 32 834 words. Only the training and test sets were used in the experiments in this chapter.

The Broadcast News task consists of different types of data which have been divided into 7 categories, denoted by F0,F1, . . . ,F5 and FX, respectively. Table 7.1 gives the definition of these so-called F-conditions and the number of words in each of them for the 1997 Broadcast News test set. According to table 7.1, planned speech is the most prominent condition in the test set

condition	description	nr. words
F0	planned, low noise broadcast speech	13197
F1	spontaneous broadcast speech	6566
F2	speech over telephone channels	4882
F3	speech in the presence of background music	1571
F4	speech under degraded acoustic conditions	3350
F5	speech from non-native speakers	669
FX	all other speech	2599
total		32 834

Table 7.1 *Definition and size of F-conditions in the 1997 Broadcast News task.*

which constitutes approximately 40% of the test set. This is followed by spontaneous speech with a share of about 20%. The relatively high proportion of the F1 condition makes the 1997 BN task a good starting point for investigating models designed for spontaneous speech.

Each segment in the test set is labelled with exactly one of the F-conditions in table 7.1 although some of them could have been labelled with more than one. Telephone speech, for instance, is very often spontaneous and could therefore be labelled with F1 as well as F2. However, a single condition is assigned to each speech segment resulting in non-overlapping sets of segments.

As part of the development of the CU-HTK system (Woodland et al., 1998) for the 1997 Broadcast News evaluation, parts of the manual transcription of the training data were corrected. This was done with the help of a forced alignment which highlighted segments exhibiting a mismatch between acoustic models and manual transcription. Each of these segments, whose alignment either produced a low average likelihood score or did not complete at all, was checked manually and either corrected or removed from the training data. These corrected transcriptions were used to train the models in the following experiments.

In contrast to the 1996 Broadcast News task (Pallett et al., 1997), the 1997 task was an unpartitioned evaluation. This means that instead of a number of acoustically homogeneous segments the test data were provided in one unsegmented 3 hour audio file which the participants had to split into segments themselves. The segments which were used in the CU-HTK system (Hain et al., 1998) form the basis for the recognition experiments that will be described in the following sections.

The dictionary of the recogniser contained about 65000 words some of which had multiple pronunciations. No pronunciation probabilities were used in the experiments.

7.4 Experimental set-up

The experiments were performed by rescoring tri-gram lattices which were generated as part of the CU-HTK system in the 1997 Broadcast News evaluation. The models used were gender independent cross-word triphone models with Gaussian mixtures models for the output pdf's for feature vector o. The mixture components had diagonal covariance matrices. The word insertion penalty and the language model scaling factor were constant in all the experiments.

The recognition experiments were run on a cluster of 19 LINUX machines with Pentium 3 processors of clock speeds between 700MHz and 1GHz. At most 11 of these machines were used to rescore the lattices in parallel and the CPU times were subsequently combined by weighting the CPU time of each process by the ratio between the machine's clock speed and a clock speed of 1GHz. These were then summed up to give the approximate CPU time that would have been used on a 1GHz machine. This value was used to calculate the real time factor of the recogniser by dividing through the total duration of the test set.

The training of the models involved two separate stages. In the first stage 12 processes were run in parallel on the same LINUX cluster as above to gather the statistics of the model parameters on the training data. The 12 resulting accumulators which had a total size of between 500MB and 2GB were temporarily stored on disk. In the second stage these accumulators were then combined to obtain the overall statistics of the model components which completed the expectation step of the EM algorithm. Subsequently, these statistics were used to update the model parameters with the maximisation step of the EM algorithm. In contrast to the first stage, the second stage of the parameter update was not performed in parallel but on one model after another in either a single process or a sequence of processes.

7.5 Experiments with 2 hidden states

In the first set of recognition experiments the hidden variable v was assumed to have two states and consequently there were two output pdf's $b(o|s, v)$ for each state s. The only exception were the silence and short pause models which were not assumed to exhibit fast and slow dynamics and whose hidden variables v had therefore only one state. In this case, the posterior probabilities $p(v|d, s)$ were set to one and the SBME-HMM's for these phonetic units were therefore equivalent to a standard HMM with an additional stream d. This section will discuss the performance of such SBME-HMM's using different initialisation methods and recognition topologies.

First, section 7.5.1 will give an overview of the different baseline systems which the models in later sections will be compared to.

Section 7.5.2 and 7.5.3 discuss recognition experiments with models that were initialised with the relabelled training initialisation and the median split method from sections 5.5.1 and 5.5.2, respectively. These models used a 12 mixture baseline system for initialisation and since they had two output pdf's $b(o|s, v)$ per state s, with the exception of short pause and silence models, the number of parameters in these SBME-HMM's was comparable to that in a 24 mixture baseline system.

The size of the training data buffers for the softmax functions in these experiments was restricted to 2000 samples. This resulted in very rapid model updates as compared to using the complete training data. In contrast to an HMM of roughly the same size for which the parameter update following the accumulation of the training statistics was completed in about 70 CPU seconds, the parameter update for the SBME-HMM models took about 200 CPU seconds. While this is an almost 3 fold increase in computational cost the overall cost is still very low and a detailed analysis of convergence properties in the softmax parameter updates has therefore been postponed to section 7.6 where higher dimensional indicating features and unrestricted sample sizes will be used.

7.5.1 Baseline systems and preliminary experiments

This section presents recognition experiments with three different model sets. The first model set has 12 Gaussian mixture components per state and 24 mixtures for the states of the silence and short pause models. From this model two others were derived which use approximately twice as many parameters as the 12 mixture system. For the first model the HMM's in the 12 mixture system where initially duplicated and then trained on relabelled training data to derive speaking rate dependent models. In comparison to chapter 6, here the training data were not relabelled depending on the durational statistics of monophones. Instead the decision if a phonetic unit was slow or fast was based on the durational statistics of each triphone. This had to be done, because applying the monophone thresholds to the triphones in the training set resulted occasionally in very biased classes, some of which were even empty. To increase the number of observations of each triphone the statistics of models which shared the same physical model were combined. The fast/slow instances of each HMM that were trained on the resulting phonetic transcriptions were then recombined in a parallel topology with 6 states per HMM as in (Iyer et al., 1998) to give a single HMM for each phonetic unit. No further EM iterations were performed on this model.

Another model was derived from the 12 mixture baseline system by gradually increasing the number of mixture components. This model was trained 4 iterations at a time after which the number of mixtures for all the models were increased by two with the model split command "MU" which is implemented by the HHEd program in HTK (Young et al., 2000). The total number of iterations to go from the 12 mixture system to the 24 mixture system was therefore 24. The recognition performance for the three different models can be found in table 7.2. The

	Overall	F0	F1	F2	F3	F4	F5	FX	xRT
12 mix base	21.8	12.4	20.5	31.5	31.5	24.3	27.5	43.9	2.69
2 parallel paths	21.8	12.5	20.5	30.5	32.8	25.0	26.9	43.7	3.81
24 mix base	21.3	12.2	20.3	29.9	32.0	24.4	25.7	41.7	3.94

Table 7.2 *Recognition performance of base line systems.*

first 8 columns contain the overall word error rate as well as the error rates for the different

F-conditions and the last column gives the total CPU time used expressed by the realtime factor of the system. The first row in table 7.2 contains the results for the 12 mixture base-line system. This is a gender independent model set that was developed for the CU-HTK system in the 1997 Broadcast news evaluation (Woodland et al., 1999). This model set contains 6684 states which are shared among a total of about 95 000 HMM's. Since each state has roughly 950 parameters, the total number of parameters in the model set was approximately 6.3m. As mentioned before, the phonetic units of this model are cross-word triphones and the acoustic features are, as in chapter 6, PLP vectors with first and second order derivatives. Since this model does not distinguish between telephone and wideband speech, its performance is slightly worse than that of the gender independent first pass of the CU-HTK system which was reported to give a word error rate of 21.4%.

The second row in table 7.2 gives the results for the parallel model which was described above. It is perhaps surprising that the performance of this model is the same as for the 12 mixture baseline system, although the parallel model has twice as many parameters. However, this is most likely the result of the fact that this parallel model was derived by training its branches individually without performing any additional EM iterations on the combined model. This result is therefore worse than the results reported in (Iyer et al., 1998) which compared a parallel model to a non-parallel model with the same number of parameters. As compared to the parallel model here, the experiments in (Iyer et al., 1998) showed no difference in performance between a parallel and a non-parallel model with the same number of parameters, only when state tying was introduced between states of different branches in the parallel topology could a slight reduction in error rate be achieved.

Another experiment, this time starting from a 6 mixture baseline system, also showed that training a parallel model on relabelled training data without subsequent EM updates resulted in suboptimal performance. Although the parallel model, with a WER of 22.9%, gave an improvement over the baseline system which had a WER of 23.8%, the performance of this system is worse compared to the WER of 21.8% for the 12 mixture baseline system in table 7.2.

The model with 24 mixture components which was derived from the 12 mixture model by mixture splitting gives the best performance in table 7.2. It is interesting to note that the relationship between the number of parameters in the model and the WER was not monotonic. In fact, by increasing the number of mixtures from 18 to 20 with HHEd resulted in a slight performance degradation from a WER of 21.4% to a WER of 21.5%.

7.5.2 Relabelled training initialisation

The experiments in this section were performed with models that were initialised with the relabelled training initialisation described in section 5.5.1. The rate dependent models, which are necessary for this type of initialisation, were the same that were used to create the 12 mixture parallel model baseline system in the last section.

To see the effect of training different system components the training of the models was divided into steps of 2 iterations each alternating between updating the $b(d|s)$ and softmax parameters and updating the parameters of the output pdf's $b(o|s,v)$. Table 7.3 gives the word error rates at each stage of the training using the squared length of the second order derivative as an indicating feature. Here error rates are reported for the three different model topologies mentioned in section 5.6, namely the full and parallel topology and the topology where the variable v has been factored out. The rows corresponding to these topologies are denoted by "full", "par" and "fact", respectively. In addition, table 7.3 also shows the CPU usage, expressed by the realtime factor, in the column denoted by "xRT" for each of the recognition runs. Finally, the last column of this table contains the average log likelihood per frame on the training data of the factorised model after each parameter update. The average log likelihood for the initialised model, which is not given in this table, was -67.60. As mentioned in section 5.6, the parameters of the parallel model were derived from a trained factorised system.

updated	top	Overall	F0	F1	F2	F3	F4	F5	FX	xRT	like
$b(d\|s)$	fact	21.7	12.6	20.5	30.7	30.9	24.4	27.5	42.6	3.74	
+	full	21.7	12.6	20.5	31.0	30.9	24.5	27.5	42.9	4.59	-67.19
soft	par	22.0	12.7	20.7	31.4	32.0	25.3	26.3	44.0	4.40	
	fact	21.3	12.5	19.9	30.6	31.6	23.6	26.9	42.0	3.75	
$b(o\|s,v)$	full	21.4	12.5	19.8	30.7	31.9	23.9	25.9	42.2	4.33	-66.87
	par	22.0	12.6	20.5	31.5	32.7	25.0	26.9	43.6	4.43	
$b(d\|s)$	fact	21.4	12.6	20.0	30.6	31.7	23.7	27.1	42.1	3.30	
+	full	21.5	12.6	20.0	30.9	31.9	23.6	26.9	42.5	4.46	-66.86
soft	par	22.2	12.7	20.8	32.2	32.7	25.2	26.8	44.0	4.38	
	fact	21.3	12.6	19.7	30.1	32.0	23.6	25.9	42.4	3.51	
$b(o\|s,v)$	full	21.4	12.7	19.8	30.2	32.1	23.6	25.9	42.4	4.30	-66.74
	par	22.2	12.9	20.5	31.9	32.1	25.0	26.5	44.9	4.34	

Table 7.3 *Recognition performance for an RTI initialised SBME-HMM with an SLSD indicating feature and 2 v-states.*

Table 7.3 shows that the likelihood on the training data increases with each iteration which results, on the other hand, in a reduction in error rate. The biggest increase in log likelihood occurs in the first two iterations on the softmax and $b(d|s)$ parameters where the initial log likelihood of -67.60 increases to -67.19 which corresponds to an increase in likelihood by a factor of 1.5. In the following two iterations on the $b(o|s,v)$ parameters the likelihood increase is smaller but still noticeable at a factor of 1.3. These two iterations are also important in terms of the performance of the system which increases by 0.4% and 0.3% absolute for the factorised and full topology, respectively. For the parallel topology the error rate stays constant at 22.0% which is a much worse performance than that of the other two topologies. Enforcing a slowly varying hidden variable for a model whose parameters were derived from a trained factorised model therefore seems to be detrimental.

It is interesting to note, that the main increase in performance occurs on the spontaneous speech condition while performance on F0 remains more or less constant over all the iterations. One can also see from table 7.3 that performance of the full topology model was consistently worse, although not by much, than that of the factorised model while the increased size of the search space meant that recognition runs for the full topology were slower by about a factor of about 1.2. The third parameter update in table 7.3 also shows that there is very little gain in likelihood in the second two iterations on the softmax and $b(d|s)$ parameters and that their likelihoods are already saturated.

Table 7.4, again, gives the results for experiments with a binary hidden variable v. This time, however, the indicating feature was derived from feature o by applying the phone dependent LDA transforms that were estimated in the experiments in section 6.2.2.2. Again the parameters of the model were updated in steps of two iterations each on either the softmax and $b(d|s)$ parameters or the parameters of the output pdf's $b(o|s,v)$. Here the average log likelihood of the initial model on the training data was -66.08. This might, at first, seem like an improvement over the initial system which uses SLSD as an indicating feature because the log likelihood in this case was -67.60. However, the likelihoods of these two systems are not directly comparable as the LDA derived indicating features occupy a much smaller part of the feature space than the SLSD values. As a result their variances tend to be smaller and their likelihoods are therefore usually larger. In contrast to the experiments with SLSD as an indicating feature, the experiments with

updated	top	Overall	F0	F1	F2	F3	F4	F5	FX	xRT	like
$b(d\|s)$	fact	21.5	12.4	20.4	30.7	30.9	24.0	28.0	42.1	3.66	
+	full	21.5	12.3	20.5	30.5	31.6	24.3	27.5	42.3	4.52	-65.90
soft	par	21.6	12.4	20.2	30.7	31.2	23.9	27.2	43.7	4.45	
	fact	21.2	12.4	20.2	29.9	30.8	23.9	26.2	41.6	3.66	
$b(o\|s,v)$	full	21.3	12.3	20.4	30.1	30.9	24.0	26.5	41.9	4.58	-65.59
	par	21.5	12.5	19.7	30.7	31.4	24.7	25.3	43.0	4.43	
$b(d\|s)$	fact	21.2	12.3	20.4	30.1	30.9	23.8	26.3	41.5	3.70	
+	full	21.3	12.4	20.5	30.3	31.0	23.9	26.8	41.3	4.52	-65.59
soft	par	21.6	12.5	19.9	30.9	31.8	24.6	25.9	43.2	4.37	
	fact	21.1	12.4	19.9	29.9	31.1	23.8	26.5	41.2	3.64	
$b(o\|s,v)$	full	21.1	12.4	19.8	30.1	31.4	23.7	26.8	41.4	4.46	-65.47
	par	21.6	12.5	20.1	30.7	31.9	24.7	27.4	42.2	4.57	

Table 7.4 *Recognition performance for an RTI initialised SBME-HMM with an LDA derived indicating feature and 2 v-states.*

the LDA derived features show a smaller gain in likelihood in the first two iterations which is about a factor of 1.2. Also the overall increase in log likelihood after a total of 8 iterations is smaller at 0.61 absolute as compared to 0.86 in the SLSD case. The decrease in error rate, however, is larger giving an overall error rate of 21.1% for the factorised and full topologies. As in table 7.3 the parallel topology performs worst. Here, however, the difference between full and factorised topologies on one side and the parallel topology on the other is not as big as in table 7.3. Especially after the first two iterations all three systems in table 7.4 show approximately the same performance. The parallel topology seems to be slightly biased towards the F1 condition which becomes even more apparent after the following 2 iterations. Here the parallel topology yields the worst performance on the F0 condition while its performance on the F1 condition is superior to that of the other two topologies. Later iterations, however, seem to remove this advantage of the parallel topology and in fact after 8 iterations full and factorised topology perform better on the spontaneous speech condition than the parallel topology.

The experiments in this section show a number of things. First by comparing table 7.3 and table 7.4, one can see that phone dependent LDA derived indicating features result in better performance than the SLSD parameters. It is also apparent that the parallel topology whose parameters were derived from a trained factorised system gives the worst performance in both cases. For LDA derived features, however, it seems that the performance degradation is less dramatic resulting at one point in the training procedure in a model that is heavily biased towards spontaneous speech on which it outperforms the other two topologies. This hints at the possibility that an indicating feature which is strongly correlated with phone duration will result in a factorised or full model which still has features of a parallel topology. The run times of parallel and full topology systems in all the experiments are approximately the same and are by a factor of about 1.2 higher than for the factorised system. The run times for the LDA derived features and SLSD are also approximately the same. Overall the factorised system with phone dependent LDA derived features gives the best performance both in terms of WER and run time.

7.5.3 Median split initialisation

This section describes experiments which used the MSI method in section 5.5.2 to initialise the models. The system on which initialisation was based, in this case, was again derived from the 12 mixture baseline system. This time the indicating feature was added as an additional stream to the baseline system with a single Gamma or Gaussian output pdf $b(d|s)$ per state. This model was subsequently trained with two EM iterations to learn the proper output pdf's associated with the new stream and were then used to apply MSI to obtain an SBME-HMM with a binary hidden variable v. This procedure therefore resulted in a model which had about the same number of parameters as the 24 mixture baseline system in section 7.5.1. Again, tables 7.5 and 7.6 give the results for SBME-HMM's with SLSD and LDA as indicating features, for various stages in the training process. As before, each model was derived by performing two iterations on the model parameters indicated by the entries in the first column. In contrast to tables 7.3 and 7.4, the first parameters to be updated in these experiments where the output pdf's $b(o|s, v)$. This is the case because the median split initialisation operates on the softmax functions and leaves the $b(o|s, v)$ unchanged. Comparing table 7.3 and 7.5, one can see that already two iterations after the initialisation the SLSD median split system outperforms the fully trained SLSD system that was initialised with a model trained on the relabelled training data. Subsequent iterations leave the performance for the factorised and full topology virtually unchanged, although the average

updated	top	Overall	F0	F1	F2	F3	F4	F5	FX	xRT	like
	fact	21.1	12.3	20.2	29.9	31.0	23.9	26.0	41.1	3.46	
$b(o\|s,v)$	full	21.1	12.3	20.2	30.0	31.0	23.8	26.2	40.8	4.34	-66.73
	par	22.7	13.4	21.1	32.3	32.5	25.6	29.0	44.5	4.40	
$b(d\|s,v)$	fact	21.1	12.3	20.2	29.8	30.7	23.9	25.9	41.1	3.46	
+	full	21.2	12.3	20.3	30.0	30.9	23.9	26.2	41.4	4.36	-66.71
soft	par	23.3	13.7	22.2	32.9	33.1	26.4	29.9	45.4	4.56	
	fact	21.1	12.3	20.0	30.0	31.5	24.1	25.4	40.7	3.54	
$b(o\|s,v)$	full	21.1	12.2	20.0	30.3	31.3	23.8	25.4	41.0	4.38	-66.59
	par	23.4	13.6	22.0	33.4	32.8	26.5	30.0	46.0	4.35	
$b(d\|s,v)$	fact	21.1	12.3	20.0	30.1	31.3	24.1	25.7	41.0	3.53	
+	full	21.1	12.3	20.1	30.3	31.4	23.8	25.7	40.9	4.24	-66.59
soft	par	23.6	13.9	22.0	33.8	33.5	26.7	29.4	45.7	4.53	

Table 7.5 *Recognition performance for an MSI initialised SBME-HMM with an SLSD indicating feature and 2 v-states.*

log likelihood still increases slightly. As in previous experiments the performance of the parallel topology becomes worse. In this case the discrepancy between the performance of the factorised and parallel model whose parameters were derived by training a factorised system becomes even more pronounced with a difference of 2.5% absolute as compared to 0.9% previously. This indicates that the models derived from the relabelled training data are still closer to a parallel topology, even after a number of iterations, than in the case of a median split initialised model.

Table 7.6 gives the results for the SBME-HMM with LDA derived indicating features. Just as before, this model was initialised from a model with the indicating feature added as a new stream which was trained with two iterations. This table shows that also for the LDA derived features the performance after only two iterations is the same as for the fully trained RTI initialised model in table 7.3. Here, however, subsequent iterations decrease the error rate even further resulting in a gain over the fully trained system in table 7.4 of 0.3% absolute. As in the case of the SLSD features, one can see that the difference in performance between the factorised and parallel topology is much larger than for the LDA model in the last section. This again hints at the fact that median split initialisation creates models which are "less" parallel in structure than models that have been initialised with relabelled data. It is interesting to note that increased performance, in this case, does not relate to an increase in the average likelihood of the training data. In fact, the likelihoods in tables 7.4 and 7.6 are virtually identical.

While all the experiments discussed so far in this chapter trained different model parameters separately, in the experiments in table 7.7 all the parameters were trained at the same time. The first column in table 7.7 repeats the baseline performance mentioned in table 7.2 for the 24 mixture baseline system, while the second and third row give the results for SBME-HMM's with SLSD and LDA derived indicating features initialised with the RTI method. The fourth and last

updated	top	Overall	F0	F1	F2	F3	F4	F5	FX	xRT	like
	fact	21.1	12.2	20.1	30.2	30.4	24.2	27.7	40.7	3.50	
$b(o\|s,v)$	full	21.2	12.2	20.0	30.1	30.4	24.4	28.1	41.1	4.50	-65.57
	par	22.3	13.0	21.0	32.5	32.0	24.9	28.3	42.8	4.42	
$b(d\|s)$	fact	21.1	12.2	20.0	30.0	30.2	24.1	28.3	40.7	3.70	
+	full	21.1	12.2	20.1	30.0	30.4	24.1	27.8	40.9	4.53	-65.57
soft	par	22.4	13.2	20.7	32.4	32.4	25.2	29.1	43.2	4.30	
	fact	20.8	12.2	19.6	29.6	29.9	23.9	26.6	40.1	3.49	
$b(o\|s,v)$	full	20.8	12.2	19.6	29.6	30.1	24.1	26.3	40.4	4.43	-65.44
	par	22.5	13.4	20.8	32.0	33.3	25.3	28.1	43.4	4.42	
$b(d\|s)$	fact	20.8	12.2	19.7	29.7	29.9	23.8	26.9	40.2	3.35	
+	full	20.8	12.2	19.7	29.6	30.1	23.8	26.6	40.2	4.32	-65.44
soft	par	22.4	13.4	20.7	32.2	31.9	25.1	28.6	43.0	4.31	

Table 7.6 *Recognition performance for an MSI initialised SBME-HMM with an LDA derived indicating feature and 2 v-states.*

row, on the other hand, give the results for models initialised with the MSI method. Only results for the factorised model are reported in this table since this was shown to be the best model in previous experiments. For the RTI initialised models 4 full iterations were performed and 6 for the MSI initialised models which is a rather small number as compared to the 24 iterations that are needed to derive the 24 mixture from the 12 mixture baseline system. The results in table 7.7 show that training the SBME-HMM model parameters in this way gives a small but consistent improvement over the performance in tables 7.3 and 7.4.

	Overall	F0	F1	F2	F3	F4	F5	FX	xRT
24 mix base	21.3	12.2	20.3	29.9	32.0	24.4	25.7	41.7	3.94
SLSD RTI (4 it.)	21.2	12.6	19.8	30.4	31.3	23.5	25.4	41.3	3.42
LDA RTI (4 it.)	21.0	12.3	19.7	29.9	31.3	23.3	27.4	41.5	3.65
SLSD MSI (6 it.)	21.0	12.3	20.1	29.6	32.0	23.7	25.3	41.0	3.63
LDA MSI (6 it.)	20.7	12.1	19.6	29.5	30.3	23.4	26.0	40.2	3.74

Table 7.7 *Recognition performance for SBME-HMM's with factorised topology and 2 v-states whose parameters were trained simultaneously.*

It is interesting to note that although the classification experiments in chapter 6 showed that the duration of vowels was more strongly correlated with the indicating features than the duration of consonants, it was not sufficient to split the states of the vowels exclusively. Table 7.8 gives the results for an SBME-HMM with a factorised topology where only triphones with a vowel in the centre were initialised with the median split method. Table 7.8 shows that this model's performance is by 0.6% absolute worse than that of the model in the last row of table

	Overall	F0	F1	F2	F3	F4	F5	FX	xRT
LDA MSI	21.3	12.4	20.0	30.3	30.8	24.4	27.8	41.0	2.89

Table 7.8 *Recognition performance for an MSI initialised SBME-HMM with an LDA derived indicating feature where only vowels had two v-states.*

7.7. However, the number of parameters in the model is considerably smaller than in any of the models in table 7.7. Comparing table 7.8 with the first row in table 7.7 shows therefore that it is possible to achieve the same performance with an MSI initialised SBME-HMM as with a standard HMM that has a much higher number of parameters. This is also reflected in the realtime factor in table 7.8 which is by more than one smaller than for the 24 mixture baseline system in table 7.7.

Finally table 7.9 gives the significance levels between the different systems in table 7.7. The numbers in this table give the probability in percent of observing the output of the two systems under the null-hypothesis. A high value in this table therefore indicates that the differences between systems are *not* statistically significant. Since the null-hypothesis is usually rejected

	24 mix base	SLSD RTI	LDA RTI	SLSD MSI	LDA MSI
24 mix base	$*$	70.4%	12.1%	19.0%	0.1%
SLSD RTI	$*$	$*$	15.3%	27.6%	0.1%
LDA RTI	$*$	$*$	$*$	94.4%	3.4%
SLSD MSI	$*$	$*$	$*$	$*$	2.7%

Table 7.9 *Significance levels of the differences between the systems in table 7.7.*

below the 5% level, table 7.9 shows that the systems in the first 4 rows of table 7.7 are essentially the same. The MSI initialised SBME-HMM model with an LDA derived indicating feature, on the other hand, is statistically significantly different from each of these models. The increase in WER over the 24 mixture baseline of about 2.8% relative is even significant at the 0.1% level.

7.6 Experiments with 3 hidden states

Preliminary experiments with a hidden variable v with 3 states showed that reducing the size of the training buffers to a maximum of 2000 samples per softmax function had a detrimental effect if the dimension of the indicating feature was greater than one. In this case, after a number of EM iterations on the softmax parameters alone another iteration on these parameters could sometimes reduce the likelihood of the training data. Although this effect was not observed to be very strong it was decided to lift the restriction of limited buffer sizes to ensure that 2 dimensional indicating features were not at a disadvantage as compared to one dimensional indicating features. Not restricting the buffer sizes meant that all the data points d_t and $\gamma_t(i,k)$ where stored in the accumulators as long as the state likelihood $\gamma_t(i)$ was above a certain threshold. This was chosen to be the same threshold as the one that decides which data points were accumulated for the other model components. The resulting accumulators contained up to 84 000 data points and the total size of the accumulators thus derived was about 2GB. Since this amount of accumulated data could not be read into memory all at once, the parameter update had to be performed on a number of mutually exclusive parameter sets. Since several states can share the same softmax functions the information whether a softmax function had already been updated needed to be stored as well so that a subsequent parameter update did not train the same model twice. The parameter update was split into 10 steps which resulted in program sizes of between 400 and 600 MB.

7.6.1 Relabelled training initialisation

The first set of experiments in this section uses the RTI method to initialise the model parameters. As in section 7.5.1 the v-state labels are derived by dividing the instances of triphones which share the same physical model into classes of approximately equal size depending on their durational statistics. Subsequently, 4 iterations were performed on the relabelled training data to give robust estimates of the v-state dependent models. These models were then combined according to the RTI method to give models which used SLSD as an indicating feature as well as features derived from the phone dependent LDA transforms in section 6.2.3.2. Since these transforms are 2 dimensional for a 3 class problem this resulted in 2 dimensional indicating features as compared to SLSD whose feature vectors are one dimensional. Once RTI was completed, only the $b(d|s)$ and softmax parameters were initially trained with 2 EM iterations. This was done to allow the softmax functions for the 2 dimensional indicating features to adapt to the statistics of the initialisation, before being trained in combination with the remaining model parameters. After the first 2 EM iterations another 4 iterations were performed on the complete parameter set. The total number of iterations including the ones during the initialisation phase were therefore 10. This slight increase in the number of iterations as compared to the training in section 7.5.2 tried to make up for the much larger number of iterations that would be used if the mixtures in a 12 mixture baseline system were increased to 36 using the same method by which the 24 mixture baseline system in section 7.5.1 was derived. Since each addition of 2 new mixture components requires 4 iterations in this method, the total number of iterations in this case would be 48.

Table 7.10 gives the average log likelihoods for the v-state dependent models during the first 4 iterations of the RTI method. Here, the column named "init" gives the likelihood of the model that was derived by simply duplicating the models in the original 12 mixture baseline system. Since the models for the different v-states are the same in this case, this is the same number as for the 12 mixture baseline system. Subsequent iterations give an increase in log likelihood by about 1.5% relative which seems to be reasonably saturated after 4 iterations.

Table 7.11 gives the average log likelihoods as well as some other information derived from the training of models that used SLSD as an indicating feature. The column named "init" in this table gives the likelihood of the model that was derived with RTI from the models that have

init	1	2	3	4
-65.30	-64.58	-64.46	-64.39	-64.35

Table 7.10 *Log likelihoods of a standard HMM trained on relabelled training data.*

	EM iteration						
	init	soft+$b(d\|s)$		all parameters			
		1	2	3	4	5	6
log like	-67.43	-66.96	-66.95	-66.64	-66.46	-66.34	-66.27
nr. iter.	$*$	5.38	4.09	3.46	3.90	3.77	3.63
ρ	$*$	0.813	0.754	0.711	0.741	0.732	0.722
CPU sec.	$*$	0.33	0.28	0.25	0.27	0.27	0.27

Table 7.11 *Log likelihoods and softmax parameter update information for an RTI initialised SBME-HMM with an SLSD indicating feature and 3 v-states.*

been trained on the relabelled training data. Comparing this with the average log likelihood of the initial model in the case of 2 hidden states, which was -67.60, one can see that the increase in hidden states results in a small increase in likelihood. Also, subsequent iterations preserve this difference as can be seen by comparing table 7.11 to table 7.3. The first update in table 7.3 was performed on the $b(d|s)$ and softmax parameters by applying two EM iterations. The log likelihood of -67.19 therefore corresponds to the value of -66.95 in table 7.11. The log likelihood at the bottom of table 7.3, which is -66.74, approximately corresponds to the log likelihood of the model in table 7.11 for the fourth iteration, which is -66.46. Although this model's softmax parameters where trained with one more iteration, the small gain in likelihood affected by updating the softmax parameters in later stages of the training suggests that this is a fair comparison. Overall, the introduction of an additional hidden state therefore has a positive effect on the likelihood of the training data.

Table 7.11 also contains information about the average number of iterations that were needed to update the softmax parameters, which is given in the second row of this table. Furthermore table 7.11 gives the average value of the learning rate ρ and the average CPU time needed for the softmax parameter update during the various stages of the training process. As previously, the CPU time is measured on a 1GHz Pentium 3 processor.

It can be seen from table 7.11 that the average number of iterations in the softmax parameter update decreases over the first three EM iterations. In the fourth iteration, however, one can observe a slight increase in this number. This is a consequence of the fact that up until the third EM iteration the unaltered $b(o|v,s)$ output pdf's are used in the expectation step of the EM algorithm and the classification problems on which the softmax functions are trained remain therefore more or less unchanged. After the first update of the $b(o|v,s)$ output pdf's in the third EM iteration, however, these classification tasks are redefined. Since the softmax functions are not as well adapted to the new classification problems, slightly more iterations on these parameters are needed to find the optimal solution.

Changes in the number of iterations on the softmax parameters directly correlate to changes in the average number of CPU seconds and they also correspond to changes in the average values of the learning rate. It seems that further away from the optimal solution the learning rate is higher and decreases in later stages of the training process. This is a similar effect as the one observed in the training of the softmax functions with an eighth degree exponential polynomial on the training set in figure 5.2(a). Here, figure 5.7(a) in section 5.4 showed that after a full step in the Newton direction in the first iteration the learning rate slowed down considerably. In the example in section 5.4, however, the learning rate increased again in later iterations. The reason why this is not observed here is probably due to the relatively low complexity of the classification tasks in the experiments in table 7.11 as compared to the one in figure 5.2(a). Overall the computational cost involved in the softmax parameter updates is relatively low in table 7.11 resulting in a total of 2270 CPU seconds for the complete model update in the first iteration and 1813 CPU seconds for the update in the final iteration, which is a decrease in computational cost of about 20%.

Table 7.12 is the equivalent of table 7.11 and gives the same information for models that used LDA derived indicating features, instead of SLSD, and softmax functions with linear and quadratic polynomials. The degree of the polynomials in this table is contained in the column

deg		EM iteration						
		init	soft+$b(d\|s)$		all parameters			
			1	2	3	4	5	6
1	log like	-66.776	-66.452	-66.439	-66.123	-65.946	-65.837	-65.763
	nr. iter.	$*$	5.30	4.07	3.43	3.73	3.57	3.43
	ρ	$*$	0.811	0.753	0.708	0.728	0.718	0.707
	CPU sec.	$*$	0.80	0.66	0.58	0.62	0.58	0.57
2	log like	-66.776	-66.447	-66.435	-66.118	-65.946	-65.835	-65.762
	nr. iter.	$*$	5.99	4.50	3.81	4.29	4.17	4.01
	ρ	$*$	0.826	0.768	0.729	0.751	0.743	0.733
	CPU sec.	$*$	2.39	1.92	1.68	1.78	1.74	1.68

Table 7.12 *Log likelihoods and softmax parameter update information for an RTI initialised SBME-HMM with an LDA derived indicating feature and 3 v-states.*

denoted by "deg". In order to be able to compare the average log likelihoods of the softmax functions with linear and quadratic polynomials these values are given in table 7.12 up to the third decimal position. The phone dependent LDA transforms used here are the same as in section 6.2.3.2. Since these were derived for a 3 class problem, their output is 2 dimensional. The polynomials in the softmax functions have therefore two indices and the number of parameters in three softmax functions with 2 linear polynomials is $2 \times 3 = 6$ which increases to $2 \times 6 = 12$ for the quadratic polynomials. The latter models therefore use twice as many parameters and studying the computational cost of updating the different models will give an indication as to how the computational cost of the softmax parameter updates scales with model complexity.

The column denoted by “init” in table 7.12 gives, again, the average log likelihood of the training data under the model that has been initialised based on the models that were trained on the relabelled training data. Since the polynomials of the softmax functions both for the linear and quadratic case were initialised to be identically 0, the two likelihoods in the “init” column are exactly the same. It is interesting to note, that the initial likelihood in this table is higher than the one in table 7.11 although the dimension of the indicating feature is higher. This is again a result of the fact that the LDA transformed indicating features occupy a much smaller area in the feature space than the SLSD features and their likelihoods are therefore higher.

Table 7.12 shows, again, that successive iterations increase the likelihood of the training data. Here the first iteration on the $b(d|s)$ and softmax parameters gives an improvement in log likelihood of about 0.3. This is similar to the increase in the third iteration which updates all the model parameters which illustrates that the softmax functions and the $b(d|s)$ output pdf’s make a substantial contribution to the overall model fit although, typically, they tend to be saturated fairly early in the training process.

Table 7.12 also shows that there is little improvement in the log likelihood by using quadratic instead of linear exponents for the softmax functions. Although the difference between these numbers is larger in early iterations they are almost identical in the final one. This is somewhat disappointing because as table 7.12 also shows, the update of the softmax functions with quadratic exponents is computationally more expensive than the update for linear exponents. While there is not a big difference between the number of iterations necessary to update the softmax parameters the increase in CPU time is quite substantial by about a factor of 3. This is due to the higher number of parameters in the quadratic model which means that the calculation of the first and second order derivatives of the cross-entropy error function requires more computational effort. The similar number of iterations for both linear and quadratic softmax exponents is also reflected in similar numbers for the learning rate. As in table 7.11, one can observe a steady decrease in the number of iterations and learning rate over the first 3 iterations which is followed by a slight increase in the fourth iteration. As in the case of an SLSD indicating feature this can be attributed to the unchanged $b(o|v, s)$ output pdf’s which are used in the first 3 expectation steps of the EM iterations.

Comparing table 7.12 to table 7.11 one can see that the average number of iterations needed to update the softmax functions for the SLSD model is slightly higher than for the LDA derived indicating feature using linear exponents in the softmax functions. For softmax functions with quadratic exponents, however, updating the parameters needs somewhat more iterations than for the one dimensional SLSD feature. Although the update of the softmax functions with linear exponents needed less iterations than the update of the SLSD model the average CPU time needed for this process is higher by a factor of 2.3 on average which can, again, be attributed to the higher number of parameters in the two dimensional softmax functions. Since the number of parameters for three one dimensional softmax functions with linear exponents is $2 \times 2 = 4$, the increase in the number of model parameters is two in this case. Plotting the average CPU times against the number of parameters in the model gives a fairly straight line which can also be seen by dividing the difference in average CPU usage between the models by the difference in parameters which results in values close to 0.17. In the range of model complexities investigated here the addition of one parameter in the softmax models corresponds to an increase in CPU time of about 0.17 seconds in the model update. One has to be careful, though, with extrapolating the CPU usage based on this scaling factor since there are two elements that contribute to the computational complexity of the softmax training. One is, obviously, the number of parameters in the softmax function, while the other is the complexity of the training set. For other indicating features, the decision boundaries might become more complicated and the CPU time estimated with the scaling factor above might therefore not be accurate.

The complete model update in table 7.12 took 5392 and 16038 CPU seconds in the first iteration for the linear and quadratic exponents, respectively. This number was reduced to 3845 and 11263 CPU seconds in the last iteration which is a reduction of almost 30% which is a larger speed up than for the model using an SLSD indicating feature.

Table 7.13 gives the word error rate of the model whose training has been documented in table 7.11 and therefore uses SLSD as an indicating feature. By comparing the first row of

topology	Overall	F0	F1	F2	F3	F4	F5	FX	xRT
fact	21.2	12.4	19.8	30.1	31.6	24.7	25.7	41.0	4.77
full	21.3	12.4	19.7	30.1	31.8	24.9	26.9	42.1	6.44
par	23.0	13.2	21.7	32.2	33.2	26.2	28.6	46.8	6.61

Table 7.13 *Recognition performance for an RTI initialised SBME-HMM with an SLSD indicating feature and 3 v-states.*

this table to the second row in table 7.7 one can see that, although an increase in likelihood was observed in table 7.11 over table 7.3, the introduction of an additional state of the hidden variable v does not improve recognition performance. Overall the performance stays the same with a slight increase on the F0 condition from 12.6% previously to 12.4% in table 7.13 and constant performance on the F1 condition and again a slight improvement on F2 going from 30.4% to 30.1%. For conditions F3-F5 the performance is worse than for the 2 state model and for FX one can again observe a small improvement of 0.3% absolute. In addition to the unchanged overall recognition performance, the realtime factor of the recognition process was also increased due to the additional v-state from 3.42 to 4.77.

These results might at first seem disappointing, but one has to bear in mind that the models whose performance is collected in table 7.13 were derived by initially duplicating the 12 mixture baseline system and they therefore do not implement any additional parameter sharing. In addition to the large number of parameters, the models in table 7.13 also suffer from the comparatively small number of EM iterations after which their log likelihoods becomes saturated. Both these effects combine to prevent any further improvement over the system in table 7.7.

Similar to the case of an SBME-HMM with 3 hidden states and SLSD as an indicating feature, table 7.14 shows that no improvement in recognition performance can be obtained by adding a hidden state to a model that uses the LDA derived indicating feature and linear softmax exponents. Comparing, again, the first row in table 7.14 to the third row in table 7.7 shows that in

topology	Overall	F0	F1	F2	F3	F4	F5	FX	xRT
fact	21.1	12.2	20.2	29.6	31.4	23.8	26.8	42.0	5.11
full	21.2	12.2	20.3	29.7	31.7	23.9	26.5	42.4	7.01
par	22.4	13.2	20.7	30.9	33.5	26.1	29.3	44.7	6.90

Table 7.14 *Recognition performance for an RTI initialised SBME-HMM with an LDA derived indicating feature, 3 v-states and softmax functions with linear polynomials.*

this case the addition of a hidden state even decreases the overall system performance by going from a WER of 21.0% to 21.1% in table 7.14. Perhaps more surprising is the fact that the model in table 7.14 is less biased towards the spontaneous speech condition than the model in table 7.7. In fact, the performance on the F0 condition is by 0.1% absolute better in table 7.14. On the other hand, performance on the F1 condition is considerably worse at 20.2% as compared to 19.7% in table 7.7. Here the increase in run time is even larger than in the case of an SLSD indicating feature giving a realtime factor of 5.11 as compared to a factor of 3.65 in table 7.7. This is, of course, due to the higher dimension of the indicating feature.

Table 7.15, finally, gives the recognition results for LDA derived indicating features with a model that uses quadratic softmax exponents instead of linear ones. The minute increase in

topology	Overall	F0	F1	F2	F3	F4	F5	FX	xRT
fact	21.2	12.4	20.4	29.5	31.0	23.6	28.7	41.9	5.20
full	21.2	12.4	20.2	29.3	30.9	23.9	28.3	42.0	7.36
par	22.4	13.2	20.7	31.0	33.1	26.3	29.7	44.7	7.27

Table 7.15 *Recognition performance for an RTI initialised SBME-HMM with an LDA derived indicating feature, 3 v-states and softmax functions with quadratic polynomials.*

likelihood on the training data in table 7.12 already indicated that one could not expect any large improvements in recognition performance. Quite on the contrary, table 7.15 shows that using quadratic softmax exponents even decreases performance by 0.1% absolute. As for the linear exponents the models in table 7.15 exhibit no bias towards the spontaneous speech condition and, in fact, add to the word error rate for both F0 and F1 as compared to the models in table 7.14. The recognition run time in table 7.15 was slightly increased as compared to the models with linear softmax exponents. The decrease in performance due to the use of quadratic softmax exponents is most likely a result of the fact that the decision boundaries are essentially linear and therefore quadratic polynomials seem to generalise less well on unseen data.

Summarising, one has to say that in spite of a better training data match, which was indicated by an increase in likelihood of the models on the training data, the recognition performance for RTI derived SBME-HMM's did not profit from the introduction of an additional state for variable v.

7.6.2 Median split initialisation

This section describes experiments which used an SBME-HMM with a hidden variable v with three states and the MSI method to initialise the model parameters. Since this method assumes a one dimensional indicating feature the two dimensional 3 class LDA transforms from section 6.2.3.2 could not be directly applied for initialisation. Instead the phone dependent, one dimensional two class LDA transforms from section 6.2.2.2 were used to obtain a one dimensional feature which was added as a new stream to the model. As previously, this stream was modelled with a single Gaussian per $b(d|s)$ output pdf. Initially, 2 EM iterations were performed to learn the $b(d|s)$ parameters which were then used to split the indicating feature space into 3 areas of equal volume with the MSI method. While the MSI initialised softmax parameters were kept fixed, 4 more EM iterations were performed on the $b(o|v, s)$ output pdf's. After that the one dimensional LDA transforms were replaced with the two dimensional ones and the $b(d|s)$ output pdf's were initialised with the same methods that are used in RTI. Here, the softmax functions were initially set to be identically zero. In order to adapt to the $b(o|v, s)$ that had already been learned, and to keep the influence of the untrained $b(d|s)$ output pdf's and softmax functions on the $b(o|v, s)$ to a minimum, only the initialised $b(d|s)$ output pdf's and softmax functions were trained in the next 2 EM iterations. Finally, the complete model parameters were trained with 4 more EM iterations to arrive at the model sets which were used in the recognition experiments of this section.

Unlike for the models using LDA derived indicating features, training the SBME-HMM's with SLSD as indicating feature was straight-forward since the squared length of the second order derivatives is one dimensional. In order to be able to compare the performance of these two systems, the same training strategy was applied in the case of SLSD as well without, however, the initialisation of the softmax and $b(d|s)$ parameters after the first 4 EM iterations.

Compared to the training of the models in section 7.6.1 the training of the MSI derived models used two more iterations which were needed at the beginning of the training procedure to learn the one dimensional output pdf's $b(d|s)$. Since the increase in likelihood at the end of the training of the models in section 7.6.1 was very small the comparison of the performance of models in this section and the latter models is therefore sensible.

Table 7.16 gives the likelihood for the first 4 iterations on the $b(o|v, s)$ output pdf's after the softmax parameters were initialised with MSI where the model used SLSD as an indicating feature. Comparing the numbers in table 7.16 to table 7.10 one can see a decrease in log likeli-

init	1	2	3	4
-67.56	-66.56	-66.33	-66.21	-66.12

Table 7.16 *Log likelihoods for the first 4 iterations of an MSI initialised SBME-HMM with an SLSD indicating feature and 3 v-states.*

hood which is the result of the presence of an indicating feature in table 7.16. Also, comparing table 7.16 to 7.11 shows that the MSI initialised model has a higher likelihood on the training data already after the first 4 iterations than the fully trained RTI initialised system. This again illustrates the superiority of MSI as an initialisation scheme over RTI. Since the $b(d|s)$ output pdf and the softmax functions remain unchanged between the fourth iteration in table 7.16 and the first iteration in table 7.17 the log likelihood after the fourth iteration in table 7.16 and the initial model in table 7.17 are the same.

Table 7.17 shows that the increase in likelihood on the training data over the last 6 iterations is smaller than the increase over the first 4 iterations. Especially, training the softmax and $b(d|s)$

	EM iteration						
	init	soft+$b(d|s)$		all parameters			
		1	2	3	4	5	6
log like	-66.12	-66.11	-66.11	-66.05	-66.00	-65.97	-65.94
nr. iter.	$*$	4.42	3.67	3.20	3.79	3.75	3.68
ρ	$*$	0.772	0.727	0.687	0.736	0.733	0.727
CPU sec.	$*$	0.28	0.27	0.25	0.26	0.25	0.27

Table 7.17 *Log likelihoods and softmax parameter update information for an MSI initialised SBME-HMM with an SLSD indicating feature and 3 v-states.*

parameters seems to give only a minute improvement. This is similar to the results in table which showed that updating the softmax and $b(d|s)$ parameters of an MSI initialised model with a two state hidden variable v does not contribute much to the likelihood of the training data under the model.

As in the previous section, table 7.17 also contains information about the update of the softmax parameters. In comparison to table 7.11, table 7.17 shows a reduced number of average iterations per softmax update up to the fourth EM iteration, while for the last two EM iterations the number of iterations in the softmax update is virtually the same. The largest difference occurs in the first iteration where the MSI initialised model needs an average of 4.42 iterations to update the softmax parameters as compared to 5.38 for the RTI initialised model. This is the case because the $b(o|v,s)$ output pdf's in the MSI initialised model have been trained on the initial softmax functions and therefore replicate to a certain extend the classification problem that was defined by the initial softmax functions.

As in the last section, the number of iterations in the softmax parameter update decreases on the first 3 EM iterations and increases in the fourth iteration due to the use of updated $b(o|v,s)$ output pdf's in the expectation step of the EM algorithm. The correlation between learning rate and average number of iterations in the softmax parameter update is also the same, the latter decreasing whenever the former does. Finally, the average number of CPU seconds for a softmax parameter update is approximately constant in table 7.17 at about the same value as for the second to last EM iterations in table 7.11. The total number of CPU seconds used in the complete model update was around 1800 in all the EM iterations.

Table 7.18 contains the average log likelihoods for the first 4 EM iterations of an MSI initialised model that used LDA derived indicating features. As mentioned before, the LDA transformations used in these experiments were the one dimensional two class transforms from section 6.2.2.2 which were chosen to be able to apply MSI which relies on one dimensional feature vectors. As a result of the more compact LDA space the log likelihoods in table 7.18 are again

init	1	2	3	4
-66.30	-65.46	-65.22	-65.09	-64.95

Table 7.18 *Log likelihoods for the first 4 iterations of an MSI initialised SBME-HMM with a one-dimensional LDA derived indicating feature and 3 v-states.*

higher than the ones in table 7.16 while the increase is approximately the same at about 2% relative.

Table 7.19 gives the average log likelihoods for the iterations following the ones in table 7.18 where the one dimensional LDA transforms were replaced by two dimensional transforms. This table contains information for models having softmax functions with both linear and quadratic polynomials. As in the previous section, the first two iterations in table 7.19 operate on the softmax and $b(d|s)$ parameters alone which is indicated by the entry in the second row of the table. The remaining iterations update all the model parameters which is indicated in the second row of the table. One can see from table 7.19 that the introduction of the 2 dimensional indicating

deg		EM iteration						
		init	soft+$b(d\|s)$		all parameters			
			1	2	3	4	5	6
1	log like	-66.346	-65.848	-65.814	-65.698	-65.629	-65.578	-65.538
	nr. iter.	$*$	6.38	4.82	3.87	3.45	3.44	3.37
	ρ	$*$	0.843	0.792	0.742	0.710	0.709	0.703
	CPU sec.	$*$	0.955	0.734	0.639	0.590	0.560	0.546
2	log like	-66.346	-65.845	-65.813	-65.697	-65.629	-65.578	-65.538
	nr. iter.	$*$	6.83	5.11	4.19	4.25	4.14	4.10
	ρ	$*$	0.853	0.803	0.760	0.758	0.752	0.750
	CPU sec.	$*$	2.70	2.07	1.77	1.73	1.75	1.74

Table 7.19 *Log likelihoods and softmax parameter update information for an MSI initialised SBME-HMM with an LDA derived indicating feature and 3 v-states.*

feature after the first four iterations in table 7.19 decreases the log likelihood of the training data by more than -1.2 which amounts to a decrease in likelihood by a factor of about 0.3. This is the effect of the higher dimension of the indicating feature which corresponds to smaller likelihood values of the $b(d|s)$ output pdf's. Since both the linear and second degree polynomials are initialised to be zero both log likelihood values in the column denoted by “init” are the same. The first two iterations in table 7.19 show a consistent increase in average log likelihood with the main contribution affected by the first iteration. The next iteration which is performed on all the model parameters again provides a substantial likelihood increase which is much higher than in the following iterations. Comparing the likelihood for the linear and second degree exponential polynomials in table 7.19 one can only notice a very small difference. This again indicates that the nature of the decision surfaces is essentially linear.

The models whose training has been discussed so far in this section were used in recognition experiments which are summarised in tables 7.20, 7.21 and 7.22. These experiments used again the three different types of model topology from section 5.6, namely the factorised, full and parallel topology whose parameters were derived from a trained factorised system.

Table 7.20 gives the WER for the SBME-HMM with 3 hidden states and SLSD as an indicating feature. This table shows small improvements over the results in table 7.13 for the factorised and full topologies whereas the performance for the parallel topology is worse by about 2% absolute. This is again caused by the MSI initialisation which does not take durational statistics

topology	Overall	F0	F1	F2	F3	F4	F5	FX	xRT
fact	21.1	12.2	19.8	30.4	32.0	23.7	25.9	41.2	4.55
full	21.2	12.3	19.8	30.6	32.0	23.7	26.2	41.2	6.12
par	25.0	14.5	24.2	35.3	35.0	27.9	29.3	50.3	6.47

Table 7.20 *Recognition performance for an MSI initialised SBME-HMM with an SLSD indicating feature and 3 v-states.*

into consideration when initialising the models but rather relies on the statistics of the indicating features. Perhaps surprisingly, the overall improvement is mainly due to an improvement on the F0 condition while for spontaneous speech the full and factorised models in table 7.20 and table 7.13 give the same performance. Compared to the SBME-HMM with a two state hidden variable v, SLSD as an indicating feature and the factorised topology, whose results are contained in the fifth row of table 7.7, the performance of the factorised model in table 7.20 is worse by 0.1% absolute, although it appears to give small improvements on the F0 and F1 conditions.

Tables 7.21 and 7.22 give the word error rates for SBME-HMM's with LDA derived indicating features and 3 hidden v-states for linear and quadratic softmax exponents. Again, the minute differences in likelihood between these models in the training indicate that there are no big differences to be expected in terms of recognition performance. And in fact, the numbers in tables 7.21 and 7.22 are almost identical. Only the runtime of the recognition is increased in

topology	Overall	F0	F1	F2	F3	F4	F5	FX	xRT
fact	20.9	12.4	19.5	28.8	30.8	24.3	25.0	41.6	4.73
full	21.0	12.4	19.5	29.0	30.8	24.1	25.1	42.0	6.79
par	23.5	14.1	21.7	33.9	33.4	26.0	28.7	46.1	6.00

Table 7.21 *Recognition performance for an MSI initialised SBME-HMM with an LDA derived indicating feature, 3 v-states and softmax functions with linear polynomials.*

table 7.22 as a result of the higher degree of the softmax exponents. Comparing tables 7.21 and 7.22 to tables 7.14 and 7.15 respectively, it becomes evident that MSI is again the better of the two initialisation methods for the full and factorised topology. Only for the parallel topology the opposite is true. However the results for the factorised topology in tables 7.21 and 7.22 are

topology	Overall	F0	F1	F2	F3	F4	F5	FX	xRT
fact	20.9	12.4	19.5	29.2	30.4	24.1	24.8	41.3	5.42
full	21.0	12.4	19.6	29.4	30.5	24.3	25.1	41.6	7.37
par	23.5	14.2	21.5	34.0	32.2	26.3	29.1	46.1	7.15

Table 7.22 *Recognition performance for an MSI initialised SBME-HMM with an LDA derived indicating feature, 3 v-states and softmax functions with quadratic polynomials.*

worse than the result for the factorised topology of an SBME-HMM with 2 hidden v-states in the last row of table 7.7. Overall, the introduction of an additional hidden v-state therefore gave no positive contribution to the system performance.

7.7 Summary

This chapter investigated a number of different SBME-HMM models and initialisation methods and evaluated them in a series of recognition experiments. The models in these experiments were derived from a state-of-the-art 12 mixture baseline system whose mixture components were Gaussians with diagonal covariance.

Experiments with a hidden variable v that had two states in section 7.5 showed small improvements for a factorised model topology over a 24 mixture baseline system which used approximately the same number of parameters. The largest improvement was obtained for LDA derived indicating features where the model was initialised with the MSI method from section 5.5.2. The LDA transforms in these experiments where the phone dependent 2 class LDA transforms from section 6.2.2.2. Significance tests showed that the improvement in recognition performance which was in this case about 2.8% relative over the 24 mixture baseline system was significant at the 0.1% level. The differences to the other SBME-HMM's with a hidden variable v with 2 states were also statistically significant.

Increasing the number of states of the variable v to three had, unfortunately, an adverse effect on the recognition performance as was discovered in section 7.6. It was concluded that this was the result of the large number of parameters in the model and also the small number of iterations after which the likelihood of the training data under the SBME-HMM's was already saturated. Furthermore, no improvement in performance was observed by choosing quadratic softmax exponents over linear exponents in the case of a two dimensional indicating feature.

Detailed investigations of the softmax parameter update in section 7.6 showed that the number of Newton iterations needed on average was relatively small and the CPU usage scaled about linearly with the number of model parameters in the range of model complexities that were discussed. It seems therefore plausible that the introduction of higher dimensional indicating features with more complicated decision boundaries in future experiments is possible without stretching CPU usage in the softmax parameter update beyond reasonable limits.

8

Conclusions and further work

This chapter reviews some of the results that were presented earlier and suggests various directions for future research in connection with SBME-HMM's.

8.1 Conclusions

After introducing the basic theory of HMM's in chapter 2, this thesis set out by discussing various models in chapter 3 which were either directly proposed to describe hidden dynamics in the speech process or which were conceptually related to such models. Special room was given in this chapter to the hidden mode model, the subband and multi-stream models as well as the mixture of experts architectures and finally the dynamic Bayesian networks. Finally, section 3.5 summarised a number of other approaches which were either related to the models discussed before or were proposed to tackle the problem of recognising spontaneous speech.

Chapter 5 introduced a new model called the state based mixture of experts hidden Markov model (SBME-HMM), which shared some of its features with the models in chapter 3 while trying to avoid some of their inherent problems. Since the SBME-HMM implements a special factorisation of the joint state and observation likelihood, special care was taken to motivate this particular choice of factorisation as compared to other possibilities. This was done in section 5.3 which compared softmax functions, neural networks and Gaussian mixture models for the purpose of modelling posterior probabilities. Subsequently, the theoretical properties of the SBME-HMM were developed and it was shown that the SBME-HMM can be efficiently trained with the EM algorithm by using a variant of the forward-backward algorithm. Also the existence and uniqueness of solutions to the softmax parameter estimation problem in the maximisation step of the EM algorithm was briefly discussed. This was later formulated in a more general frame-work in appendix C where such solutions where shown to exist. In addition to the theoretical analysis of the SBME-HMM training, chapter 5 also proposed three different types of topologies which can be used for recognition.

The development of indicating features which play an integral part in the SBME-HMM was investigated in chapter 6. Here the indicating features were derived from the standard observation o, with three different types of dimensionality reducing transforms. Two of these transforms were linear, namely PCA and LDA, while the other was the squared Euclidean norm of some of the subvectors of o. The different schemes were evaluated in classification experiments on the training set of the 1997 Broadcast News task in either phone dependent or phone independent fashion. The results in this chapter showed that phone dependent LDA transforms gave the best classification performance for both 2 class and 3 class experiments and that taking the squared length of the second order derivative of o gave the best one dimensional phone independent feature in the 2 class experiments. These two features were therefore used in the recognition experiments in chapter 7 to evaluate the influence of phone dependent versus phone independent indicating features on the recognition performance of an SBME-HMM.

Chapter 7 discussed results of a number of recognition experiments which contrasted the performance of SBME-HMM's with standard HMM's that had a similar number of parameters. These experiments showed a small but statistically significant improvement of 2.8% relative over the HMM baseline for an SBME-HMM with a binary hidden variable v and an indicating feature that was derived by applying the phone dependent LDA transforms from section 6.2.2.2. Experiments with a hidden variable v with 3 states did not give any improvements and, in fact, degraded system performance slightly. A detailed analysis of the computational cost of training the softmax parameters in section 7.6 showed that CPU usage scaled about linearly with the number of parameters which seems promising for future experiments with higher dimensional indicating features and more complicated classification problems. Overall the results in chapter 7 showed that the phone dependent LDA derived indicating features were superior to the SLSD feature and that the factorised model topology gave the best performance while the parallel model whose parameters were derived from a trained factorised model performed worst.

8.2 Suggested further work

This section discusses some of the techniques that could be applied to the SBME-HMM further to the ones that were investigated in this thesis.

8.2.1 Training the transforms

The recognition experiments in chapter 7 showed that the quality of the dimensionality reduction scheme by which the indicating feature is derived has a direct influence on the recognition performance of the SBME-HMM. The measure of quality that was applied to evaluate these schemes initially was the performance of the classifiers in chapter 6 that were built for the corresponding indicating features. However, this is a rather crude optimality criterion that has no obvious relation to the more commonly used ML criterion. Since the indicating features are eventually designed to improve recognition performance a different more model specific criterion would be more appropriate. As already pointed out at the end of section 6.1.2 such a criterion exists and is simply the likelihood of the training data under the model. Using the same techniques that were developed in (Schukat-Talamazzini et al., 1995; Gopinath, 1998; Saon et al., 2000; Gales, 1999) to optimise HDA transforms or semi-tied covariances it would therefore be possible to learn the optimal linear dimensionality reducing transforms within the EM frame-work. This seems to be a promising approach, especially, in the light of recognition results which have been reported for linear feature space transforms that have been trained in this way.

8.2.2 Using the states of v in the recognition process

The disappointing performance of the models that explicitly used the state of the hidden variable v for recognition needs further investigation. In the case of the parallel model it is obvious that a better method to derive the model parameters needs to be investigated. Here the model was directly derived from a trained factorised SBME-HMM which is clearly a suboptimal solution. However, other types of parallel models should also be investigated. One of the problems in the design of the parallel topology in section 5.6 is that it only considers the indicating feature at time t to infer the state of v. If v is, however, assumed to change slowly over time, the window over which the probability of the states of v is estimated should also be wider. This can be done in two different ways. First, one can concatenate the indicating features in a window surrounding time t and thereby create a higher dimensional indicating feature which can again be used to train all the system components. Alternatively, one can train the softmax functions on the original low dimensional indicating features and use a smoothing technique to include information about surrounding time instants. Such a smoothing technique, although in a different context, has already been presented in section 3.2 where equations (3.5) and (3.6) were used to calculate the probability that a combination of subbands are useful for recognition given that the individual subbands are reliable sources of information. This method can also be adopted to calculate the probabilities that the hidden variable v is in state k at time t given the probabilities $p(v = k|d_{t'}, s_{t'})$ in a time interval surrounding t, i.e. $t' = t - n, t - n + 1, \ldots, t + n - 1, t + n$. Applying (3.5) and (3.6) in this situation therefore yields

$$p(v_t = k|d_{t-n}, \ldots, d_{t+n}, s_{t-n}, \ldots, s_{t+n}) = \frac{p_k}{\sum_{j=1}^{K} p_j} \tag{8.1}$$

where the following holds

$$p_j = \frac{\prod_{t'=t-n}^{t+n} p(v = j|d_t, s_t)}{p(v = k)^{2n}} \tag{8.2}$$

Here $p(v = k)$ is the overall probability of observing state $v = k$. If the distribution of the states of v is not strongly biased as is most likely the case in an MSI initialised SBME-HMM this value will be close to $1/K$.

Another way to use the state of the hidden variable v explicitly in the recognition process is by making it available to objects that are at higher levels in the hierarchy of the speech recogniser. Such objects are, for instance, the probabilities for different pronunciations of a single word or transition probabilities between words in the language model. In the former situation the state of v could be used to switch between fast and slow pronunciations of a word. In the latter case the state of v could be used to change the probabilities of transitions to words in the dictionary that are models for hesitations. In any case, the use of the hidden variable in higher levels of the recogniser hierarchy will require a slowly changing state since these objects themselves usually span longer time intervals. For this reason, a smoothing technique as the one suggested for the parallel topology seems appropriate to obtain more slowly changing hidden variables.

8.2.3 Combination with other sources of information

The indicating features studied here are not the only conceivable source of information about the state of variable v. One could also use some of the speaking rate measures discussed in chapter 4 or the features used in (Shriberg and Stolcke, 1996) to detect spontaneous speech events like false starts and speech repairs. Higher level features like the ones discussed in (Shriberg et al., 1997) might also be of interest.

8.2.4 Selecting which models to split

In section 7.5 an experiment was conducted that showed that it was possible to achieve the same recognition performance as the 24 mixture baseline system with an SBME-HMM whose triphones had two v-states only if their central phoneme was a vowel. This gave a model set with a much smaller number of parameters which was also reflected in reduced recognition run time. While the performance of this model was not as good as for the SBME-HMM where all the models were initialised with MSI, this experiment shows that an application dependent optimal trade-off between system complexity and recognition performance can be found by selecting which triphones should have how many v-states. Clearly, in this experiment the choice which models had 2 v-states was rather ad hoc and was only based on the observation in chapter 6 that classification performance of the indicating features was usually better on vowels than on consonants. A more principled way of deciding which states should have how many v-states would be more appropriate.

8.2.5 Application to other hidden modes

This thesis focused on the development of a model that was performing well for spontaneous speech. However, the SBME-HMM is not restricted to this situation. As already pointed out in the introduction of this thesis the more general theme presented here is that of soft model selection which replaces the usual segmentation, labelling and model selection approach. The SBME-HMM frame-work can therefore be also applied to other situations where a soft decision as to which hidden mode is present is more appropriate. This is, for instance, the case for speech that is corrupted by background noise which can vary in both type and level. In this case, one could use a signal-to-noise measure as an indicating feature to decide which model should be used for recognition. Another application is the automatic switching between gender dependent models. Here an estimate of the f0 formant could be used as an indicating feature.

8.2.6 Single Step Newton with fixed learning rate

Section 5.4 discussed two different methods that can be used to train the softmax parameters in an SBME-HMM. The second one which was called the full Newton algorithm was the one used in the experiments in this thesis. This method was preferred over the other method in section 5.4 which was called the one step Newton algorithm because it gives the exact solution to the maximisation problem in the EM algorithm. It would be interesting to see the difference in performance between these two methods, especially, since the one step Newton algorithm is computationally much less complex than the full Newton algorithm. The one step method also has the additional advantage that it does not need to accumulate the complete sequence of training data and requires instead just the accumulation of the first and second order derivatives which are needed in the Newton update. The disadvantage of this method is that the learning rate has to be set manually. However, the experiments in section 7.6 that evaluated the performance of the softmax parameter update showed that the learning rate was always relatively high and setting this value sensibly by hand should therefore not lead to oscillatory behaviour in the model training. It is interesting to note that in the case of MMI training a parameter called d has also be set manually which has similarities to the learning rate in this case.

8.2.7 Experiments on other data bases

Although the 1997 Broadcast News task is a sensible starting point to investigate the performance of a system on spontaneous speech, performance evaluations should also be conducted on other databases like, for instance, Switchboard. This is a data base that consists entirely of spontaneous telephone conversations between two individuals and constitutes one of today's major challenge in the field of speech recognition.

A

Phone set statistics

This appendix lists all the phones of the phone set that was used in the classification experiments in chapter 6 and from which the triphones in the recognition systems in chapter 7 were derived. This phone set is given in table A.1 with an example for the pronunciation of each of the phones which also indicates whether the phone was considered to be a vowel or a consonant. This is shown in the columns denoted by “v/c” where “v” stands for vowel and “c” stands for consonant. In addition this appendix contains the statistics of the test and training sets on which the

Phone	Example	v/c	Phone	Example	v/c	Phone	Example	v/c
aa	bott	v	en	button	c	ow	boat	v
ae	batt	v	er	bird	v	oy	boy	v
ah	but	v	ey	bait	v	p	pet	c
ao	bought	v	f	fan	c	r	red	c
aw	bout	v	g	get	c	s	sue	c
ax	about	v	hh	hat	c	sh	shoe	c
axr	butter	v	ih	bit	v	t	tat	c
ay	bite	v	ix	dating	v	th	thin	c
b	bet	c	iy	beet	v	uh	book	v
ch	cheap	c	jh	jeep	c	uw	boot	v
d	bdebt	c	k	cat	c	v	van	c
dh	that	c	l	led	c	w	wed	c
eh	bet	v	m	met	c	y	yet	c
el	bottle	c	n	net	c	z	zoo	c
em	bottom	c	ng	thing	c	zh	measure	c

Table A.1 *The phone set.*

classification experiments in chapter 6 were performed which are given in table A.2.

Phone	train	test	Phone	train	test	Phone	train	test
aa	41211	10302	en	3878	968	ow	31605	7901
ae	75798	18949	er	15203	3799	oy	2531	631
ah	69244	17310	ey	48752	12188	p	50726	12681
ao	38458	9614	f	42920	10729	r	107723	26930
aw	15695	3923	g	21256	5313	s	120161	30040
ax	154451	38611	hh	31558	7889	sh	18251	4561
axr	44390	11097	ih	136459	34113	t	180440	45109
ay	40578	10143	ix	20056	5012	th	14179	3544
b	43210	10801	iy	85781	21444	uh	9471	2367
ch	10966	2741	jh	14959	3738	uw	41122	10280
d	85866	21466	k	79453	19863	v	46555	11638
dh	71559	17888	l	76438	19109	w	47164	11790
eh	67369	16841	m	66531	16631	y	21651	5411
el	17014	4252	n	176689	44171	z	72613	18153
em	498	123	ng	27215	6802	zh	1113	277

Table A.2 *Phone set statistics.*

B

Gamma densities

This section will derive the reestimation formulae that were presented in section 2.4.2 for an HMM whose output pdf's are Gamma distributions. As mentioned in section 2.4.2, Gamma densities are pdf's of a one dimensional random variable which are given by

$$\frac{\eta^\nu}{\Gamma(\nu)} d^{\nu-1} e^{-\eta d} \tag{B.1}$$

The part of the HMM auxiliary function involving the parameters of the Gamma pdf for the i-th state is therefore given by

$$\sum_t \gamma_t(i)(\nu \log \eta - \log \Gamma(\nu) + (\nu - 1)\log d - \eta d) \tag{B.2}$$

Taking the derivatives with respect to η and ν therefore yields

$$\frac{\partial}{\partial \eta} Q(\lambda, \bar{\lambda}) = \sum_t \gamma_t(i) \left(\frac{\nu}{\eta} - d_t \right) \tag{B.3}$$

$$\frac{\partial}{\partial \nu} Q(\lambda, \bar{\lambda}) = \sum_t \gamma_t(i) \left(\log \eta - \frac{\Gamma'(\nu)}{\Gamma(\nu)} + \log d_t \right) \tag{B.4}$$

Observing that some of the terms in the sums in (B.3) and (B.4) are independent of the time variable t one therefore finds the following reestimation formulae, which were already quoted in section 2.4.2.

$$\frac{\nu}{\eta} = \frac{\sum_t \gamma_t(i) d_t}{\sum_t \gamma_t(i)} \tag{B.5}$$

$$\psi(\nu) - \log \nu = \frac{\sum_t \gamma_t(i) \log d_t}{\sum_t \gamma_t(i)} - \log \left(\frac{\sum_t \gamma_t(i) d_t}{\sum_t \gamma_t(i)} \right) \tag{B.6}$$

where $\psi(\nu) = \Gamma'(\nu)/\Gamma(\nu)$ is the digamma function. The interpretation of equation (B.5) is obvious. The left hand side of this equation is the mean of the Gamma distribution, while the right hand side is the mean estimated on the training data. One would therefore probably expect to obtain the following as the second reestimation formula

$$\frac{\nu}{\eta^2} = \frac{\sum_t \gamma_t(i)(d_t - m_i)^2}{\sum_t \gamma_t(i)} \tag{B.7}$$

Here the left hand side is the variance of the Gamma density and the right hand side is the variance estimated on the training data. In fact, this equation has also been used to fit a Gamma density to a set of samples (Burhstein, 1995). However, since this type of parameter estimation does not maximise the likelihood of the model it does, strictly speaking, not fit within the EM algorithm. It is interesting to note that the right and left hand side of equation (B.6) can be given a similar interpretation as (B.7), i.e. both sides can be interpreted as second order statistics of the distribution under consideration. To see this it is useful to introduce the following notation

$$p(t) = \frac{\gamma_t(i)}{\sum_t \gamma_t(i)} \tag{B.8}$$

The function p is therefore the probability of observing state i at time t given that it was observed anywhere in the utterance. Equation (B.6) can now be rewritten as

$$\begin{aligned} \sum_t p(t) \log d_t - \log\left(\sum_t p(t) d_t\right) &= \sum_t p(t) \log \frac{d_t}{\sum_t p(t) d_t} \\ &= \sum_t p(t) \log \frac{\frac{p(t)d_t}{\sum_t p(t)d_t}}{p(t)} \\ &= -D(p\|q) \end{aligned} \tag{B.9}$$

Here $D(p\|q)$ is the Kullback-Leibler distance between the probability distributions $p(t)$ and $q(t)$ where the latter is given by

$$q(t) = \frac{p(t)d_t}{\sum_t p(t)d_t} \tag{B.10}$$

This, in fact, defines a probability distribution since the d_t are all positive and therefore the $q(t)$ are positive, lie between zero and one and sum to one. The value $-D(p\|q)$ can now be interpreted as a variance measure, because it depends on the distribution of the d_t around their estimated mean $\sum_t \gamma_t(i) d_t$. The closer each individual d_t is to the average the closer the distribution q is to p. In fact if each d_t is the same as the mean then p and q are identical. Since equation (B.10) involves division by the estimated mean the difference between q from p is scaled by the magnitude of the mean. If the mean is therefore high the difference between d_t and the mean is therefore less weighted than for small means.

Interestingly, the left hand side of (B.6) can be seen to be just the continuous equivalent of this variance measure for the gamma distribution. One can see this from the subsequent calculations, where $b(d)$ denotes the gamma distribution.

$$\begin{aligned} \int_0^\infty b(d) \log \frac{d}{\frac{\nu}{\eta}} \mathbf{d}d &= -\log \frac{\nu}{\eta} \int_0^\infty b(d) \mathbf{d}d + \int_0^\infty b(d) \log(d) \mathbf{d}d \\ &= -\log \nu + \log \eta + \int_0^\infty b(d) \log(d) \mathbf{d}d \end{aligned} \tag{B.11}$$

Here, the fact that $b(d)$ is a probability distribution was used. Next the second term in (B.11) is calculated.

$$\begin{aligned}
\int_0^\infty b(d)\log(d)\mathbf{d}d &= \int_0^\infty \frac{\eta^\nu}{\Gamma(\nu)} d^{\nu-1}e^{-\eta d}\log(d)\mathbf{d}d \\
&= \frac{\eta^\nu}{\eta\Gamma(\nu)}\int_0^\infty \left(\frac{d}{\eta}\right)^{\nu-1} e^{-d}\log\frac{d}{\eta}\mathbf{d}d \\
&= \frac{1}{\Gamma(\nu)}\left(\underbrace{\int_0^\infty d^{\nu-1}e^{-d}\log(d)\mathbf{d}d}_{\Gamma'(\nu)} - \log\eta \underbrace{\int_0^\infty d^{\nu-1}e^{-d}\mathbf{d}d}_{\Gamma(\nu)}\right) \\
&= \psi(\nu) - \log\eta
\end{aligned}$$

Adding the last expression to (B.11), finally, gives the left hand side of (B.6), which proves the assertion.

In summary, the ML estimation of the Gamma density parameters involves the solution of one first and one second order statistics equations where the latter replaces the normal variance concept with another one that is special to the Gamma densities.

C

Existence and uniqueness results for the softmax parameter estimation problem

As mentioned earlier in chapter 5, the softmax functions which are used in the SBME-HMM to model the probabilities of states of the hidden variable v have a number of very useful properties. This appendix will explore the shape of their error surfaces and will show that, unlike for the probability models discussed in sections 5.3.1 and 5.3.2, here the existence of unique global extrema in the error surface can be established.

First the relevant theorems will be stated together with two corollaries and will later be proved in sections C.1 and C.2.

The error function of choice in this chapter is the cross-entropy error since this is the one most commonly used in connection with softmax functions(Bishop, 1995; Bridle, 1989). As pointed out in section 5.2, the cross-entropy error is related to the log likelihood of the model and maximising the latter is equivalent to minimising the former. In order to show the existence of a unique set of softmax parameters which maximises the auxiliary function of an SBME-HMM, it is therefore necessary to prove the existence of a unique global minimum of the cross-entropy error function.

To put the discussion in a more general frame-work, which does not rely on the HMM formulation in chapter 5, the cross-entropy error function will be denoted by $E(x, p, w, \mathbf{c})$ and is defined to be

$$E(x, p, w, \mathbf{c}) = \sum_{n=1}^{N} w_n \sum_{k=1}^{K} p_{n,k} \log\left(\frac{p_{n,k}}{S_k(x_n)}\right) \tag{C.1}$$

Here $x = (x_1, \dots, x_N)$ is a set of data points each of which has a set of probabilities $p_{n,k}$, $k = 1, \dots, K$ associated with it for the K different classes of the classification problem. These probabilities indicate how likely it is that a data point x_n belongs to one of the K classes. The w_n in (C.1) are positive weights and $\mathbf{c}$ is the set of softmax parameters. The data points x_n in C.1 correspond to the d_t in 5.21, while the $p_{n,k}$ and w_n correspond to the $\gamma_t(i, k)$ and $\gamma_t(i)$, respectively. This formulation is slightly more general than the error function in (5.22) in that it does not assume that the w_n and $p_{n,k}$ are both derived by the same estimation process.

For the softmax functions the same notation will be used as in chapter 5. Here, however, the exponents of the softmax functions do not necessarily have to be polynomials. Instead they can be linear combinations of arbitrary functions of the data points x. The exponents considered will therefore have the following form

$$q_j(x) = \sum_{l=0}^{L_j} c_{j,l}\phi_{j,l}(x) + \phi_j(x) \tag{C.2}$$

and the vector of values of the $\phi_{j,l}$ for a given x_n and j will be denoted by

$$\Phi_j(x_n) = (\phi_{j,0}(x_n), \ldots, \phi_{j,L_j}(x_n)) \tag{C.3}$$

The first theorem in this appendix addresses the question whether the minima of the cross-entropy error surface, if such minima exist, are unique. This problem will be tackled by asserting that the error surface is convex. A function f is called convex if the following holds for any two points x_1 and x_2 in the region where f is defined.

$$f(tx_1 + (t-1)x_2) \leq tf(x_1) + (t-1)f(x_2) \tag{C.4}$$

Here $t < 1$ and $x_1 t + (t-1)x_2$ is therefore a point on the line connecting x_1 and x_2. The theorem asserting the convexity of the cross-entropy error function can now be stated as follows.

Theorem 1 *The cross-entropy error function $E(x, p, w, \mathbf{c})$ of a K-class problem of arbitrary finite dimension, regarded as a function of the model parameters* $\mathbf{c}$*, is always convex. If there exists a j with $1 \leq j \leq K-1$ for which there are at least $L_j + 1$ data points x_n such that the $\Phi_j(x_n)$ are linearly independent then the error function $E(x, p, w, \mathbf{a})$ is strictly convex.*

An immediate consequence of this theorem is the following corollary.

Corollary 1 *The error function $E(x, p, w, \mathbf{c})$ of a one-dimensional K-class problem is convex if the $q_j(x)$ are polynomials and strictly convex if there exist at least* $\min_j L_j + 1$ *different x_n.*

The reason why strict convexity is important for ensuring unique minima is that because of equation (C.4), for any two global minima every point on the line connecting the two has to be a minimum as well. If the inequality in (C.4) is furthermore strict then there cannot be two global minima, otherwise any point connecting these two would have a function value below the minimum, which is, obviously, a contradiction.

The last theorem showed that under special conditions the minima of the cross-entropy error surface, if such minima exist, have to be unique. The next theorem addresses the problem of determining when such minima exist. Together both theorems therefore answer the question under which conditions there exist unique global minima in the cross-entropy error surface.

Theorem 2 *Let (x, p, w) be a multi-class problem of arbitrary finite dimension. Suppose none of the $p_{n,j}$ are zero and that there exists for each set of functions $q_j(x)$, $j = 1, \ldots, K-1$, and each x_n at least one j for which $q_j(x_n) \neq 0$ then there exists a point* $\mathbf{c}$ *in the parameter space which minimises $E(x, p, w, \mathbf{c})$. If, furthermore, there exists at least one j for which there are $L_j + 1$ data points x_n such that the $\Phi_j(x_n)$ are linearly independent then this solution is unique.*

This theorem has again the following useful corollary.

Corollary 2 *Let (x, p, w) be a one-dimensional K-class problem and suppose there exist at least $\min_j L_j + 1$ different x_n such that $p_{n,j} \neq 0$ for all these j. Then there exists exactly one point $\mathbf{c}$ in the parameter space which minimises $E(x, p, w, \mathbf{c})$.*

C.1 Proof of Theorem 1

This section will proof the convexity of the cross-entropy error function by establishing that the Hessian of this function is always positive definite.

In section 5.2 the gradient of the cross-entropy error function was shown to be

$$\frac{\partial}{\partial c_l^{v,s}} E(x, p, w, \mathbf{c}) = \sum_{n=1}^{N} w_n \sum_{k=1}^{K} \left(p_{n,k} - S_k(x_n)\right) \phi_l(x_n) \tag{C.5}$$

In order to determine the form of the Hessian of the cross-entropy error function, it is therefore necessary to calculate the partial second order derivatives of the softmax functions. This can be accomplished as follows.

$$\frac{\partial}{\partial a_{j,l}} S_k(x) = \frac{e^{q_k(x)} \frac{\partial}{\partial a_{j,l}} q_k(x)(1 + \sum_{k=1}^{K-1} e^{q_k(x)}) - e^{q_k(x)} \sum_{k=1}^{K-1} e^{q_k(x)} \frac{\partial}{\partial a_{j,l}} q_k(x)}{(1 + \sum_{k=1}^{K-1} e^{q_k(x)})^2} \tag{C.6}$$

For $k = j$, equation (C.6) becomes

$$\frac{\partial}{\partial a_{j,l}} S_j(x) = \left(\frac{e^{q_j(x)}}{1 + \sum_{k=1}^{K-1} e^{q_k(x)}} - \frac{(e^{q_j(x)})^2}{(1 + \sum_{k=1}^{K-1} e^{q_k(x)})^2} \right) \phi_{j,l}(x) \tag{C.7}$$

and for $k \neq j$ equation (C.6) reduces to

$$\frac{\partial}{\partial a_{j,l}} S_k(x) = -\frac{e^{q_k(x)} e^{q_j(x)}}{(1 + \sum_{k=1}^{K-1} e^{q_k(x)})^2} \phi_{j,l}(x) \tag{C.8}$$

Substituting the definitions of the softmax functions, these last two equations can be reformulated as follows

$$\frac{\partial}{\partial a_{j,l}} S_k(x) = \begin{cases} (S_k(x) - S_k(x)^2)\phi_{j,l}(x) & : \quad j = k \\ \\ -S_j(x) S_k(x) \phi_{j,l}(x) & : \quad j \neq k \end{cases} \tag{C.9}$$

This shows that the second order derivatives of the error function $E(x, p, w, \mathbf{a})$ are given by

$$\frac{\partial}{\partial c_{j,l} c_{k,l'}} E(x, p, w, \mathbf{c}) = \begin{cases} \sum_{n=1}^{N} w_n (S_k(x_n) - S_k(x_n)^2) \phi_{k,l}(x_n) \phi_{k,l'}(x_n) & : \quad j = k \\ \\ -\sum_{n=1}^{N} w_n S_k(x_n) S_j(x_n) \phi_{j,l}(x_n) \phi_{k,l'}(x_n) & : \quad j \neq k \end{cases} \tag{C.10}$$

For the special case where the $q_j(x)$ are polynomials, equation (C.10) reduces to

$$\frac{\partial}{\partial a_{j,l} a_{k,l'}} E(x, p, w, \mathbf{a}) = \begin{cases} \sum_{n=1}^{N} w_n (S_k(x_n) - S_k(x_n)^2) x_n^{l+l'} & : \quad j = k \\ \\ -\sum_{n=1}^{N} w_n S_k(x_n) S_j(x_n) x_n^{l+l'} & : \quad j \neq k \end{cases} \tag{C.11}$$

Equation (C.10) gives the components of the Hessian of $E(x,p,w,\mathbf{a})$ which will be denoted by $H(x,p,w,\mathbf{a})$ in the following. This will also sometimes be abbreviated with H if the particular classification problem (x,p,w) is not important. One can see from (C.10) that for a two class problem with a single exponential function $q(x)$ the second order derivative of the error function $E(x,p,w,\mathbf{a})$ can be written as the sum of simpler matrices, i.e. $H = \sum_{n=1}^{N} H_n$, where H_n is given by

$$H_n = h_n \Phi(x_n)' \Phi(x_n) \tag{C.12}$$

Here h_n is the scalar value

$$h_n = w_n(S_1(x_n) - S_1(x_n)^2) = w_n S_1(x_n) S_2(x_n) \tag{C.13}$$

and $\Phi(x_n)$ is the vector in (C.3) for $q(x)$. Since w_n is positive and $0 < S_k(x) < 1$ holds for all x, the h_n in (C.13) are always positive. This immediately implies that each H_n is positive semi-definite and since H is the sum of the H_n it too is positive semi-definite. If there are $L+1$ data points x_n such that the $\Phi(x_n)$ are linearly independent then H is strictly positive definite. These considerations show that the error surface $E(x,p,w,\mathbf{a})$ for a two-class problem of arbitrary finite dimension is always convex and strictly convex if there $L+1$ linearly independent $\Phi(x_n)$ and therefore proves theorem 1 in the case of a two class problem.

Also for a K-class problem with $K > 2$ the Hessian can be written as a sum of simpler matrices H_n. However, these matrices are slightly more complex than in the two class case. For instance, for a 3-class problem, according to equation (C.10), the H_n are block matrices of the following form

$$H_n = \begin{pmatrix} H_{n11} & H_{n12} \\ H'_{n12} & H_{n22} \end{pmatrix} \tag{C.14}$$

where the blocks H_{nij} are given by

$$H_{nii} = (S_i(x_n) - S_i(x_n)^2)\Phi_i(x_n)'\Phi_i(x_n) \tag{C.15}$$

$$H_{n12} = -S_1(x_n)S_2(x_n)\Phi_1(x_n)'\Phi_2(x_n) \tag{C.16}$$

and i can take the values 1 or 2. These considerations generalise straight-forwardly to a K-class problem with arbitrary $K > 2$. In this case the matrices $H_n = (H_{nij})_{i,j=1}^{K-1}$ are block matrices whose blocks H_{nij} are given by

$$H_{nij} = h_{nij}\Phi_i(x_n)'\Phi_j(x_n) \tag{C.17}$$

with the scalar h_{nij} defined as follows

$$h_{nij} = \begin{cases} w_n(S_i(x_n) - S_i(x_n)^2) & : \quad i = j \\ \\ -w_n S_i(x_n) S_j(x_n) & : \quad i \neq j \end{cases} \tag{C.18}$$

In order to prove the positive semi-definiteness of H_n it has to be shown that $vH_nv' \geq 0$ for arbitrary vectors v of dimension $K - 1 + \sum_{j=1}^{K-1} L_j$. For a 3-class problem the vector v can be written as $v = (v_1, v_2)$ where v_1 are the first $L_1 + 1$ components and v_2 are the remaining $L_2 + 1$ components. The positive semi-definiteness of H_n can therefore be shown by proving that the following is true.

$$h_{n11} < \Phi_1(x_n), v_1 >^2 + 2h_{n12} < \Phi_1(x_n), v_1 >< \Phi_2(x_n), v_2 > + h_{n22} < \Phi_2(x_n), v_2 >^2 \geq 0 \quad \text{(C.19)}$$

Since the inner products $< \Phi_1(x_n), v_1 >$ and $< \Phi_2(x_n), v_2 >$ can take arbitrary values, proving inequality (C.19) is equivalent to showing that the following matrix is positive semi-definite.

$$\mathbf{h}_n = \begin{pmatrix} h_{n11} & h_{n12} \\ h_{n12} & h_{n22} \end{pmatrix} \quad \text{(C.20)}$$

Again, these considerations can be easily generalised to a K-class problem with more than 3 classes. In this case, proving that the individual terms H_n of the second order derivative are positive semi-definite is equivalent to showing that the matrix $\mathbf{h}_n = (h_{nij})_{i,j=1}^{K-1}$ is positive semi-definite. This can be done by observing that the matrix $\mathbf{h}_n$ is diagonally dominant which can be seen as follows.

$$\begin{aligned} |h_{nii}| &= S_i(x_n) - S_i(x_n)^2 = S_i(x_n)(1 - S_i(x_n)) = S_i(x_n) \sum_{j=1, j \neq i}^{K} S_j(x_n) \\ &> S_i(x_n) \sum_{j=1, j \neq i}^{K-1} S_j(x_n) = \sum_{j=1, j \neq i}^{K-1} |h_{nij}| \end{aligned} \quad \text{(C.21)}$$

Since, furthermore, the diagonal elements of $\mathbf{h}_n$ are positive the matrix is positive definite. If there exists a j with $1 \leq j \leq K - 1$ for which there are $L_j + 1$ data points x_n such that the $\Phi_j(x_n)$ are linearly independent then the second order derivative of the error function $E(x, p, w, \mathbf{a})$ is positive definite. This concludes the proof of Theorem 1.

It is interesting to note that in the proof of the second theorem in (Jordan and Xu, 1995) Jordan and Xu calculate the Hessian of the cross-entropy error function. However, they arrive at an expression which is somewhat different form the one given here because they do not consider the situation where $j \neq k$ in (C.9). This therefore leads them to assume that the Hessian always has the same form as in the two class case. Since this is wrong for more than 2 classes their proof is therefore, strictly speaking, only valid for $K = 2$.

C.2 Proof of Theorem 2

To show the existence of a global minimum of the error function it is sufficient to prove that there exists a local minimum along each direction in the parameter space. This together with the fact that the set of directions is compact and taking the minimum of the error function along each direction is a continuous operation will show that there exists a global minimum of the error function.

To show that there exists a minimum along each direction it is necessary to observe that the error function restricted to a particular direction in parameter space becomes a one-dimensional convex function. One therefore only has to show that the error function tends to infinity if the one-dimensional parameter goes to infinity or minus infinity. For a K-class problem of arbitrary finite dimension this can be done as follows. Let $q_j(x)$, $j = 1, \dots, K-1$, be an arbitrary set of polynomials such that not all of them are identically zero, then the direction specified by the $q_j(x)$ is the set of polynomials given by $cq_j(x)$, $j = 1, \dots, K-1$, where c is a real number. Suppose there exists an x_n, such that $q_j(x_n) \neq 0$ for at least one j. Without restricting the generality of the proof one can assume that $q_j(x_n) > 0$. Now let J be defined by $J = \arg\max_j q_j(x_n)$. Then $q_J(x_n) > 0$ and therefore the following holds true.

$$\frac{e^{cq_J(x_n)}}{1+\sum_{j=1}^{K-1} e^{cq_j(x_n)}} \to 1 \quad \text{for} \quad c \to \infty \tag{C.22}$$

This implies that $S_j(x_n) \to 0$ if $j \neq J$. Similarly, one has $S_J(x_n) \to 0$ for $c \to -\infty$. This implies that for $|c| \to \infty$ there is always one j such that $S_j(x_n)$ converges to zero. For such a j the following therefore holds, assuming that $p_{n,j} \neq 0$.

$$p_{n,j} \log\left(\frac{p_{n,j}}{S_j(x_n)}\right) \to \infty \tag{C.23}$$

And as a result the error function $E(x, p, w, \mathbf{a})$ tends to infinity. This concludes the proof of Theorem 2.

Bibliography

Abramovitz, M. and Stegun, J. A., editors (1965). *Handbook of Mathematical Functions*. New York: Dover Publications, Inc.

Anastasakos, A., Schwartz, R., and Shu, H. (1995). Duration modeling in large vocabulary speech recognition. In *Proc. ICASSP,* pages 628–631.

Bahl, L. R., Brown, P. F., de Souza, P. V., and Mercer, R. L. (1986). Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proc. ICASSP*, pages 49–52.

Bahl, L. R., Brown, P. F., de Souza, P. V., and Mercer, R. L. (1993). Estimating hidden Markov model parameters so as to maximise speech recognition accuracy. *Trans. IEEE Speech and Audio Processing*, 1(1):77–83.

Baum, L., Petrie, T., Soules, G., and Weiss, N. (1970). A maximisation technique in the statistical analysis of probabilistic functions of finite Markov chains. *Annals of Mathematical Statistics*, 41:164–171.

Bengio, Y. and Frasconi, P. (1995). An input output HMM architecture. In *Advances in Neural Information Processing Systems 7*, pages 427–434.

Bilmes, J. A. (1998). Data-driven extensions to HMM statistical dependencies. In *Proc. ICSLP*, volume 1, pages 69–72.

Bilmes, J. A. (1999). Buried markov models for speech recognition. In *Proc. ICASSP*, volume 2, pages 713–716.

Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford.

Blackburn, C. S. and Young, S. J. (1995). Towards improved speech recognition using a speech production model. In *Proc. Eurospeech*, pages 1623–1626.

Bourlard, H., Dupont, S., and Ris, C. (1996). Multi-stream speech recognition. Technical Report IDIAP-RR 96-07, IDIAP, Martigny-Valais-Suisse.

Bourlard, H., Konig, Y., and Morgan, M. (1995). REMAP: Recursive Estimation and Maximization of A Posteriori Probabilities in connectionist speech recognition. In *Proc. Eurospeech*.

Bourlard, H. and Morgan, N. (1994). *Connectionist Speech Recognition - A Hybrid Approach*. Kluwer Academic Publishers.

Bourlard, H. and Morgan, N. (1997). Hybrid HMM/ANN Systems for Speech Recognition: Overview and New Research Directions. In *Internation School on Neural Nets: Adaptive Processing of Temporal Information*.

Breiman, L., Friedman, J., Ohlsen, R., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth and Brooks.

Bridle, J. S. (1989). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Fougelman-Soulie, F. and Herault, J., editors, *Neurocomputing: Algorithms, Architectures and Applications*, pages 301–307. Springer Verlag.

Bridle, J. S., Deng, L., Picone, J., Richards, H. B., Ma, J., Kamm, T., Schuster, M., Pike, S., and Regan, R. (1998). An investigation of segmental hidden dynamic models of speech coarticulation for automatic speech recognition. Technical report, LVCSR Summer Research Workshop, Johns Hopkins University.

Burhstein, D. (1995). Robust parametric modeling of durations in hidden Markov models. In *Proc. ICASSP*, pages 548–551.

Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405.

Cover, T. M. and Thomas, J. A. (1991). *Elements of information theory*. John Wiley,New York.

Dagum, P. and Luby, M. (1993). Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60:141–153.

Dean, T. and Kanazawa, K. (1988). Probabilistic temporal reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*.

Deller, J. R., Proakis, J. G., and Hansen, J. H. L. (1993). *Discrete-Time Processing of Speech Signals*. Prentice Hall.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1 – 38.

Donovan, R. (1996). *Trainable speech synthesis*. PhD thesis, Cambridge University Engineering Department.

Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. John Wiley.

Dupont, S. and Bourlard, H. (1996). Multiband approach for speech recognition. In *Proc. ProRISC/IEEE Workshop on Circuits, Systems and Signal Processing*, pages 113–118.

Dupont, S. and Luettin, J. (1998). Using the multi-stream approach for continuous audio-visual speech recognition: Experiments on the M2VTS database. In *Proc. ICSLP*, volume 2, pages 1283–1286.

Eide, E. and Gish, H. (1996). A parametric approach to vocal tract length normalization. In *Proc. ICASSP*, pages 346–349.

Evermann, G. and Woodland, P. C. (2000). Large vocabulary decoding and confidence estimation using word posterior probabilities. In *Proc. ICASSP*.

Fant, C. G. (1973). *Speech Sounds and Features*. MIT Press.

Fanty, M. and Cole, R. (1991). Spoken letter recognition. In *Advances in Neural Information Processing Systems*, volume 3, pages 220–226.

Finke, M. and Waibel, A. (1997). Speaking mode dependent pronunciation modeling in large vocabulary conversational speech recognition. In *Proc. Eurospeech*, pages 2379–2382.

Fletcher, H. (1940). Auditory patterns. *Review of Modern Physics*, 12:47–65.

Fletcher, H. and Munson, W. A. (1933). Loudness, its definition, measurement and calculation. *Journal of the Acoustical Society of America*, 5:82–108.

Fukunaga, K. (1990). *Introduction to statistical pattern recognition*. Academic Press.

Gales, M. and Young, S. J. (1993a). Segmental HMMs for speech recognition. In *Proc.European Conf. on Speech Commun. and Technology*, pages 1579–1582.

Gales, M. and Young, S. J. (1993b). The theory of segmental hidden Markov models. Technical Report CUED/F-INFENG/TR.133, Cambridge University Engineering Department.

Gales, M. J. F. (1999). Semi-tied covariance matrices for hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, 7:272–281.

Ghahramani, Z. and Hinton, G. (1996). Switching state-space models. Technical Report CRG-TR-96-3, Department of Computer Science, University of Toronto.

Ghahramani, Z. and Jordan, M. I. (1997). Factorial hidden Markov models. *Machine Learning*, 29:245–275.

Gillick, L. and Cox, S. (1989). Some statistical issues in the comparison of speech recognition algorithms. In *Proc. ICASSP*, pages 532–535.

Gish, H. and Ng, K. (1993). A segmental speech model with applications to word spotting. In *Proc. ICASSP*, pages 447–450.

Gish, H. and Ng, K. (1996). Parametric trajectory models for speech recognition. In *Proc. ICSLP*, pages 466–469.

Gish, H. and Schmidt, N. (1994). Text-independent speaker identification. *IEEE Signal Processing Magazine*, pages 18–32.

Gopinath, R. A. (1998). Maximum likelihood modeling with Gaussian distributions for classification. In *Proc. ICASSP*.

Gravier, G., Sigelle, M., and Chollet, G. (1998). Toward Markov random field modelling of speech. In *Proc. ICSLP*, pages 1807–1810.

Haeb-Umbach, R. and Ney, H. (1992). Linear discriminant analysis for improved large vocabulary continuous speech recognition. In *Proc. ICASSP*, pages 13–16.

Hagan, M. T. and Menhaj, M. (1994). Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993.

Hagen, A., Morris, A., and Bourlard, H. (1998). Subband-based speech recognition in noisy conditions: The full combination approach. Technical Report IDIAP-RR 98-15, IDIAP, Martigny-Valais-Suisse.

Hain, T., Johnson, S. E., Tuerk, A., Woodland, P. C., and Young, S. J. (1998). Segment generation and clustering in the HTK broadcast news transcription system. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pages 133–137.

Hermansky, H. (1990). Perceptual linear predictive analysis of speech. *Journal of the Acoustical Society of America*, 87(4):1738–1752.

Hinton, G. E. and Brown, A. D. (2001). Training many small hidden Markov models. In *Proc. of the Instiute of Acoustics*, volume 23, pages 1–17.

Iyer, R., Gish, H., Siu, M.-H., Zavaliagkos, G., and Matsoukas, S. (1998). Hidden Markov models for trajectory modelling. In *Proc. ICSLP*, pages 1811–1814.

Jacobs, R. A., Jordan, M. I., Nowland, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3:79–87.

Jensen, F. V., Lauritzen, S. L., and Olesen, K. G. (1990). Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 5:269–282.

Jones, M. and Woodland, P. C. (1993). Using relative duration in large vocabulary speech recognition. In *Proc. Eurospeech*, pages 311–314.

Jordan, M. I., Ghahramani, Z., and Saul, L. K. (1997). Hidden Markov decision trees. In *Advances in Neural Information Processing Systems 9*, pages 501–507.

Jordan, M. I. and Jacobs, R. A. (1992). Hierarchies of adaptive experts. *Advances in Neural Information Processing Systems 4*, pages 985–992.

Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214.

Jordan, M. I. and Xu, L. (1995). Convergence results for the EM approach to mixtures of experts architectures. *Neural Networks*, 8(9):1409–1431.

Juang, B. H. and Katagiri, S. (1992). Discriminative training. *Journal of the Acoustic Society of Japan*, 13(6):333–340.

Junqua, J.-C. (1993). The Lombard reflex and its role on human listeners and automatic speech recognizers. *Journal of the Acoustical Society of America*, 93(1):510–524.

Kendall, M. G. and Stuart, A. (1973a). *The Advanced Theory of Statistics*, volume 2. Griffin, London, 3rd edition.

Kendall, M. G. and Stuart, A. (1973b). *The Advanced Theory of Statistics*, volume 3. Griffin, London, 3rd edition.

Kendall, M. G. and Stuart, A. (1973c). *The Advanced Theory of Statistics*, volume 1. Charles Griffin.

Kullback, S. and Leibler, R. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86.

Lass, H. and Gottlieb, P. (1971). *Probability and Statistics*. Addison-Wesley.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society B*, 50:157–224.

Lee, L. and Rose, R. C. (1996). Speaker normalization using efficient frequency warping procedures. In *Proc. ICASSP*, pages 353–356.

Levinson, S. (1986). Continuously variable duration hidden Markov models for automatic speech recognition. *Computer Speech and Language*, 1:29 – 45.

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

Liu, D., Nguyen, L., Matsoukas, S., Davenport, J., Kubala, F., and Schwartz, R. (1998). Improvements in spontaneous speech recognition. In *DARPA Broadcast News Transcription and Understanding Workshop*.

Logan, B. and Moreno, P. (1998). Factorial HMMs for acoustic modeling. In *Proc. ICASSP*.

MacKay, D. J. C. (1992). The evidence framework applied to classification networks. *Neural Computation*, 4(5):720–736.

Makhoul, J. (1975). Spectral linear prediction: Properties and applications. *IEEE Transactions ASSP*, 23:283–296.

Makhoul, J. and Cosell, L. (1976). LPCW: An LPC vocoder with linear predictive spectral warping. In *Proc. ICASSP*, pages 466–469.

Mangu, L., Brill, E., and Stolcke, A. (1999). Finding consensus among words: Lattice-based word error minimization. In *Proc. Eurospeech*.

McCullagh, P. and Nelder, J. A. (1989). *Generalised Linear Models*. Chapman and Hall, London.

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

Merwe, C. J. and du Preez, J. A. (1991). Calculation of LPC based cepstrum coefficients using mel-scale frequency warping. In *Proc. IEEE COMSIG*, pages 17–21.

Mirghafori, N., Fosler, E., and Morgan, N. (1995a). Fast speakers in large vocabulary continuous speech recognition: analysis and antidotes. In *Proc. Eurospeech*, pages 491–494.

Mirghafori, N., Fosler, E., and Morgan, N. (1995b). Making automatic speech recognition more robust to fast speech. Technical Report TR-95-067, ICSI.

Mirghafori, N., Fosler, E., and Morgan, N. (1996). Towards robustness to fast speech in ASR. In *Proc.ICASSP*, pages 335–338.

Monkowski, M. D., Picheny, M. A., and Rao, P. S. (1995). Context dependent phonetic duration models for decoding conversational speech. In *Proc. ICASSP*, pages 528–531.

Morgan, N., Fosler, E., and Mirghafori, N. (1997). Speech recognition using on-line estimation of speaking rate. In *Proc.Eurospeech*, volume 4, pages 2079 – 2082.

Morgan, N. and Fossler-Lussier, E. (1998). Combining multiple estimators of speaking rate. In *Proc. ICASSP*, pages 729–732.

Morris, A. (1998). Latent variable decomposition for posteriors or likelihood based subband ASR. Technical Report IDIAP-RR 99-04, IDIAP, Martigny-Valais-Suisse.

Morris, A., Hagen, A., Glotin, H., and Bourlard, H. (2000). Multi-stream adaptive evidence combination for noise robust ASR. Technical Report IDIAP-RR 99-26, IDIAP, Martigny-Valais-Suisse.

Nguyen, D. and Widrow, B. (1990). Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, pages 18–23.

Nowlan, S. J. and Hinton, G. E. (1991). Evaluation of adaptive mixtures of competing experts. *Advances in Neural Information Processing Systems 3*.

Ostendorf, M., Byrne, W., Bacchiani, M., Finke, M., Gunawardana, A., Ross, K., Roweis, S., Shriberg, E., Talkin, D., Waibel, A., Wheatley, B., and Zeppenfeld, T. (1996). Modelling systematic variations in pronunciation via a language-dependent hidden speaking mode. Technical report, LVCSR Summer Research Workshop, Johns Hopkins University.

Pallett, D. S., Fiscus, J. G., Fisher, W. M., Garofolo, J. S., Lund, B. A., and Przybocki, M. A. (1994). 1993 WSJ-CSR benchmark test results. In *Proc. ARPA's Spoken Language Systems Technology Workshop*.

Pallett, D. S., Fiscus, J. G., Garofolo, J. S., Martin, A., and Przybocki, M. A. (1999). 1998 broadcast news benchmark test results. In *Proc. DARPA Broadcast News Workshop*.

Pallett, D. S., Fiscus, J. G., Martin, A., and Przybocki, M. A. (1998). Broadcast news benchmark test results: English and non-English. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*.

Pallett, D. S., Fiscus, J. G., and Przybocki, M. A. (1997). 1996 Preliminary Broadcast News benchmark tests. In *Proc. DARPA Speech Recognition Workshop*.

Pallett, D. S., Fisher, W. M., and Fiscus, J. G. (1990). Tools for the analysis of benchmark speech recognition tests. In *Proc. ICASSP*, pages 97–100.

Papoulis, A. (1984). *Probability, random variables and stochastic processes*. McGraw-Hill.

Patterson, R. D. (1976). Auditory filter shapes derived with noise stimuli. *Journal of the Acoustical Society of America*, 59:640–654.

Patterson, R. D. and Nimmo-Smith, I. (1980). Off-frequency listening and auditory filter asymmetry. *Journal of the Acoustical Society of America*, 67:229–245.

Paul, D. (1997). Extensions to phone-state decision-tree clustering: Single tree and tagged clustering. In *Proc. ICASSP*, pages 1487–1490.

Peterson, G. E. and Barney, H. L. (1952). Control methods used in a study of vowels. *Journal of the Acoustical Society of America*, 24:175–184.

Pfau, T. and Ruske, G. (1998a). Creating hidden Markov models for fast speech. In *Proc. ICSLP*, pages 205–208.

Pfau, T. and Ruske, G. (1998b). Estimating the speaking rate by vowel detection. In *Proc. ICASSP*, pages 945–948.

Pfitzinger, H. R. (1996). Two approaches to speech rate estimation. In *Proc. SST*, pages 421–426.

Pfitzinger, H. R. (1998). Local speech rate as a combination of syllable and phone rate. In *Proc. ICSLP*.

Pfitzinger, H. R., Burger, S., and Heid, S. (1996). Syllable detection in read and spontaneous speech. In *Proc. ICSLP*, pages 1261–1264.

Picone, J., Pike, S., Kamm, T., Bridle, J., Deng, L., Ma, Z., Richards, H., and Schuster, M. (1999). Initial evaluation of hidden dynamic models on conversational speech. In *Proc. ICASSP*, pages 2339–2342.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1997). *Numerical Recipes in C*. Cambridge University Press.

Rabiner, L. and Juang, B. (1993). *Fundamentals of Speech Recognition*. Prentice Hall.

Rainton, D. and Sagayama, S. (1992). Appropriate error criterion for continuous speech HMM minimum error training. In *Proc. ICSLP*, pages 233–236.

Redner, R. A. and Walker, H. F. (1984). Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239.

Reichl, W. and Chou, W. (1999). A unified approach of incorporating general features in decision-tree based acoustic modeling. In *Proc. ICASSP*, pages 573–576.

Richards, H. B. and Bridle, J. S. (1999). The HDM: A segmental hidden dynamic model of coarticulation. In *Proc. ICASSP*, pages 1930–1933.

Robinson, A. J. (1994). An application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks*, 5(2):298–305.

Robinson, A. J. and Fallside, F. (1991). A recurrent error propagation speech recognition system. *Computer Speech and Language*, 5:259–274.

Robinson, D. W. and Dadson, R. S. (1956). A redetermination of the equal-loudness relations for pure tones. *British Journal of Applied Physics*, 7:166–181.

Saon, G., Padmanabhan, M., Gopinath, R., and Chen, S. (2000). Maximum likelihood discriminant feature spaces. In *Proc. ICASSP*.

Schroeder, M. R. (1977). Recognition of complex acoustic signals. *Life Sciences Research Report*, 5:324.

Schukat-Talamazzini, E. G., Hornegger, J., and Niemann, H. (1995). Optimal linear feature space transformations for semi-continuous hidden Markov models. In *Proc. ICASSP*, pages 369–372.

Shafran, I. and Ostendorf, M. (2000). Use of higher level linguistic structure in acoustic modeling for speech recognition. In *Proc. ICASSP*.

Shriberg, E., Bates, R., and Stolcke, A. (1997). A prosody only decision-tree model for disfluency detection. In *Proc. Eurospeech*, pages 2383–2386.

Shriberg, E. and Stolcke, A. (1996). Word predictability after filled pauses: A corpus-based study. In *Proc.ICSLP*, pages 1868–1871.

Siegler, M. A., Jain, U., Raj, B., and Stern, R. M. (1997). Automatic Segmentation, Classification and Clustering of Broadcast News Data. In *Proc. DARPA Speech Recognition Workshop*, pages 97–99.

Siegler, M. A. and Stern, R. M. (1995). On the effects of speech rate in large vocabulary speech recognition systems. In *Proc. ICASSP*, pages 335–338.

Siu, M., Iyer, R., Gish, H., and Quillen, C. (1998). Parametric trajectory mixtures for LVCSR. In *Proc. ICSLP*.

Stephenson, T. A., Bourlard, H., Bengio, S., and Morris, A. C. (2000). Automatic speech recognition using dynamic Bayesian networks with both acoustic and articulatory variables. In *Proc. ICSLP*.

Stevens, S. S. (1936). A scale for the measurement of a psychological magnitude: loudness. *Psychological Review*, 43:405–416.

Stevens, S. S. (1957). On the psychophysical law. *Psychological Review*, 64:153–181.

Stevens, S. S. and Volkmann, J. (1940). The relation of pitch to frequency: A revised scale. *American Journal of Psychology*, 53:329–353.

Suaudeau, N. and Andre-Obrecht, R. (1994). An efficient combination of acoustic and suprasegmental informations in a speech recognition system. In *Proc. ICASSP*, pages 65–68.

Tomita, M. (1982). Dynamic construction of finite-state automata from examples using hill-climbing. In *Proc. 4th Cognitive Science Conference*, pages 105–108.

Tuerk, A. and Young, S. J. (1999). Modelling speaking rate using a between frame distance metric. In *Proc. Eurospeech*, pages 419–422.

Tuerk, A. and Young, S. J. (2001a). Indicator variable dependent output probability modelling via continuous posterior functions. In *Proc. ICASSP*.

Tuerk, A. and Young, S. J. (2001b). Polynomial softmax functions for pattern classification. Technical Report CUED/F-INFENG/TR 402, Cambridge University Engineering Department.

Valtchev, V., Odell, J. J., Woodland, P. C., and Young, S. J. (1997). MMIE training of large vocabulary recognition systems. *Speech Communication*, 22:303–314.

Verhasselt, J. P. and Martens, J.-P. (1996). A fast and reliable rate of speech detector. In *Proc. ICSLP*, pages 2258–2261.

Waterhouse, S. (1997). *Classification and Regression using Mixtures of Experts*. PhD thesis, Cambridge University Engineering Department.

Waterhouse, S. R. and Robinson, A. J. (1994). Classification using hierarchical mixtures of experts. In *Proc. IEEE Workshop on Neural Networks for Signal Processing 4*, pages 177–186.

Watrous, R. L. and Kuhn, G. M. (1992). Induction of finite-state languages using second-order recurrent networks. *Neural Computation*, 4(3):406–414.

Weintraub, M., Taussig, K., Hunicke-Smith, K., and Snodgrass, A. (1996). Effect of speaking style on LVCSR performance. In *Proc. ICSLP, Addendum*, pages 16–19.

Wilks, S. (1962). *Mathematical Statistics*. John Wiley, New York.

Woodland, P. C., Hain, T., Johnson, S. E., Niesler, T. R., Tuerk, A., Whittaker, E. W. D., and Young, S. J. (1998). The 1997 HTK Broadcast News Transcription System. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pages 41–48.

Woodland, P. C., Hain, T., Moore, G. L., Niesler, T. R., Povey, D., Tuerk, A., and Whittaker, E. W. D. (1999). The 1998 HTK Broadcast News transcription system: Development and results. In *Proc. DARPA Broadcast News Workshop*, pages 265–270.

Xu, L. and Jordan, M. I. (1996). On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, 8(1):129–151.

Young, S., Kershaw, D., Odell, J., Ollason, D., and Valtchev, V. (2000). *The HTK Book*. Microsoft.

Zhan, P. and Westphal, M. (1997). Speaker normalization based on frequency warping. In *Proc. ICASSP*, pages 1039–1042.

Zhao, Y., Schwartz, R., Sroka, J., and Makhoul, J. (1995). Hierarchical mixtures of experts methodology applied to continuous speech recognition. In *Proc. ICASSP*, pages 3443–3446.

Zheng, J., Franco, H., Weng, F., Sankar, A., and Bratt, H. (2000). Word-level rate of speech modeling using rate-specific phones and pronunciations. In *Proc. ICASSP*, pages 1775–1778.

Zweig, G. (1998). *Speech Recognition with Dynamic Bayesian Networks*. PhD thesis, University of California, Berkeley.

Zweig, G. and Russell, S. (1998a). Probabilistic modeling with Bayesian networks for automatic speech recognition. In *Proc. ICSLP*, volume 7, pages 3011–3014.

Zweig, G. and Russell, S. (1998b). Speech Recognition with Dynamic Bayesian Networks. *American Association for Artifical Intelligence*, pages 173–180.

Zwicker, E. (1961). Subdivision of the audible frequency range into critical bands. *Journal of the Acoustical Society of America*, 33(2):248.