Aalborg University

Institute of Electronic Systems
Department of Communication Technology

Title:

Speaker Adaptation of Hidden Markov Models using Maximum Likelihood Linear Regression.

Project period:

01.02.96 - 04.06.96

Author:

Heidi Christensen

Supervisors:

Ove Andersen

Børge Lindberg

Number of pages: 111

Reports printed: 9

Abstract

The work presented in this report focuses on an essential problem when doing speaker adaptation; namely how effectively the speaker specific information in the adaptation data is used.
In the project a system has been implemented for speaker adaptation of hidden Markov models (HMM's) using the Maximum Likelihood Linear Regression (MLLR) method. MLLR is a method that transforms mixture components of HMM's by multiplying the mean vectors with a transformation matrix. It introduces the concept of regression classes as a set of mixture components that are transformed similarly. The adaptation technique is implemented in C. The data used in the tests are taken from the Danish EUROM.1 database. All results are averaged over ten speakers.
Three issues have been addressed: 1) the effect of varying the amount of adaptation material, 2) the effect of using different regression class divisions and 3) the importance of the phonetic content in the adaptation material. Tests show that the MLLR technique is very data effective. Only $3s$ of speech is needed when a diagonal transformation matrix is used before a positive effect of the adaptation is seen. When using a full matrix $5s$ are sufficient. It was observed that there is a high dependence between the achieved performance and the regression class division. Transforming each mono-phone individually improves the phoneme correctness from 50.9% for the initial speaker independent models to 58.8% for the adapted models. Based on several approaches it was concluded that there are differences in the speaker specific information available from different phonemes. Vowels were seen to vary a lot from speaker to speaker and to have relatively much influence on the effect of the adaptation. Fricatives on the other hand have very small inter speaker variances.

Preface

This thesis is part of the documentation of my MSc.E.E. specialising in speech technology. The project was carried out at the Institute of Electronic Systems, Aalborg University, Denmark during the period from 1st of February 1996 to 4th of June 1996.

The project addresses the problem of speaker adaptation of hidden Markov models using the Maximum Likelihood Linear Regression technique. Some prior knowledge of the theory of Markov models is assumed. The report is divided into four parts. The *Preliminaries* part contains the introductory sections including the definition of the project. In the *Theory* part theory, techniques and methods are described. The implementation and tests are described in the *Implementation, test and conclusion* part. The last part of the report is the *Appendix* where additional information is given.

Developed software is included on a floppy disk.

References to literature are written in brackets containing the authors surname followed by the year of publishing. Extensive information can be found in the bibliography.

A nomenclature list is provided in the back of the report. Here the used abbreviations are explained together with a list of the mathematical notation used throughout the thesis.

Aalborg, Denmark 4th June 1996

Heidi Christensen

Contents

Part I

Preliminaries

Chapter 1
Introduction

Achievement of a high performance is often the most dominating design criterion, when implementing speech recognition systems. One of the most commonly used recognition approaches is the hidden Markov modelling technique. Part of the method is a training session during which the parameters of the models are estimated using an amount of training material. When the models are sufficiently trained, the system is ready for use.

It is generally agreed that a speaker dependent system will give the best performance in applications involving a specific speaker. This requires, however, that enough training material is available for training the system from scratch. Obviously this is often not the case, and with the still increasing size of vocabulary that speech recognition systems operate with, it has become even more impossible to collect enough data from the new speaker to train a full large vocabulary speaker dependent recognition system. An often used solution is to train speaker independent systems using material from many speakers. But experiments have shown that using such systems in general involves obtaining a lower performance, than what is achievable with a speaker dependent system [Zavaliagkos *et al*, 1995], [Huang and Lee, 1991] and [Leggetter, 1995]. Figure 1.1 illustrates a solution to the problem: A speaker dependent system not trained from scratch, but by combining the general speech knowledge incorporated in a set of well trained models (speaker dependent or independent) with speaker specific information from a relatively small amount of data provided by the new speaker.

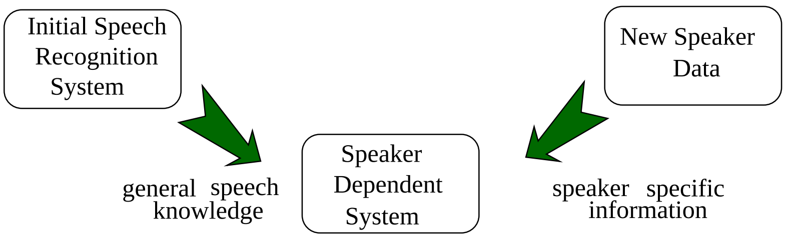

Figure 1.1: *A speaker dependent recognition system can be implemented using a combination of general speech knowledge from well trained hidden Markov models and speaker specific information from a new speaker's data.*

This process is generally known as speaker adaptation and the data available from the new speaker is referred to as the adaptation data.

In the recent years much research has been carried out in order to implement speaker specific systems following the approach described above. To improve the performance of the resulting speaker dependent system it has been studied, why the performance of a speaker independent system is in general lower than the performance of a speaker dependent system. One of the main reasons is that basically the speaker independent recogniser does in fact not model any speaker precisely, and there is a risk that a new speaker is characterised very badly with the models of the system. Also when a given speaker uses a speaker independent system, there is no way of utilising the fact that it is indeed, the same speaker that has uttered all the words, that are to be recognised.

Speaker variations are typically caused by differences such as different vocal tract sizes and shapes, different accents and different speaking styles. These are all physical inter-speaker variations, and it seems likely, that there will be some kind of speaker-specific pattern present in speech produced by a given speaker. More or less of this information must be implicit available in any speech example from the new speaker. It is therefore natural to try to extract the specific voice characteristics of the new speaker and use the knowledge to increase the recognition performance. For many specific applications the amount of adaptation data is often needed to be small.

Given the above considerations and the overall aim of an increased performance, it is reasonable to state that the success of a speaker adaptation technique lies in its ability to extract the maximum speaker specific information from a minimum amount of adaptation data.

The above interpretation of the problem of adapting a speech recogniser to a new speaker is the main focus of the work presented here. The thesis will concern speaker adaptation of recognisers that are based on the hidden Markov modeling technique. Before a more precise specification of the project is given, some of the principles of speaker adaptation techniques are presented in chapter 2. It will introduce the more general used terms and give a review of the main approaches to speaker adaptation. The last chapter in this preliminary part of the thesis, chapter 3, gives the exact definition of the contents of the project as well as an outline of the organisation of the remaining parts of the thesis.

Chapter 2

Review of speaker adaptation principles

Research in the area of speaker adaptation has been carried out for several years, attempting to increase the performance of the resulting speaker dependent system. Before describing the different types of techniques in section 2.2, some commonly used basic terms in the area of speaker adaptation are presented in the following section.

2.1 Speaker adaptation terminology

The principle of a speaker adaptation system applied to a hidden Markov model speech recogniser is sketched in figure 2.1.

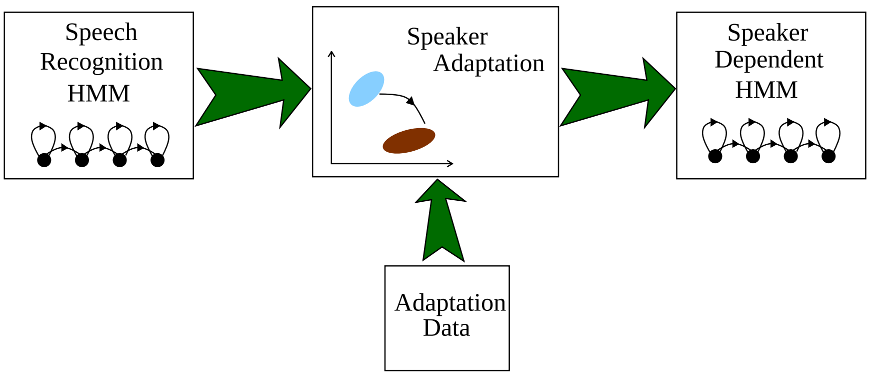

Figure 2.1: *A schematic representation of the underlying principle of speaker adaptation. Here it is shown for a hidden Markov model (HMM) based recognition system. The speaker adaptation extracts the speaker-specific information from the given adaptation data. Which adaptation method is chosen depends on a trade off between the requirements to processing speed and the nature of and amount of available adaptation data. The resulting system consists of speaker dependent hidden Markov models.*

Supervised/Un-supervised

Numerous factors influence the choice of adaptation method. The conditions under which the user uses the speech recognition system plays a major role because this determines the type of adaptation material. In some applications the interaction with the speaker is of a kind that makes it possible to text-prompt the speaker for adaptation data. This provides the advantage of enabling the ability to choose the sound segments contained in the adaptation data. Systems, where transcriptions of adaptation data are available, are generally termed **supervised**. In contrast there are applications, where it is inconvenient or even impossible to control the contents of the adaptation data. The term **un-supervised** is used in such cases.

Batch/Incremental/Instantaneous

The choice of adaptation method is also influenced by the amount of available adaptation data. The amount of data could be seconds or minutes dependent upon the type of application. If the interaction with the speaker is short and only temporary, it is inconvenient to require many minutes of adaptation data, whereas if the speaker is intended to use the system for a long period, it seems more acceptable to request the speaker to spend time generating the sufficient adaptation material in trade for a higher performance.

Below are listed three different modes of speaker adaptation, that arise from the intended usage of the application. They are describe in more details in [Zavaliagkos *et al*, 1995] and [Zavaliagkos, 1995]. In figure 2.2 the different modes are illustrated.

Batch The most generally used adaptation mode, where some amount of data is provided separately as adaptation material, and the system is adapted before the actual usage of the adapted systems starts.

Incremental Here the speech recognition system adapts continuously to the user. When a sentence is said by the user it is sought recognised. In parallel with this, an adaptation of the system is performed in order to improve the performance of the system, before the next utterance is said. This procedure is repeated for every utterance.

Instantaneous This adaptation mode is used, when the only available adaptation data is the utterance which is also sought recognised. Therefore instantaneous adaptation involves adapting and recognising at the same time.

In general, whether the chosen method uses supervised or un-supervised adaptation and whether a batch, an incremental or an instantaneous adaptation mode is the best, depends upon the application. Some techniques are good at giving fast performance improvements with only a limited amount of data, whereas other techniques need more material, but then guarantees a performance that moves asymptotically towards the one which could have been obtained with a speaker dependent system trained from scratch [Zavaliagkos, 1995]. Some of the most commonly used techniques are reviewed in the next section.

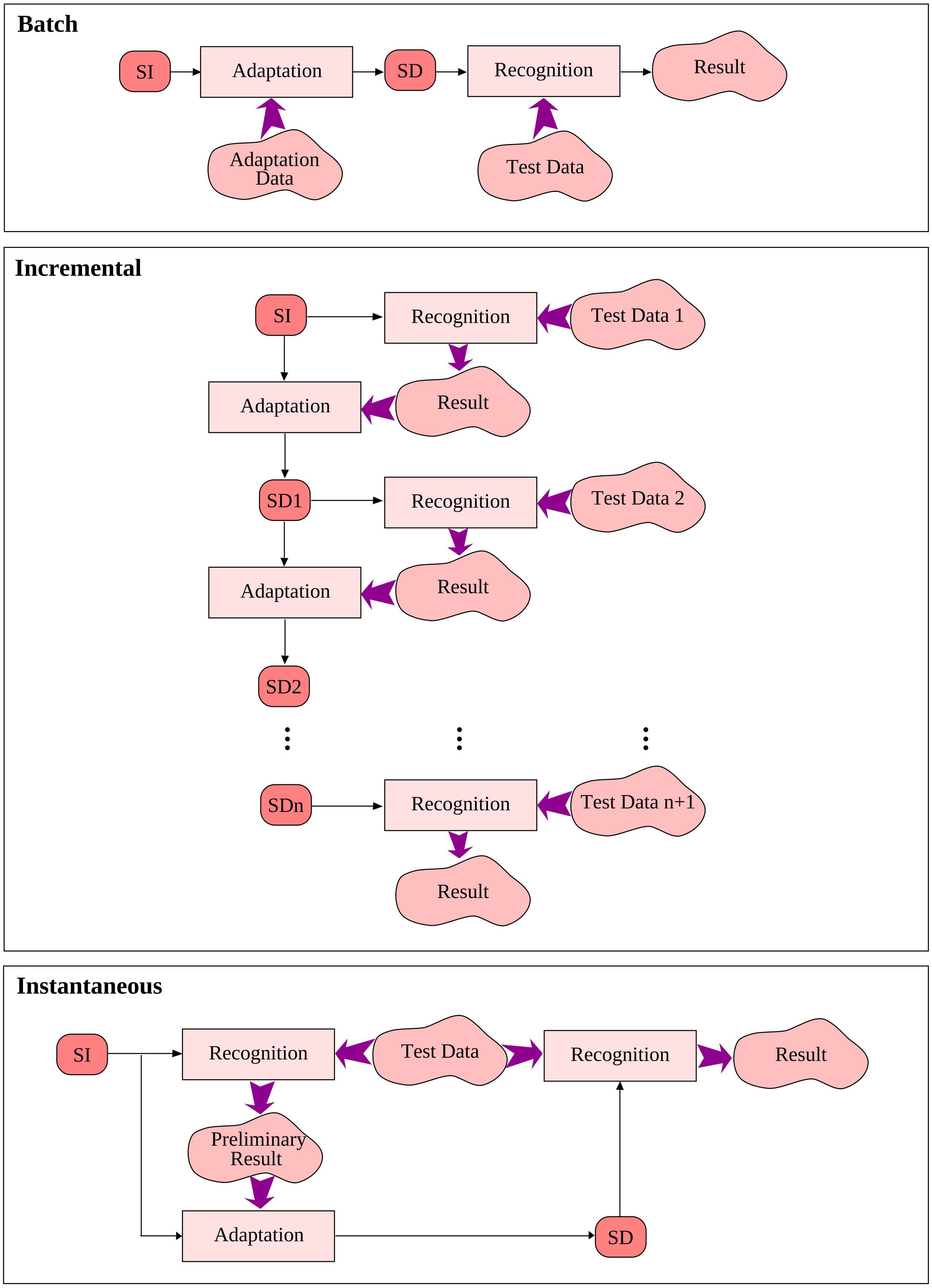

Figure 2.2: *Illustration of the three adaptation modes: batch, incremental and instantaneous. SI - Speaker Independent system, SD - Speaker Dependent system.*

2.2 Speaker adaptation approaches

When applying speaker adaptation techniques to speaker independent systems based on the hidden Markov modeling method, especially two types of approaches have been dealt with. One family of speaker adaptation techniques seeks to transform the spectral characteristics of the speech of the new speaker to those of the speech which the initial system is trained upon. Such transformations are applied in the feature space and are known as **spectral mapping techniques**. A summary of the various techniques for spectral mapping is presented in section 2.2.1. Another approach is to tune the parameters of the hidden Markov models of the initial system to the characteristics represented in the adaptation data. Section 2.2.2 describes the **model mapping approach**. Recently a third type of adaptation approach has been investigated which is basically a combination of the spectral mapping and the model mapping. Section 2.2.3 deals with an example of such a combined method.

Notation

In the next sections the following notation is used.

$\boldsymbol{x}_n$	feature vector for new speaker
$\boldsymbol{y}_n$	adapted feature vector for new speaker
$\boldsymbol{x}_r$	feature vector for reference speaker
$\boldsymbol{y}_r$	adapted feature vector for reference speaker

2.2.1 Spectral mapping approach

Fundamental to the spectral mapping approach is the idea, that the performance of an initial speech recognition system, can be increased by a mapping of the feature vectors of the new speaker. The transformation is carried out in such a way that the difference between the spectra of the reference speakers (possibly only one) and the spectrum of the new speaker is minimised. The exact way of defining this difference and minimising the corresponding distance has led to different techniques. However the basic assumption applies to all attempts; namely that the inter-speaker variations cause differences in the spectral contents of the speech signal, which are reflected in the feature representations used. An effort is therefore taken to create a feature space which is robust to these spectral variations. In some of the reported methods both the feature spaces of the reference speakers and of the new speaker are transformed into some common feature space. In others, a transformation that maps the spectral variations of the new speaker on to those given by the reference speakers, is applied to the feature vectors of the new speaker.

Many of the first attempts to use spectral mapping as a speaker adaptation technique were tailored towards speech recognisers based on Dynamic Time Warping (DTW).

In 1990 Class *et al* presented a comparison of different spectral mapping methods, testing the methods with both a DTW and an HMM speech recognition system [Class *et al*, 1990]. The simplest investigated strategy was a transformation of the following form:

$$\mathbf{y}_n = P(\mathbf{x}_n) = \mathbf{A}^T\mathbf{x}_n \tag{2.1}$$

where $P(\cdot)$ is the transformation which is applied to the feature vector for the new speaker, $\mathbf{x}_n$, $\mathbf{A}^T$ is the transformation matrix, and $\mathbf{y}_n$ is the resulting feature vector. The transformation matrix is estimated in such a way, that the mean-squared error between the adapted vector, $\mathbf{y}_n$ and the feature vector for the reference speaker, $\mathbf{x}_r$ is minimised.

$$D \quad = \quad \mathrm{E}\left[(\mathbf{x}_r - \mathbf{y}_n)^T \cdot (\mathbf{x}_r - \mathbf{y}_n)\right] = \mathrm{E}\left[(\mathbf{x}_r - \mathbf{A}^T\mathbf{x}_n)^T \cdot (\mathbf{x}_r - \mathbf{A}^T\mathbf{x}_n)\right] \tag{2.2}$$

$$\min_{\mathbf{A}}(D) \quad \Rightarrow \quad \mathbf{A} = \left(\mathrm{E}[\mathbf{x}_n\mathbf{x}_n^T]^{-1}\right) \cdot \mathrm{E}[\mathbf{x}_n\mathbf{x}_r^T] \tag{2.3}$$

The principle of mapping the feature vector space of the new speaker to that of a reference speaker is illustrated in figure 2.3.

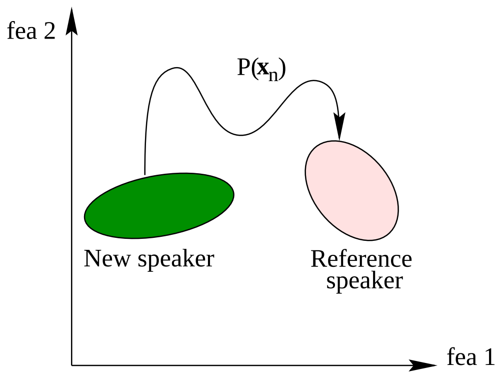

Figure 2.3: *Illustration of a spectral mapping principle, where the feature vector space of a new speaker is transformed into the feature vector space of a reference speaker.*

One of the other methods Class *et al* investigated was originally proposed by Choukri *et al* in 1986 [Choukri *et al*, 1986]. Choukri *et al* presented a transformation method where two different transformations are estimated in order to map the new speaker and the reference speaker template to a common feature space. The transformation, P_a maps the feature vector of the new speaker, $\mathbf{x}_n$ to another vector, $\mathbf{y}_n$, and similarly P_b maps a feature vector of the reference speaker, $\mathbf{x}_r$ to $\mathbf{y}_r$. This is illustrated in figure 2.4.

$$\mathbf{y}_n = P_a(\mathbf{x}_n) \tag{2.4}$$
$$\mathbf{y}_r = P_b(\mathbf{x}_r) \tag{2.5}$$

Choukri *et al* tested the method on an isolated word recognition task and achieved an increase in the word accuracy rate from 87 % without adaptation to 90.6 % with adaptation.

The comparison of the different methods in [Class *et al*, 1990] was done in a phoneme recognition task using both a DTW and a discrete HMM system. Both reference systems were implemented as speaker dependent systems, however not including the new speaker to which they were sought adapted. The methods illustrated in figures 2.3 and 2.4 both give reductions in error rates. Both for the DTW and the HMM system, the method taken from [Choukri *et al*, 1986] (figure 2.4) performs the best. Specifically the word error rate drops from 42.7 % to 11.5 % for the DTW and from 22 % to 11.5 % for the HMM.

However, common for all the above described variants of the spectral mapping approach is that none of them take into consideration, that the feature vectors arise from speech. In 1989 Cox and Bridle

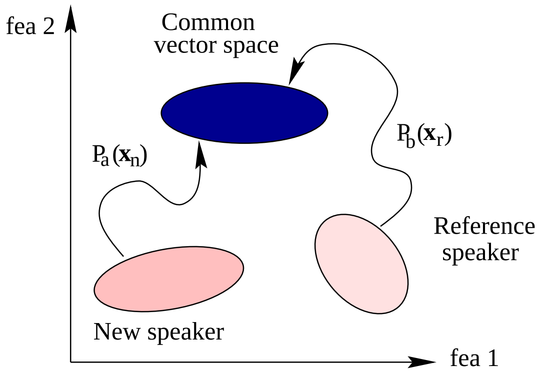

Figure 2.4: *Illustration of the spectral mapping method presented by Choukri et al. One transformation maps the feature space for the new speaker to a common vector space. Another transformation maps the feature space for the reference speaker to the same vector space. The two transformations are estimated in such a way that the correlation between* $\mathbf{y}_n$ *and* $\mathbf{y}_r$ *is maximised.*

in [Cox and Bridle, 1989] presented a method which uses the information incorporated in speaker independent Markov models and estimate a transformation of the means of the models. The method is known as the spectrum-bias adaptation. Even though the method transforms the means of the hidden Markov models, it is still characterised as a spectral mapping method. This is done because the object of the method still is to improve the match between the reference speakers and the new speaker (the spectral mapping idea), rather than to improve the modeling accuracy for the new speaker (the model mapping idea) [Leggetter, 1995].

In Cox and Bridle's method the mean of the adapted speaker dependent class is calculated as the mean of the distribution of the speaker independent class plus a bias-vector. The bias-vector is calculated as the averaged distance between the observation vectors and the means of the speaker independent classes.

$$\boldsymbol{\mu}_{new} = \boldsymbol{\mu}_{present} + p \cdot \boldsymbol{\delta} \quad , \qquad 0 \leq p \leq 1 \tag{2.6}$$

where $\boldsymbol{\mu}_{new}$ is the new mean vector for the adapted models, $\boldsymbol{\mu}_{present}$ is the mean vector of the speaker independent models, p is a weighting factor that determines how much the new mean should depend on the bias-vector and $\boldsymbol{\delta}$ is the distance calculated as:

$$\boldsymbol{\delta} = \frac{1}{X} \sum_{i=1}^{X} (\boldsymbol{x}_i - \boldsymbol{\mu}_{present}) \tag{2.7}$$

where X is the number of labeled examples from a given class and $\boldsymbol{x}_i$ is the observation vector for the new speaker. This is illustrated in figure 2.5

In 1994 Zhao also did an attempt to use the fact that the signal that is to be recognised is speech [Zhao, 1994]. Her theory was that the necessary transformation can be viewed as a combination of two transformations. A transformation arising from acoustic variations independent of the single phonemes, and another transformation which is meant to compensate for the more phoneme-specific variations. This can be formulated as:

$$\mathbf{y}_n = \mathbf{A}\mathbf{P}_i\mathbf{x}_n \tag{2.8}$$

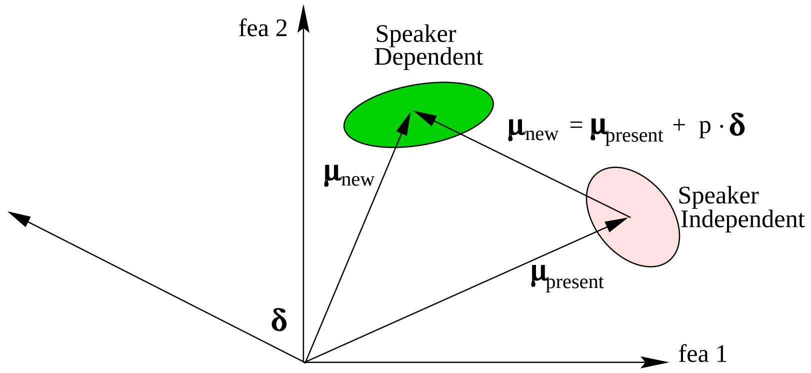

Figure 2.5: *Illustration of the spectrum-bias adaptation techniques presented by Cox and Bridle in 1989. A speaker independent system is used as example.*

where $\mathbf{x}_n$ is the observed feature vectors for the new speaker and $\mathbf{y}_n$ is the corresponding adapted feature vector, $\mathbf{P}_i$ is the phone-specific transformation vector for phone i, and $\mathbf{A}$ is the overall transformation matrix related to the change in acoustic source which takes place from one speaker to another. This matrix is estimated using the equations derived by Cox and Bridle as described above. Zhao achieves a reduction in word error rate of 27.5 %. She used Gaussian mixture density based hidden Markov models of phone units.

2.2.2 Model mapping approach

The methods for spectral mapping do either make global transformations of the feature vectors or requires adaptation data for every adaptation class in order to estimate the transformation. Also only a few of the techniques take into account the information about the generation of speech which is present in the hidden Markov models. The model mapping methods which are presented in this section incorporate this information.

When a hidden Markov model has been trained for a speaker independent system, it is trained in such a way that it is capable of simulating or generating feature vectors with the same stochastic characteristics as those observed in the training data. It is natural to utilise this information in a speaker adaptation method. The model mapping approach adapts hidden Markov models to speaker specific models.

Most of the methods re-estimate parameters of continuous density HMM's by maximising the probability of a model, $\boldsymbol{\lambda}$ given a set of observations, $\mathbf{O}$

$$P(\boldsymbol{\lambda}|\mathbf{O}) = \frac{P(\mathbf{O}|\boldsymbol{\lambda})P_p(\boldsymbol{\lambda})}{P(\mathbf{O})} \tag{2.9}$$

and assuming that $\boldsymbol{\lambda}$ is drawn from some prior distribution $P_p(\boldsymbol{\lambda})$. This is known as the **Maximum A Posteriori** approach (MAP).

Equation 2.9 can be solved in different ways. One of the traditional references is the work of Lee and Gauvain presented in [Gauvain and Lee, 1994], and recently Zavaliagkos has investigated MAP adaptation techniques in his Ph.D. thesis [Zavaliagkos, 1995].

The estimation method which is normally used when parameters for hidden Markov models are to be re-estimated is the Maximum Likelihood (ML) method. The special case of maximising equation 2.9, when no prior distribution is assumed is the ML estimate. It can be shown that the MAP estimates for the model parameters is in fact an interpolation between the ML estimate of the new data and the old parameter. Assuming a prior Gaussian distribution with mean $\boldsymbol{\mu}_{prior}$, for the case of a single Gaussian distribution the MAP adapted mean, $\hat{\boldsymbol{\mu}}_{MAP}$ can be expressed as:

$$\hat{\boldsymbol{\mu}}_{MAP} = \frac{\tau \boldsymbol{\mu}_{prior} + c\boldsymbol{\mu}_{ML}}{\tau + c} \tag{2.10}$$

where $\boldsymbol{\mu}_{ML}$ is the ML estimate of the mean using the new data, τ is a parameter describing the relative weight of the prior to the adaptation data, and c is a factor depending on the probability of observing the adaptation data given the model parameters. This way of computing the MAP estimate is often referred to as Bayesian learning. For a small amount of adaptation data the τ factor is large, and the term containing the prior estimate will be dominating. As the amount of adaptation data increases, the τ factor decreases and the influence of the ML estimate to the resulting MAP estimate increases. This illustrates a very important property of the MAP speaker adaptation technique: When the amount of adaptation data increases, the estimate approaches the ML estimate asymptotically, and as the ML estimate is the one used in the ordinary training algorithms, this means, that the results obtained using MAP estimates will approach the results obtained using a speaker dependent system fully trained from scratch.

For all the above described techniques, the main object has of course been to develop methods that improve the relation between the performance and the amount of adaptation data. Another keyword has been fast computations. However, none of the methods solve the problem of how models, for which no adaptation material is available, are adapted. As described in the introduction it seems likely that any segment of adaptation data implicitly contains some general information about the new speaker which can be used to adapt all model.

In 1993 Cox presented a speaker adaptation technique based on linear regression [Cox, 1993]. It was later further extended by Ahadi and Woodland in [Ahadi and Woodland, 1995]. The idea is to first use the available adaptation material to make initial MAP estimates for the model means as described above. These MAP estimates are then used to predict the means of the models which were not present in the adaptation data. This is done via a set of regression coefficients computed using previously trained speaker dependent models. The method is known as the Regression-based Model Prediction method (RMP), and the resulting predicted mean, $\hat{\boldsymbol{\mu}}_{RMP}$ is found using the principle of Bayesian training as described above.

$$\hat{\boldsymbol{\mu}}_{RMP} = \frac{\boldsymbol{\mu}_{reg}\sigma^2_{MAP} + \boldsymbol{\mu}_{MAP}\sigma^2_{reg}}{\sigma^2_{MAP} + \sigma^2_{reg}} \tag{2.11}$$

where $\boldsymbol{\mu}_{reg}$ and σ^2_{reg} are the regression predicted mean and its variance respectively, and $\boldsymbol{\mu}_{MAP}$ and σ^2_{MAP} the MAP estimated mean and its variance respectively.

Ahadi and Woodland compared their RMP approach to the MAP method, where only models with corresponding adaptation material are updated. The number of adaptation sentences was varied from 1 to 600. For a few number of sentences the RMP method showed clearly superior to the MAP method. As the number of sentences was increased, the two methods approached each other as is expected from equation 2.11. Also both methods approached the result obtained for a speaker dependent system trained from scratch.

2.2.3 Combined mapping approach

Recently a speaker adaptation technique which combines the spectral mapping approach with the model mapping approach has been proposed in [Leggetter, 1995]. The idea is to combine the good properties achieved with the MAP estimates with the spectral mapping approach, in order to make it possible to adapt all models in a system, regardless of which models occur in the adaptation data. Again the main object is to optimise the speaker adaptation process in order to reduce the amount of adaptation data.

Leggetter's technique is called Maximum Likelihood Linear Regression (MLLR). Like the MAP approaches it adapts hidden Markov model parameters. The method transforms the means of continuous hidden Markov models. It is done by multiplying an extended mean vector, $\boldsymbol{\xi}_s$ with a transformation matrix, $\mathbf{W}_s$

$$\hat{\boldsymbol{\mu}}_s = \mathbf{W}_s \cdot \boldsymbol{\xi}_s \tag{2.12}$$

where $\mathbf{W}_s$ is the transformation matrix for a given mixture component s, $\hat{\boldsymbol{\mu}}_s$ is the adapted mean vector of the model and $\boldsymbol{\xi}_s$ is the extended mean vector,

$$\boldsymbol{\xi}_s = [\omega, \mu_{s_1}, \ldots, \mu_{s_n}]' = [\omega : \boldsymbol{\mu}_s]' \tag{2.13}$$

where $\boldsymbol{\mu}_s$ is the original mean vector and ω is an offset term.

The transformation matrix is determined with a re-estimation algorithm based upon the principle of maximum likelihood estimation (ML). In this way, the re-estimated transformation matrix is the one that maximises the probability of having generated the observed adaptation data using the model.

In the most general case a transformation matrix is calculated for every mixture component. However, this requires examples of all classes that are to be adapted as is the case with the conventional MAP approaches. To circumvent this, Leggetter incorporates linear regression in the theory by introducing the concept of regression classes. A regression class is set of mixture components for which it is assumed that the same transformation matrix may be used for all mixture components in the regression class. The advantage is that, a transformation matrix for a given regression class is estimated using adaptation data corresponding to all present mixture components belonging to the regression class. Therefore models which are not present in the adaptation data can still be transformed using the regression class transformation matrix. The success of this approach is of course highly dependent upon the choice of division into regression classes.

Leggetter investigated the effect of varying different parameters in the MLLR technique: the number of regression classes, the amount of adaptation data, the adaptation mode and whether the adaptation was supervised or un-supervised. The amount of adaptation data is varied between 1 and 40 utterances and the recognition results are compared to the results obtained for the initial speaker independent system and a speaker dependent system trained from scratch. Even when the adaptation is carried out using only 3 utterances the word error rate drops below the one observed for the speaker independent system. From 4.3 % average word error rate to 3.8 % using 5 utterances. The word error rate for the speaker dependent system is 1.8 %. When using more than 15 utterances the error rate seems to stabilize around 3.2 %.

These results were all obtained using a global regression class. In another series of tests the number of regression classes was varied from 1 to 40 while the number of utterances kept fixed at 40. As the number of regression classes increased, the word error rate decreased reaching a minimum at around 15 regression classes, 2.7 %. As the number of regression classes was further increased, the word error increased again indicating that the transformation matrixes are difficult to estimate well enough when the amount of adaptation data is reduced too much due to a fine regression class division.

Chapter 3

Definition of project

In this preliminary part a brief introduction to speaker adaptation has been given. It was identified that the reason to perform speaker adaptation is to increase the performance obtained when a speech recognition system is operating on a specific speaker, not included in the training material - a situation which is almost unavoidable. Consequently the term speaker adaptation typically covers the process of tuning a speech recognition system to a new speaker, thereby creating a new speaker dependent system. The adaptation is directed by an amount of adaptation material provided by the new speaker. Numerous techniques and different approaches to speaker adaptation have been studied over the years.

The previous chapter contained a review of some of the major techniques. They all aim at getting the best performance for the resulting speaker dependent system, and due to factors concerning the implementation, naturally also aim at using as little adaptation material as possible in the process. This implies a demand for high data effectiveness of the speaker adaptation method. Some relevant questions are how much the amount of adaptation material can be reduced and still give an acceptable increase in performance, and whether the phonetic contents of the adaptation material have any influence upon the increase in performance achieved with the adaptation? Evidently as the amount of adaptation material decreases considerations regarding information content become more and more important.

3.1 The aim of the project

The current thesis deals with the problem of speaker adaptation of hidden Markov models as sketched above. The project focuses mainly upon two strongly related and very important issues and the aim can be formulated as being

- To investigate the effect on the performance of an adapted speaker dependent system, when the **amount of training material** is varied. Is there a tendency in the variations of the performance as the number of adaptation utterances increases, and is it possible to determine some lower limit of how much material is required to achieve an acceptable increase in performance?
- To investigate the effect on the performance when the **phonetic contents of the adaptation material** is varied. Is it possible to determine some phoneme classes which seem to contain the most speaker specific information? And to what extend are the results speaker dependent?

In addition the following questions will be investigated:

- How does the improvement of performance obtained by performing speaker adaptation vary with the speaker?
- What is the effect of varying the quality of the initial models?

3.2 Strategies for implemention of speaker adaptation system

This report documents the theory and implementation of a framework under which the aim of the project as specified above is tested. As the main focus of the thesis is on the speaker adaptation problems, the complete system has been realised as a combination of available speech recognition tools and an own implementation of a speaker adaptation technique. In figure 3.1 a schematic overview of the complete speaker adaptation system has been given. It shows the interactions between parts which have been implemented in the project and general speech recognition tools, not implemented in the project.

For reasons of clarity the system has been divided into an implementation and a test part. In the implementation part a number of speech recognition systems are made either by complete training from scratch or by adaptation. In the test part the performances of all speech recognition systems have been measured, and the results from these tests constitute the basis for the final conclusions.

In the list below the systems are described; the numeration follows the one in figure 3.1.

1. A **speaker independent system** trained on material from 50 reference speakers. These models constitute the initial models for the adaptation.
2. A set of 10 **speaker dependent systems trained from scratch**, one for each adaptation speaker. These systems constitute a set of well trained models as opposed to the models obtained by adaptation.
3. A set of 10 **adapted speaker dependent systems**, one for each adaptation speaker.
4. The performance of the speaker independent system is measured to provide a baseline performance to which the adaptated systems are compared.
5. The speaker dependent systems trained from scratch are tested in order to get results to which the adapted speaker dependent systems can be compared.
6. The adapted speaker dependent systems are tested.

The complete speaker adaptation system constitutes a framework for investigating the effects of the items listed in section 3.1. This is done mainly by varying the nature of the adaptation material, or by trying out different HMM topologies and observing the corresponding change in performances. Each configuration of the system is tried out for a total of 10 different adaptation speakers. The adapted speaker dependent systems are evaluated by comparing the recognition results obtained in part b) of the speaker adaptation system.

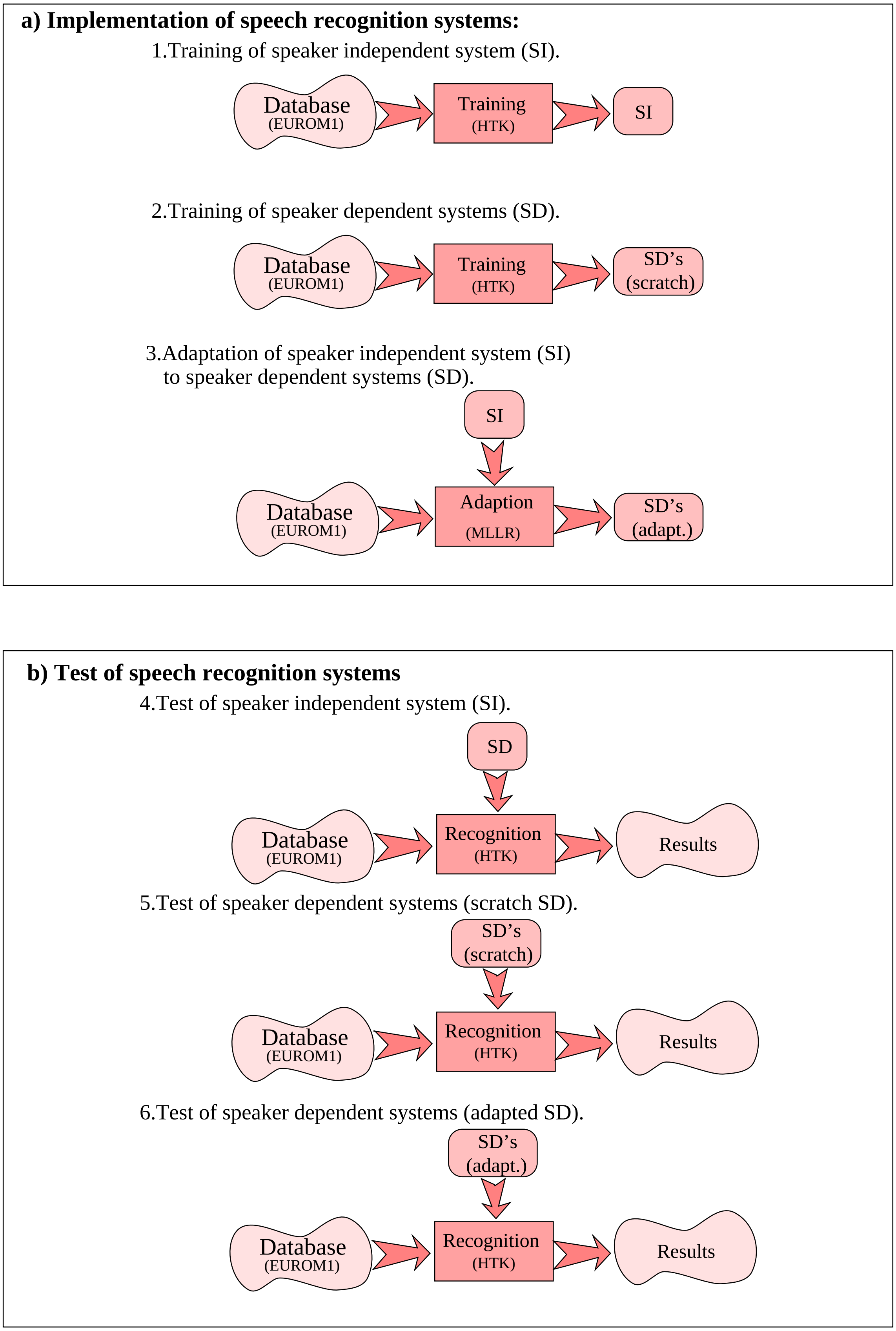

Figure 3.1: *A schematic overview of the implemented speaker adaptation system. It has been illustrated as consisting of two parts: a) the implementation of the recognition systems and b) the test of the various recognition systems. See text for further descriptions of the different blocks.*

The following sections contain the more detailed specifications of the different parts introduced in figure 3.1 including a brief introduction to the used tools and database. Further descriptions are given later in the report.

The adaptation technique

The chosen speaker adaptation technique is the Maximum Likelihood Linear Regression (MLLR) technique which has recently been introduced in[Leggetter, 1995]. As described in section 2.2.3, it is a combination of the spectral mapping approach and the model mapping approach. This provides it with the ability to update the means of models which are not present in the data; a property which is very useful in this project, when investigations concerning the amount of data is carried out. Also the MLLR method has proven to give very promising results [Leggetter and Woodland, 1994], [Leggetter, 1995], [Leggetter and Woodland, 1995a] and [Leggetter and Woodland, 1995b]. The adaptation is chosen to be done in a supervised manner following the batch principle. This is done in order to provide the implemented systems with properties often seen in applications. With text-prompted systems, which are supervised in nature, the phoneme sequence of the adaptation data can be controlled to a high degree. In such systems it is relevant to have a knowledge of which phonemes that constitute the best adaptation material.

The method is implemented using the GNU C compiler.

Phoneme modelling of speech recognition systems

Both the speaker independent system and the speaker dependent systems are based on the continuous density hidden Markov modelling (HMM) technique using mono-phone models. All systems are implemented using the HTK Toolkit version 1.5 [Young *et al*, 1993]. It is a generally used toolkit that provides tools for both training and testing of HMM's. A further description of HTK1.5 is given in appendix D. The HTK1.5 tool is also used for parameterising the speech data into Mel Frequency Cepstral Coefficients (MFCC's). This feature representation is chosen because it is widely used and known to give a good performance [Mermelstein and Davis, 1980], [Mariani, 1989]. As the thesis mainly deals with the speaker adaptation problem less focus has been put on optimising the parameters defining the HMM's. The performance of the initial speaker independent system is regarded as a baseline to which the results from the adapted models can be compared. However, an attempt has been made to choose typical and generally used parameter values for the feature processing.

Database

As seen in figure 3.1 the data used for training, adapting and testing are taken from the Danish EUROM.1 database [Lindberg and Christensen, 1995]. The database consists of three separate parts of which two are used in the current work:

- a Many Talker Corpus.
- a Few Talker Corpus.

A total of 60 speakers have been involved in the recordings. They have been chosen so as to give a reasonable equal distribution of age and gender. All 60 speakers are included in the Many Talker Corpus and of these, 10 speakers have also been included in the Few Talker Corpus. The division of speakers and speech material into an *adaptation group* and a *reference group* is sketched in figure 3.2. A more detailed description of the usage of the data is given in chapter 6, where the implementation

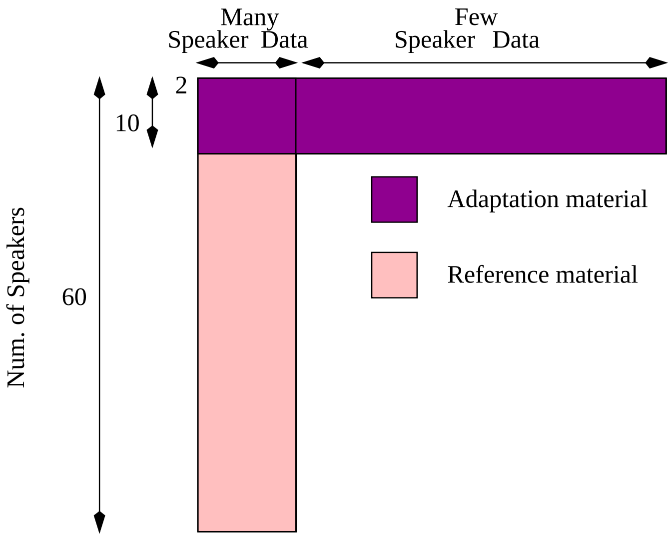

Figure 3.2: *Illustration of the distribution of speakers in the Many Speaker Corpus and the Few Speaker Corpus of the EUROM.1 database. It is marked which data have been selected as adaptation material and reference material respectively in the current project. Data from the 10 speakers in the Few Speaker Corpus constitute the material used for adaptation. The reference material is the data from the Many Speaker Corpus provided by the 50 speakers not also included in the Few Speaker Corpus.*

of the MLLR system shown in figure 3.1 is described, and in appendix C the database is described more thoroughly.

3.3 Organisation of the thesis

The preliminary part of the thesis has given an introduction to the problems of speaker adaptation ending with a definition of the project. The next part contains a description of the theories used in the project. An introduction to the theories behind hidden Markov models and Mel Frequency Cepstral Coefficients is given in chapter 4. The Maximum Likelihood Linear Regression theory and the implemented mathematical algorithms are described in chapter 5, with the full derivation of equations placed in appendix A.

The next part of the thesis deals with the implementation, tests and conclusions. Chapter 6 describes the implementation of the complete MLLR system. This include details about the specific choices concerning model topology and feature processing, the software specifications, and a presentation of the features and structure of the implemented program along with computational considerations done. Again additional details such as a description of the EUROM.1 database, the HTK1.5 tool, and part of the software documentation are given in appendixes.

A thorough description of test procedures and a discussion of the results obtained is given in chapter 7. Chapter 8 summarises the work presented in the current thesis and concludes on the research done.

Part II

Theory

Chapter 4

Phoneme modelling

The basic challenge for any speech recognition system is the variability of the speech presented to the system when operating. Speech utterances differ in both duration and spectral contents even though, they are examples of the same speech segment and therefore correctly should be associated with the same label by the recogniser. Not only the obvious differences between speakers, but also the inevitable changes in speech utterances from a single speaker, influence. The differences can be due to the conditions under which the speech is produced and they originate from both the speaking environment and the physical and psychological condition of the speaker. Therefore techniques which try to incorporate these variabilities in the modelling of the speech are highly suitable for speech recognition.

In this chapter the theories used for doing phoneme modelling in the current project are described. Section 4.1 introduces the general theory behind hidden Markov modelling of speech segments, and the theory of Mel Frequency Cepstral Coefficients are reviewed in section 4.2. Section 4.3 specifies the used feature representation and HMM topology.

4.1 Hidden Markov models

The hidden Markov model technique is a mathematical well founded and computational attractive way of modelling and recognising speech. The basic principle is to make a model of the production of a specific speech segment, which could be a word or a phoneme. The model is implemented as a generator capable of emitting feature vectors with the same stochastic characteristics as feature vectors extracted from different utterances of that particular segment. Each model is defined by a set of parameters estimated in a training session. During training, utterances of the speech segments are presented to the corresponding model, which is then updated using an estimation algorithm. The number of speech examples used to train the model must be enough to cover the stochastic variability of the speech segment in question.

When an unknown speech segment is presented to the fully trained speech recogniser, one of the models is chosen as the one which fit the segment best. The algorithm that determines which model and corresponding label to assign to the segment is based on a calculation of which model that has the highest probability of generating the feature vectors observed for the unknown word. That is to calculate the probability

$$P(\boldsymbol{\lambda}_i|\mathbf{O}) \tag{4.1}$$

for all $i \in \{1, \ldots, V\}$ where V is the number of models, $\boldsymbol{O} = \{\mathbf{o}_1\mathbf{o}_2\cdots\mathbf{o}_T\}$ is the sequence of observed feature vectors and $\boldsymbol{\lambda}$ defines the parameter set of the model.

Using Bayes' formula, equation 4.1 can be written as

$$P(\boldsymbol{\lambda}_i|\mathbf{O}) = \frac{P(\mathbf{O}|\boldsymbol{\lambda}_i) \cdot P(\boldsymbol{\lambda}_i)}{P(\mathbf{O})} \tag{4.2}$$

but if $P(\boldsymbol{\lambda}_i)$, the probability for a given model can be considered to be constant over all models. Also if all sets of feature vectors are equally likely, that will cause $P(\mathbf{O})$ to be invariable across models. This is often the case, and the result is that maximising equation 4.2 reduces to maximising the single term $P(\mathbf{O}|\boldsymbol{\lambda}_i)$, the probability of observing the set of feature vectors given a model. The recognition process is illustrated in figure 4.1.

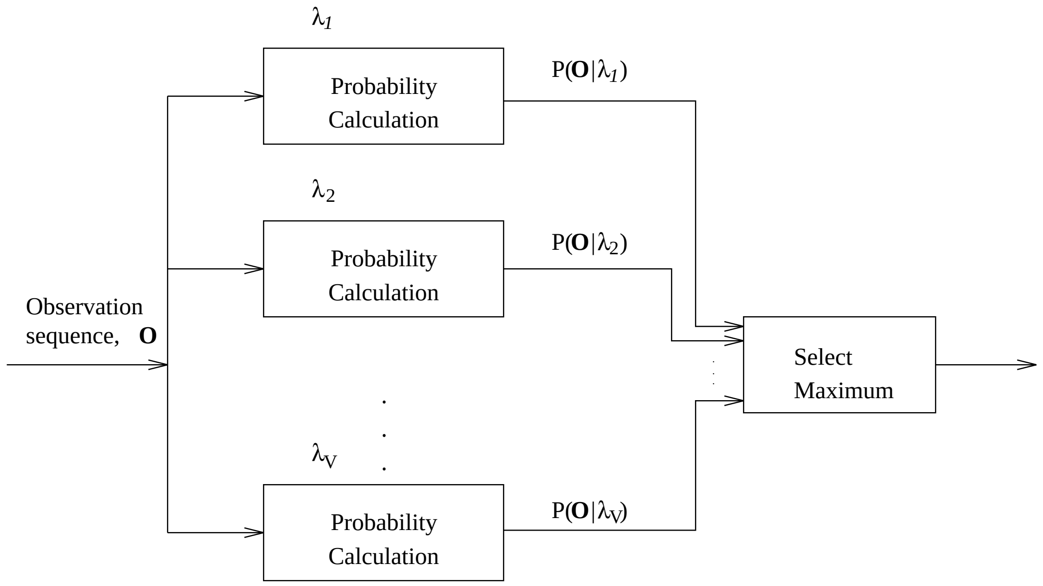

Figure 4.1: *Block diagram for a recognition process using the hidden Markov model technique. V is the number of different models in the recogniser, the size of the vocabulary.*

4.1.1 General theory

A Markov model defines a given number of states that a system can be in. At any time, t the system is in one of the states and with discrete time intervals, the system changes state either to a new state or back to the same state. The choice of the new state is random according to a set of transition probability distributions which are associated with each state. For a first order Markov model these transitions depend only upon the current state. In combination with entering a new state, the system outputs a symbol, which in the speech recognition case is a feature vector. In basic Markov models the emitted symbol is explicitly given with each state in the model. By observing the sequence of emitted symbols, the state sequence is therefore given unambiguously. If however the decision of which symbol to emit at a given state is itself a stochastic process, then the system can be modelled with a hidden Markov model. Here 'hidden' refers to the fact that the observation sequence can not be directly associated with a state sequence. Instead the probability that a given state sequence has emitted some observed set of feature vectors can be calculated. This probability is used in the determination of $P(O|\boldsymbol{\lambda}_i)$ in equation 4.1

4.1.2 Parameters defining a hidden Markov model

The parameters needed to specify a hidden Markov model are listed below. In figure 4.2 an example of a three state hidden Markov models is shown with indications of the different parameters.

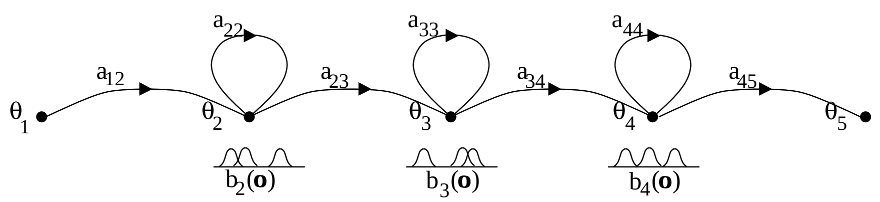

Figure 4.2: *A three state left-right model with three emitting states. State 1 and 5 are the non-emitting entry and end states respectively.*

To define a hidden Markov model with N states $\{\theta_1, \theta_2, \ldots, \theta_N\}$, where state θ_1 and state θ_N are non-emitting entry and exit states respectively, the following need to be specified:

- Transition probabilities for all the N states of the model. For convenience these probabilities are often viewed as a transition matrix.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix} \tag{4.3}$$

  where $a_{ij} = P[\theta_{t+1} = j|\theta_t = i]$ for $1 \leq i \leq N$; $1 \leq j \leq N$. If, in a model, the number of transitions from a state are required to be restricted, the corresponding a_{ij}'s are set to 0. Because the elements of $\mathbf{A}$ define the probability of being in state j at time $t+1$ given that the system was in state i at time t, the rows of $\mathbf{A}$ have to sum up to one, and none of the elements can be negative:

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad \text{for all} \quad i \tag{4.4}$$

$$a_{ij} \geq 0 \qquad \text{for all} \quad j, i \tag{4.5}$$

- The symbol probability distribution for each state, θ_j for $1 \leq j \leq N$: $\mathbf{B} = b_j(\mathbf{o}) = P[\mathbf{o}|\theta_t = j]$. In the case of a discrete parameter representation for the observations the $P[\cdot]$ is a discrete probability function and in the continuous case it represents a continuous probability density function

$$b_j(\mathbf{o}_t) = \sum_{k=1}^{M} c_{jk}\ \mathcal{N}(\mathbf{o}_t, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}), \quad 2 \leq j \leq N-1 \tag{4.6}$$

where M is the number of mixtures, $\mathbf{o}_t$ is the feature vector for frame t, c_{jk} is the mixture coefficient for the kth mixture component in state j and $\mathcal{N}$ is Gaussian with mean vector μ_{jk} and covariance matrix $\mathbf{\Sigma}_{jk}$ for the kth mixture component in state j. To ensure that the the mixture of the Gaussian distributions is itself a probability density distribution, the following constraints apply to the mixture coefficients:

$$\sum_{k=1}^{M} c_{jk} = 1, \quad 2 \leq j \leq N-1 \tag{4.7}$$

$$c_{jk} \geq 0, \quad 2 \leq j \leq N-1, \quad 1 \leq k \leq M \tag{4.8}$$

A hidden Markov model, $\boldsymbol{\lambda}$ is completely defined given the above described matrices:

$$\boldsymbol{\lambda} = (\mathbf{A}, \mathbf{B}) \tag{4.9}$$

4.2 Mel Frequency Cepstral Coefficients

All data have been pre-processed into Mel Frequency Cepstral Coefficients (MFCC's). The human perception of frequency has been incorporated into the computation of the MFCC's. This is done by using the *mel* scale which is based on experiments investigating the relation between the frequency of a pure tone and the frequency perceived by a human. *mel* is the unit of the perceived frequency [Moore, 1989]. A non-linear correspondence between *mel* and Hz is observed, and it can be modelled by introducing a transformation from the linear frequency scale onto the physiological motivated *mel* scale. The transformation equations can be expressed as:

$$f_{mel}(f_{Hz}) = 2595 \cdot \log_{10}\left(1 + \frac{f_{Hz}}{700}\right) \tag{4.10}$$

$$f_{Hz}(f_{mel}) = 700 \cdot \left(10^{\left(\frac{f_{mel}}{2595}\right)} - 1\right) \tag{4.11}$$

This relation is illustrated in figure 4.3

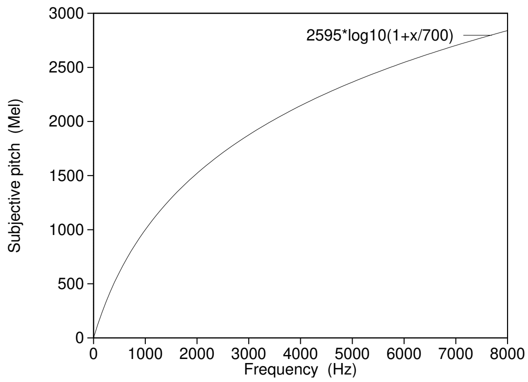

Figure 4.3: *The relation between the mel frequency scale and the linear frequency scale.*

Any spectral analysis can now be performed using these frequency transformation equations. One commonly used analysis is to transform the signal into the cepstrum domain. The cepstrum of a signal, $C(n)$, is calculated as the magnitude spectrum obtained as the discrete Fourier transformation of the logarithm of the Fourier transformed discrete signal.

$$C(n) = |\,\mathcal{F}\left(\log_{10}\left(\,|\,\mathcal{F}(x[n])\,|\,\right)\right)| \tag{4.12}$$

$$C_{mel}(n) = |\,\mathcal{F}\left(\log_{10}\left(|\,\mathcal{F}_{mel}(x[n])\,|\,\right)\right)| \tag{4.13}$$

The *mel* cepstrum, $C_{mel}(n)$ in equation 4.13 is obtained by substituting the initial Fourier transformation with the *mel* scaled Fourier transformation:

$$X(f_{mel}) = \sum_{n=0}^{L-1} x[n] \cdot e^{-j \cdot n \cdot f_{Hz}(f_{mel}) \cdot \frac{2\pi}{L}} \tag{4.14}$$

where L is the number of points in the transformation. Alternatively the power spectrum may be used.

The transformation from the frequency to the cepstrum domain is a non-linear homomorph transformation. The main property of the transformation is that it converts convolution in the frequency domain into addition in the cepstral domain. Investigating the spectral contents by Fourier transforming a signal in the cepstrum domain will therefore give Fourier coefficients, called cepstral coefficients, that describes the signals which were originally convoluted. Applied to speech analysis a cepstral transformation will in theory separate the pitch from the spectral envelope originating from the transfer function of the vocal tract. These properties make the cepstral coefficients useful in speech analysis and they have been widely used as parameter representations in speech recognisers. By introducing the *mel* scale as in equation 4.13 the frequency selectivity of the inner ear is furthermore incorporated into the analysis.

A method for computing *mel* scale Frequency Cepstral Coefficients (MFCC's) were presented in 1980 [Mermelstein and Davis, 1980]. The computational procedure is to take the magnitude of the Fourier transformed signal and weigh it with a set filters in a filter bank. The filter bank has triangular filters distributed linearly on the *mel* frequency scale. An example of such a filter bank is shown in figure 4.4.

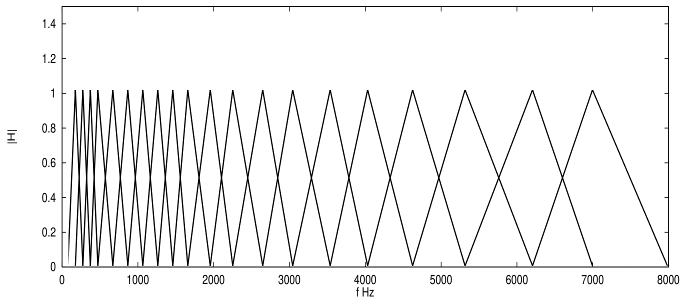

Figure 4.4: *The mel scaled filter bank shown with 20 filters as an example. The bandwidth of each filter is 300 mels and the spacing 150 mels.*

The logarithm is then applied to the weighted spectral magnitude. Finally this signal is cosine transformed to give the desired number of less correlated MFCC's:

$$MFCC(i) = \sum_{j=1}^{P} m_j \cos\left(\frac{\pi i}{P}(j - 0.5)\right) \quad , \quad 1 \le i \le K \tag{4.15}$$

where P is the analysis order, corresponding the number of filters in the triangular filter bank, K is the required number of output parameters and m_j is the logarithm of the output from the jth filter in the filter bank.

In HTK1.5 [Young *et al*, 1993] it is the method of Mermelstein and Davis which is implemented for computing the MFCC's. The number of filters in the filter bank, and the number of resulting cepstral coefficients can both be set by the user.

4.3 Specification of feature representation and topology

As mentioned above the speech recognition systems used in the current project are based on the hidden Markov modelling technique and the used data are pre-processed as Mel Frequency Cepstral Coefficients.

The topology for the continuous density HMM's is chosen to be a left-right model as the one in figure 4.2. For all phoneme models three emitting states are used. The model for silence ('SIL') is modelled using six emitting states and the model representing word silence ('SP') has one emitting state. The number of mixtures are varied in the different tests carried out.

The data material is pre-processed to 12 MFCC's with 12 first order derivatives using the HTK1.5 tool `HCode`.

The pre-processing is performed on a per frame basis. The frames are extracted with a frame period of $10ms$ and with a window duration of $20ms$. Each frame is multiplied with a Hamming window function to reduce window effects.

Table 4.1 summarises all parameters that specifies the mono-phone modelling in the current project.

Model topology	A left-right straight through model
Feature representation	12 MFCC's with 12 first order derivatives. A total of 24 features in each feature vector
Number of filters in *mel* filter bank	24
Frame period	10 ms
Window type	Hamming window
Window duration	20 ms
Pre-emphasis coefficient	0.95

Table 4.1: *Parameters specifying mono-phone modelling in the current project.*

Chapter 5

Maximum Likelihood Linear Regression

Speaker adaptation is applied to a speech recognition system with the intention to get a speaker dependent system with a better performance than obtainable with the original system. When dealing with HMM's most adaptation techniques use the original models as a starting point and adds speaker specific information by transforming some of the parameters in the models. The transformation is directed by an amount of adaptation data. The idea is that fully trained initial models contain some general speech information that will be useful for the new system as well. Additionally it is assumed that even the smallest amount of adaptation data contains some speaker specific information. This information will to some extent be class invariant, though the amount of information present in different classes may differ.

The Maximum Likelihood Linear Regression technique (MLLR) follows the above assumptions for adapting the mean vectors of continuous density HMM's. It is presented in [Leggetter, 1995] and in a number of articles [Leggetter and Woodland, 1994], [Leggetter and Woodland, 1995a] and [Leggetter and Woodland, 1995b]. As described in section 2.2.3 it can be seen as a combination of both spectral mapping methods and model mapping methods. In this chapter the general theory behind the technique is presented (section 5.1), and the idea of regression classes is dealt with (section 5.2). The main re-estimation equations are described in section 5.3, and the further derivations can be found in appendix A. The final section summarises the mathematical algorithms which are implemented.

5.1 General theory behind MLLR

Chapter 4.1 described the different types of parameters that constitute a hidden Markov model. As mentioned the MLLR technique only adapts the mean vectors of the density functions. The remaining parameters are kept unchanged in the adapted models. Ideally all parameters should be changed as they are all speaker dependent to some extend. However in applications where only a limited amount of adaptation data is available, there are not enough data to adapt all parameters.

As the MLLR is developed for such applications, only the means are adapted. According to [Leggetter, 1995] the adaptation of the transition probabilities and the mixture component weights will have very little effect on the final performance, and very few adaptation techniques transform these parameters. This is not the case for the covariance matrixes. [Leggetter, 1995] argues that rather than transforming the covariance matrixes with the same transformation matrix as the one estimated for the mean vectors, the covariance matrixes are better kept untransformed. In this way the individual distributions are preserved. However if a transformation matrix can be estimated specifically for the covariance matrix, it is likely that an improvement in performance can be achieved by transforming the covariances as well. In figure 5.1 the effect of only transforming the mean vectors is illustrated in the acoustic feature space.

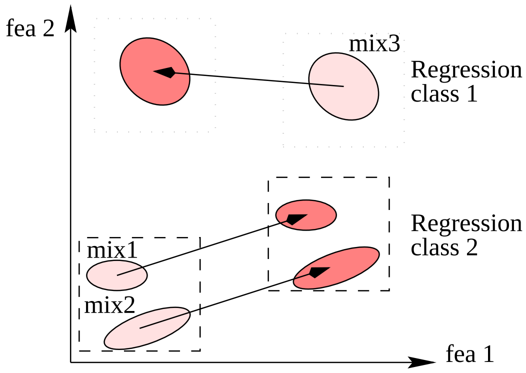

Figure 5.1: *Transforming only the mean vectors of the HMM's has the effect of moving the location of each mixture component in the acoustic feature space, but without changing the shape of it. The example illustrates a simple case with two features, three mixture components and two regression classes.*

The mean vector for the adapted mixture component is calculated by multiplying the mean vector for the initial model with a transformation matrix:

$$\hat{\boldsymbol{\mu}}_s = \boldsymbol{W}_s \cdot \boldsymbol{\xi}_s \tag{5.1}$$

where $\boldsymbol{W}_s$ is the transformation matrix for mixture component s, $\hat{\boldsymbol{\mu}}_s$ is the adapted mean vector and $\boldsymbol{\xi}_s$ is the extended mean vector for mixture component s. The extended mean vector is defined as

$$\boldsymbol{\xi}_s = [\omega, \mu_{s_1}, \ldots, \mu_{s_n}]' = [\omega \ : \ \boldsymbol{\mu}_s]' \tag{5.2}$$

where $\boldsymbol{\mu}_s$ is the original mean vector, ω is an offset term and n is the number of features.

The offset term, ω is constant in each regression class and the effect is illustrated in figure 5.2. The offset term can be used to model when the feature vectors differ from the mean vectors of the initial models by an additive term. This could typically be different recording environments etc.

5.2 Regression classes

The concept of regression classes was briefly mentioned in section 2.2.3. In this section the idea is investigated further including a discussion of different ways of determining how the division of mixture components into regression classes can be done.

The regression classes define sets of mixture components. It is important to notice that mixture components from the same state may, in the general case, be tied to two different regression classes. Doing this enables for an increased liberty in the way the distribution for a given state is adapted.

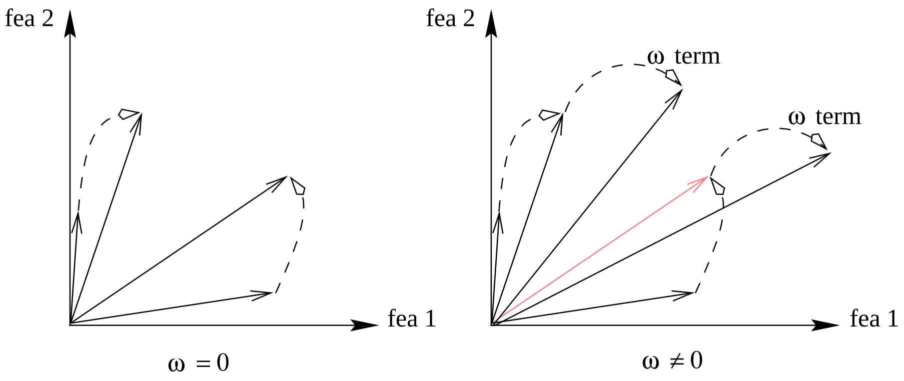

Figure 5.2: *Illustration of the MLLR transformation for a two-dimensional example. Each solid line represents a mean vector and each dashed line a transformation. The left-hand diagram shows a transformation where $\omega = 0$, and the right-hand diagram the situation where $\omega \neq 0$.*

Viewed in the acoustic feature space, tying the mixture components to different regression classes, will possibly make them move in different directions, whereas if they were tied to the same regression class, they would all move in the same direction.

The determination of the number of regression classes, and the way the mixture components are divided into them, are naturally very important. In theory the number of regression classes can vary from a single regression class covering all mixture components to a single regression class for every mixture component. However in practise the amount of available adaptation data sets an upper limit to the number of possible regression classes, as there need to be sufficient adaptation material for each regression class. In some applications the amount of adaptation material may be known in advance, in which case the division into regression classes can be fixed beforehand. In other situations the division is determined dynamically as part of the adaptation process, when the exact amount of adaptation data is known.

Leggetter suggests to do the dynamical division using a tree-based approach. The top node of such a tree represents a global regression class, and each leaf represents a single mixture component. Tying mixture components together is done by grouping branches in the tree. This clustering is stopped when a certain criterion is fulfilled. For instance when enough adaptation material exists for each of the resulting regression classes. One obvious advantage of defining the division during the adaptation is that the optimal division is likely to vary from one speaker to another; something which is impossible to take into consideration with a fixed regression class division.

When tying mixture components the main object is to tie components that are assumed to undertake a somewhat similar transformation, when they are adapted to the new speaker. Two different ways of determining which mixture components are to be tied together are studied in [Leggetter, 1995].

- **Phonetic characteristics:** Broad phonetic classes are defined and the phonetic type of the model, that a given mixture component belongs to, determines which regression class it is assigned to. Note that this causes all components for a given state to be transformed with the same transformation matrix. Table 5.1 shows the phonetic classes that Leggetter used.

- **Distance measures:** Two mixture components are grouped, if they are similar according to some distance measure. The choice of distance measure depends on the type of feature representation used.

Very front vowels
Near front vowels
Front diphthongs
Back diphthongs
Near back vowels
Very back vowels
Liquids
Nasals
Strong fricatives
Weak fricatives
Closures
Unvoiced stops
Voiced stops

Table 5.1: *A list of the broad phonetic classes used in [Leggetter, 1995] as regression class division.*

Leggetter argues for the fact that dividing into regression classes using distance measures is the most logical method. However his tests show no significant difference between the two approaches. In this thesis it is chosen to implement the method based on broad phonetic classes.

5.3 Re-estimation of the transformation matrix

In this section the re-estimation equations for the transformation matrix is presented. Leggetter derives the equations fully for the case of using a single-mixture Gaussian distribution, where a single observation is observed, and where every state constitutes its own regression class. In most cases however an extension to

- multiple regression classes
- multiple mixture components
- multiple observation sequences

is necessary. The basic formulae that apply to each of these cases are given in [Leggetter, 1995]. In appendix A these equations are combined and the extended solution to the re-estimation problem is derived. The procedure is summarised below, and the re-estimation equations for the most extended case are described.

As mentioned in section 2.2.3 the re-estimation equations for the transformation matrix are derived using the principle of maximum likelihood (ML). In the case of using regression classes, the aim is to determine the re-estimated transformation matrix, $\hat{\boldsymbol{W}}_s$ shared by all members of the regression class.

A regression class is a set of R mixture components $\{s_1k, s_2k, \ldots, s_Rk\}$; $1 \leq k \leq M$, where M is the number of mixtures per state. Following the ML principle $\hat{\boldsymbol{W}}_s$ is found as the $\boldsymbol{W}_s$ that maximises the probability of generating the observed P utterances with the adapted model parameters, $\hat{\boldsymbol{\lambda}}$

$$\hat{\boldsymbol{W}}_s = \max_{\boldsymbol{W}_s} P(\boldsymbol{O}_P|\hat{\boldsymbol{\lambda}}) \tag{5.3}$$

where $\boldsymbol{O}_P$ is the set of P utterances, $\boldsymbol{O}^{(1)} \ldots \boldsymbol{O}^{(P)}$, each consisting of an individual number of frames; the observation sequence $\boldsymbol{O}^{(p)}$ has T_p number of frames ($\boldsymbol{O}^{(p)} = \boldsymbol{o}_1^{(p)} \ldots \boldsymbol{o}_{T_p}^{(p)}$). An often used approach to solving ML problems is to define an auxiliary function.

$$Q(\boldsymbol{\lambda}, \hat{\boldsymbol{\lambda}}) = \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \sum_{K \in \Omega_b} P(\boldsymbol{O}_P, \boldsymbol{\theta}, K|\boldsymbol{\lambda}) \cdot \log(P(\boldsymbol{O}_P, \boldsymbol{\theta}, K|\hat{\boldsymbol{\lambda}})) \tag{5.4}$$

where K denotes a given sequence of mixture components, Ω_b is the set of all possible mixture component sequences, $\boldsymbol{\theta}$ is a given state sequence and $\boldsymbol{\Theta}$ is the set of all possible state sequences with length T, the number of observed feature vectors. An auxiliary function defined in this way can be shown to have the property that if a $\hat{\boldsymbol{\lambda}}$ is found that maximises equation 5.4, then the following relation applies [Huang *et al*, 1990]

$$Q(\boldsymbol{\lambda}, \hat{\boldsymbol{\lambda}}) \geq Q(\boldsymbol{\lambda}, \boldsymbol{\lambda}) \Rightarrow P(\boldsymbol{O}_P|\hat{\boldsymbol{\lambda}}) \geq P(\boldsymbol{O}_P|\boldsymbol{\lambda}) \tag{5.5}$$

that is, the probability in equation 5.3 does not decrease. Therefore an estimation of the transformation matrix can be found by maximising the auxiliary function with respect to $\hat{\boldsymbol{W}}_s$. The procedure is to express $P(\boldsymbol{O}_P, \boldsymbol{\theta}, K|\hat{\boldsymbol{\lambda}})$ as a function of the model parameters: the transition probabilities and the symbol density functions, where $\hat{\boldsymbol{W}}_s$ occurs. The auxiliary function is then maximised by differentiating with respect to $\hat{\boldsymbol{W}}_s$ and equating to zero. In appendix A the complete derivations are given. The result of the procedure is the following equation

$$\sum_{p=1}^{P} \sum_{t=1}^{T_p} \sum_{r=1}^{R} \gamma_{s_r k}^{(p)}(t) \boldsymbol{\Sigma}_{s_r k}^{-1} \boldsymbol{o}_t^{(p)} \boldsymbol{\xi}'_{s_r k} = \sum_{p=1}^{P} \sum_{t=1}^{T_p} \sum_{r=1}^{R} \gamma_{s_r k}^{(p)}(t) \boldsymbol{\Sigma}_{s_r k}^{-1} \hat{\boldsymbol{W}}_s \boldsymbol{\xi}_{s_r k} \boldsymbol{\xi}'_{s_r k} \tag{5.6}$$

where $\boldsymbol{\Sigma}_{s_r k}$ is the covariance matrix for the $s_r k$ th mixture component, and $\gamma_{s_r k}^{(p)}(t)$ is a helping variable that is introduced. It represents the mixture component occupancy probability defined as

$$\gamma_{s_r k}^{(p)}(t) = \frac{1}{P(\boldsymbol{O}^{(p)}|\boldsymbol{\lambda})} \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \sum_{K \in \Omega_b} P(\boldsymbol{O}^{(p)}, \theta_t = s_r, k_t = k|\boldsymbol{\lambda}) \tag{5.7}$$

So far no assumptions have been made about the nature of the covariance matrix, $\boldsymbol{\Sigma}_s$, but in the remaining part of the chapter, only the case of having a diagonal covariance matrix is dealt with. There is no closed form solution for the full covariance case [Leggetter, 1995].

In the following equation 5.6 is rewritten using a number of matrixes. These matrixes are part of the final mathematical algorithm listed in section 5.4. Introducing the matrixes $\boldsymbol{V}^{(r)}$ and $\boldsymbol{D}^{(r)}$ defined as

$$\boldsymbol{V}^{(r)} = \sum_{p=1}^{P} \sum_{t=1}^{T_p} \gamma_{s_r k}^{(p)}(t) \boldsymbol{\Sigma}_{s_r k}^{-1} \tag{5.8}$$

and

$$\boldsymbol{D}^{(r)} = \boldsymbol{\xi}_{s_r k} \boldsymbol{\xi}'_{s_r k} \tag{5.9}$$

equation 5.6 is brought on the form

$$\sum_{p=1}^{P} \sum_{t=1}^{T_p} \sum_{r=1}^{R} \gamma_{s_r k}^{(p)}(t) \boldsymbol{\Sigma}_{s_r k}^{-1} \boldsymbol{o}_t^{(p)} \boldsymbol{\xi}'_{s_r k} = \sum_{r=1}^{R} \boldsymbol{V}^{(r)} \hat{\boldsymbol{W}}_s \boldsymbol{D}^{(r)} \tag{5.10}$$

A matrix, $\boldsymbol{Y}$ is defined as the right hand side of equation 5.10, and the elements can be calculated as

$$y_{ij} = \sum_{p=1}^{n} \sum_{q=1}^{n+1} w_{pq} \left[\sum_{r=1}^{R} v_{ip}^{(r)} d_{qj}^{(r)} \right] \tag{5.11}$$

$$y_{ij} = \sum_{q=1}^{n+1} w_{iq} \sum_{r=1}^{R} v_{ii}^{(r)} d_{jq}^{(r)} \tag{5.12}$$

where n is the number of features. In the latter equation it has been utilised that $\boldsymbol{V}^{(r)}$ is diagonal and $\boldsymbol{D}^{(r)}$ is symmetric. The multiplication of the terms from the $\boldsymbol{V}^{(r)}$ and the $\mathbf{D}^{(r)}$ matrixes are then combined in the matrix, $\boldsymbol{G}^{(i)}$ where $1 \le i \le n$.

$$\boldsymbol{G}^{(i)} = \left[\; g_{jq}^{(i)} \; \right] = \begin{bmatrix} \sum_{r=1}^{R} v_{ii}^{(r)} d_{11}^{(r)} & \sum_{r=1}^{R} v_{ii}^{(r)} d_{12}^{(r)} & \cdots & \sum_{r=1}^{R} v_{ii}^{(r)} d_{1\,n+1}^{(r)} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{r=1}^{R} v_{ii}^{(r)} d_{n+1\,1}^{(r)} & \sum_{r=1}^{R} v_{ii}^{(r)} d_{n+1\,2}^{(r)} & \cdots & \sum_{r=1}^{R} v_{ii}^{(r)} d_{n+1\,n+1}^{(r)} \end{bmatrix} \tag{5.13}$$

Finally a $\boldsymbol{Z}$ matrix with the elements z_{ij} is defined as the left hand side of equation 5.10. This yields

$$\boldsymbol{Z} = \sum_{p=1}^{P} \sum_{t=1}^{T_p} \sum_{r=1}^{R} \gamma_{s_r k}^{(p)}(t) \boldsymbol{\Sigma}_{s_r k}^{-1} \boldsymbol{o}_t^{(p)} \boldsymbol{\xi}'_{s_r k} \tag{5.14}$$

Equating $\boldsymbol{Z}$ to the right hand side, $\boldsymbol{Y}$ and writing it per element gives

$$z_{ij} = y_{ij} = \sum_{q=1}^{n+1} w_{iq} g_{jq}^{(i)} \tag{5.15}$$

The re-estimate of the transformation matrix $\hat{\boldsymbol{W}}_s$ is then found on a per row basis as

$$\hat{\boldsymbol{w}}'_i = \boldsymbol{G}^{(i)-1} \; \boldsymbol{z}'_i \tag{5.16}$$

where $\hat{\boldsymbol{w}}_i$ and $\boldsymbol{z}_i$ are the i th rows in $\hat{\boldsymbol{W}}_s$ and $\boldsymbol{Z}$ respectively.

5.3.1 Diagonal transformation matrix

The method described above is computationally expensive. However the number of computations can be reduces considerably if the estimated transformation matrix, $\hat{\boldsymbol{W}}_s$ is restricted to a diagonal matrix

$$\hat{\boldsymbol{W}}_s = \begin{bmatrix} w_{1,1} & w_{1,2} & 0 & \cdots & 0 \\ w_{2,1} & 0 & w_{2,3} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ w_{n,1} & 0 & \cdots & 0 & w_{n,n+1} \end{bmatrix} \tag{5.17}$$

Rewriting the non-zero elements of $\hat{\boldsymbol{W}}_s$ to a vector, $\hat{\mathbf{w}}_s$ gives:

$$\hat{\mathbf{w}}_s = \begin{bmatrix} w_{1,1} \\ \vdots \\ w_{n,1} \\ w_{1,2} \\ \vdots \\ w_{n,n+1} \end{bmatrix} \tag{5.18}$$

By following the same procedure as for the case of a full transformation matrix an auxiliary function, $Q(\boldsymbol{\lambda}|\hat{\boldsymbol{\lambda}})$ is defined and differentiated with respect to $\hat{\mathbf{w}}_s$. Again the complete derivation procedure is given in appendix A. The derivation leads to the following equation similar to equation 5.6

$$\sum_{p=1}^{P}\sum_{t=1}^{T_p}\sum_{r=1}^{R}\gamma_{s_rk}^{(p)}(t)\boldsymbol{D}'_{s_rk}\boldsymbol{\Sigma}'_{s_rk}\boldsymbol{o}_t^{(p)} = \sum_{p=1}^{P}\sum_{t=1}^{T_p}\sum_{r=1}^{R}\gamma_{s_rk}^{(p)}(t)\boldsymbol{D}'_{s_rk}\boldsymbol{\Sigma}'_{s_rk}\boldsymbol{D}_{s_rk}\hat{\mathbf{w}}_s \tag{5.19}$$

where the matrix $\boldsymbol{D}_{s_rk}$ is a matrix defined as

$$\boldsymbol{D}_{s_rk} = \begin{bmatrix} \omega & 0 & \cdots & 0 & \mu_{s_rk1} & 0 & \cdots & 0 \\ 0 & \omega & \ddots & \vdots & 0 & \mu_{s_rk2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \omega & 0 & \cdots & 0 & \mu_{s_rkn} \end{bmatrix} \tag{5.20}$$

Two matrixes, $\boldsymbol{A}$ $(2n \times 2n)$ and $\boldsymbol{B}$ $(2n \times 1)$ are now defined as

$$\boldsymbol{A} = \sum_{p=1}^{P}\sum_{t=1}^{T_p}\sum_{r=1}^{R}\gamma_{s_rk}^{(p)}(t)\boldsymbol{D}'_{s_rk}\boldsymbol{\Sigma}'_{s_rk}\boldsymbol{D}_{s_rk}\hat{\mathbf{w}}_s \tag{5.21}$$

and

$$\boldsymbol{B} = \sum_{p=1}^{P}\sum_{t=1}^{T_p}\sum_{r=1}^{R}\gamma_{s_rk}^{(p)}(t)\boldsymbol{D}'_{s_rk}\boldsymbol{\Sigma}'_{s_rk}\boldsymbol{o}_t^{(p)} \tag{5.22}$$

The transformation vector $\hat{\boldsymbol{w}}_s$ is then found from equation 5.19 as

$$\boldsymbol{B} = \boldsymbol{A}\hat{\mathbf{w}}_s \tag{5.23}$$

$$\hat{\mathbf{w}}_s = \boldsymbol{A}^{-1}\boldsymbol{B} \tag{5.24}$$

In the current project both re-estimation equations for the case of having a diagonal and for the case of using a full transformation matrix have been implemented.

5.4 Implemented mathematical algorithms

The implemented mathematical algorithms are listed in this section in order to clarify the computational procedure of re-estimating the transformation matrix and adapting the mean vectors of the initial models. More specific information about the implementation is given in chapter 6.

Mathematical algorithm for a full transformation matrix

```
for all regression classes
    for all mixture components, s_r k
        for all observation sequences, p
            for all frames, t
                Calculate the state occupancy probability, γ^(p)_{s_r k}(t) [Eq. 5.7].
            end
        end
        Calculate the two matrixes V^(r) and D^(r) [Eq. 5.8 and 5.9].
    end
    for every feature, i
        Calculate G^(i) [Eq. 5.13].
        Invert G^(i).
    end
    Calculate Z [Eq. 5.14].
    Calculate W_s [Eq. 5.16].
    for all mixture components
        Adapt the means of the mixture component.
    end
end
```

Mathematical algorithm for a diagonal transformation matrix

```
for all regression classes
    for all mixture components, s_r k
        for all observation sequences, p
            for all frames, t
                Calculate the state occupancy probability, γ^(p)_{s_r k}(t) [Eq. 5.7].
            end
        end
    end
    Calculate A [Eq. 5.21].
    Invert A
    Calculate B [Eq. 5.22].
    Calculate ŵ_s [Eq. 5.24]
    for all mixture components
        Adapt the means of the mixture component.
    end
end
```

Part III

Implementation, tests and conclusion

Chapter 6
Implementation of MLLR system

To fulfill the overall purpose of the project as it is described in chapter 3, a MLLR system is implemented. An outline of it was given in figure 3.1 at page 23. This chapter will focus mainly on describing the implementation of the speech recognition system as illustrated in part *a* of the figure. In sections 6.1 and 6.2 the training of the speaker independent system and the speaker dependent systems are described respectively. Section 6.3 describes the process of adapting the speaker independent system. The software implementation of the MLLR technique is described in section 6.3.2 including some of the computational considerations. Chapter 7 concerns the tests of the implemented systems.

Data

The data used for the complete system are taken from the Danish EUROM.1 database [Lindberg and Christensen, 1995]. As described before, the data have been divided into a part with utterances from 50 reference speakers and a part including material from 10 adaptation speakers. Figure 6.1 illustrates the split of the data for these two speaker groups into a training and a testing part. The names defined in figure 6.1 are used throughout the report.

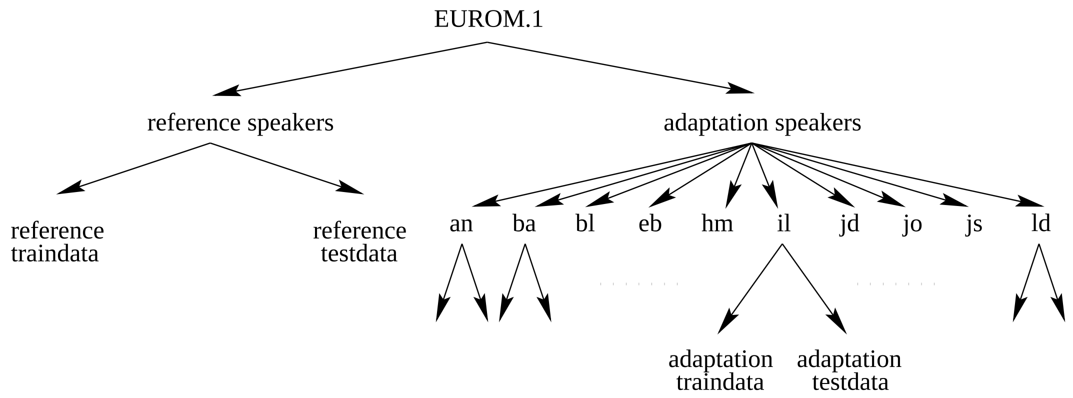

Figure 6.1: *Illustration of the split of the reference data and the adaptation data into training and testing parts.*

Appendix C described the database in more details. The number of occurrence of the different phonemes have been counted for the data parts defined in figure 6.1. The results are presented in tables C.4 to C.6.

6.1 Speaker independent system

The first part of the MLLR system is the speaker independent system. It is trained using material from all reference speakers, and the HMM's are used as the initial models that are adapted to the group of adaptation speakers using the MLLR technique. The performances of the speaker independent models are tested, and these results serve as a baseline for the performance obtained by testing the adapted HMM's. In figure 6.2 the part of figure 3.1 that covers the speaker independent system has been repeated.

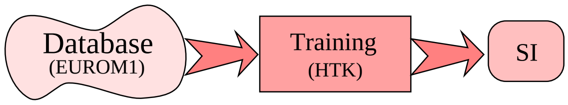

Figure 6.2: *Implementation of speaker independent system (SI).*

As the main focus of the project is on the speaker adaptation problem, the requirements to the implementation of the speaker independent system have mainly been in the direction of obtaining a system that is widely representative of typical speaker independent systems. The parameters defining the feature processing have therefore been fixed to typically used values and attempts have been done to train the HMM's in the best way with the available amount of data. In order to be able to test the effect of the quality of the initial models, nine different speaker independent systems have been trained. These systems only differ in that the number of mixtures in the models vary from two to ten.

As mentioned above all data used for training and testing the speaker independent system are taken from the data provided by the reference speakers in the EUROM.1 database. All data have been pre-processed to MFCC's using the parameter values already described in section 4.3. The pre-processing has been carried out using the HTK1.5 tool `HCode`.

When the data have been pre-processed, the HMM's are trained. Again the HTK1.5 tool kit is used. Some efforts have been put into training the system as well as possible given the available amount of data. The training process consists of a number of re-estimations. Between re-estimations the complexity of the HMM's is increased by adding more mixtures in all states. The first models have 2 mixtures. The number of mixtures is then constantly incremented by one until a total of 10 mixtures is reached. This is considered to be a reasonably maximum number of mixtures for mono-phone models taking into account the amount of available training material. This iterative training procedure is chosen because it is believed to give better results for high-mixture models. Appendix D gives a brief description of all used HTK1.5 tools and lists the complete training procedure.

Table 6.1 summarises the specification of the speaker independent system. A description of how the speaker independent system is tested is found in section 7.2.

Type of parameter	Chosen value
Feature type	12 MFCC with 12 derivatives
Model topology	straight through, left-right
Number of states	three[a]
Number of mixtures	2 -10
Training data	train data for reference speakers
Testing data	test data for reference speakers
Number of speakers	50

[a] There are used six states in the HMM for the 'SIL' class and one states in the HMM for the 'SP' class.

Table 6.1: *Summary of the parameters specifying the speaker independent system. Further details about the mono-phone modelling can be found in section 4.3, and more information about the data is provided in appendix C.*

6.2 Speaker dependent systems

A set of speaker dependent systems are trained from scratch to provide an upper limit to how high a performance can be achieved for a given speaker. The performance of the adapted models and the models trained from scratch are therefore naturally compared. A speaker dependent system is trained for each of the ten speakers in the adaptation group. In figure 6.3 the part of figure 3.1 that concerns the training of these speaker dependent systems has been repeated.

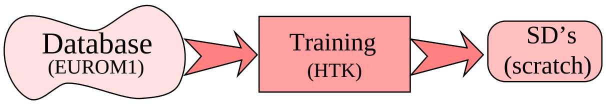

Figure 6.3: *Implementation of speaker dependent systems trained from scratch (SD)*

The feature processing is similar to the one applied in the speaker independent case, and each speaker dependent system is trained from scratch using the same training procedure as the one described above, where the number of mixtures is increased successively between re-estimations. The data are taken from the training part of the respective speakers as defined in figure 6.1. Again the number of mixtures has been varied. A summary of the specification for the speaker dependent systems is given in table 6.2. The tests of the speaker dependent systems are described in section 7.2.

6.3 Speaker adaptation system

Both the performances of the speaker independent system and of the speaker dependent systems described above are compared to the performance of the adapted models. The speaker independent models are also used as the initial models that are sought adapted to each of the adaptation speakers. This is illustrated in figure 6.4 which reviews the speaker adaptation parts from figure 3.1.

The topology and feature representation of the resulting speaker dependent models are similar to the one used for the speaker dependent systems trained from scratch. The adaptation material is

Type of parameter	Chosen value
Feature type	12 MFCC with 12 derivatives
Model topology	straight through, left-right
Number of states	three[a]
Number of mixtures	2 - 10
Training data	train data for adaptation speakers
Testing data	test data for adaptation speakers
Number of speakers	1

[a] There are used six states in the HMM for the 'SIL' class and one states in the HMM for the 'SP' class.

Table 6.2: *Summary of the parameters specifying each of the speaker dependent systems trained from scratch. A system is trained for each of the ten adaptation speakers.*

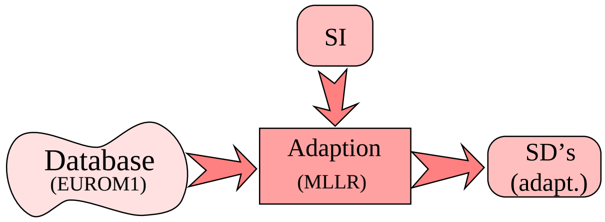

Figure 6.4: *Adaptation of speaker independent system to speaker dependent system (Adapted SD).*

in general taken from the test parts for the adaptation speakers and the adapted models are tested using material taken from the training parts for the adaptation speakers. This division is chosen in order to get as much material for testing the models as possible, and because only a small amount of adaptation material is necessary. However in some of the test a large amount of adaptation material is needed why the training part is used for adaptation and the smaller testing part is used for test.

6.3.1 Divisions into regression classes

To get the full benefit of the Maximum Likelihood Linear Regression technique, regression classes should be used. As described in chapter 5 this enables that even though phonemes are not present in the adaptation material, they are still adapted with a transformation matrix estimated with data present in the same regression class. A good choice of how the models are divided into regression classes is of course crucial and highly dependent upon the available adaptation material.

In the current project four different types of divisions have been tested. All of the divisions are based on phonetic classes, and they use 1, 2, 10 and 33 regression classes respectively. Table 6.3 lists the divisions. The outer limits have been to use either a global regression class, where all models are adapted with the same transformation matrix, or an individual regression class division where each model constitute its own regression class. In the case of having two regression classes, the models are divided into consonants and vowels. The ten phonetic classes used for defining the last division is based upon the phonetic classes described in the specification of the phonetic transcription of the Danish EUROM.1 database given in the SAM STAGE REPORT Sen.3 [Wells *et al*, 1992]. In appendix C the ten classes are listed together with the corresponding phonetic symbols in table C.1.

Number of regression classes	Type of division
1	Global
2	vowel/consonant
10	phonetic classes (table C.1)
33	Local

Table 6.3: *Specification of the different divisions into regression classes used in the current project.*

6.3.2 The `MLLR` program

The main part of the adaptation system is the software implementation of the Maximum Likelihood Linear Regression technique. The program is implemented using the C programming language and is referred to as `MLLR` throughout the thesis. The theory of the technique is described fully in appendix A, and the implemented mathematical algorithms were listed in section 5.4.

Requirements

The implementation of the program has mainly been based on the following requirements

- Enable usage of full and diagonal transformation matrix.
- Enable usage of offset term.
- Enable usage of different regression class divisions (global, individual or another specific division)
- Enable the saving of transformation matrixes.
- Enable the possibility to ignore certain groups of phonemes in the estimation of the transformation matrix.
- Enable the same flexibility with regard to file names, extension names, paths, file formats, usage of script files etc. as the HTK1.5 tools provide.

When `MLLR` is called with an empty command line a list of syntax and options is printed. In figure 6.5 a listing of the output from the program is given.

The program is controlled via the command line in a way similar to the HTK1.5 tools to facilitate the interactions with those. The procedure of dividing the HMM's into regression classes has not been implemented as a decision tree but rather a fixed division controlled via a command line option. This will in general not be an optimal solution for a specific application, but has been considered sufficient for the investigations covered in the current thesis.

6.3.3 Computational considerations

Inversion of matrixes using Singular Value Decomposition

As seen from the mathematical algorithms listed in section 5.4 both the re-estimation of a full and of a diagonal transformation matrix involves the inversion of a matrix. In the case of having an offset term of zero ($\omega = 0$) combined with a diagonal transformation matrix, this inversion is trivial as the matrix to be inverted is reduced to a diagonal matrix. However in the remaining cases a more general inversion method has to be used.

```
USAGE: MLLR [options] hmmList adaptFiles
Option                                 Default
-R r   Use multiple regression classes  global
       r=0: global
       r=1: euromdiv2
       r=2: euromdiv10
       r=3: local
-D     Use diagonal transf. matrix       full
-M m   Group of labels to ignore         none
       m=x: phonetic class x , 0<x<9
       m=1000: none
       m=100: vowels
       m=200: consonants
-i s   Extension for SI models           none
-I d   Dir to find SI models            current
-l s   Extension for label files          lab
-L d   Dir to find label files           current
-m t   Offset term for W                  0
-o s   Extension for SD models          as src
-O d   Dir to save SD models            current
-F fmt Set data file format to fmt        HTK
-G fmt Set label file format to fmt       HTK
-P f   Set the pruning threshold to f     0.0
-S s   Set script file to s              none
-T t   Trace level                        0
-w s   Set W-file to s                   none
-W d   Dir to save W files              current
```

Figure 6.5: *Usage of the `MLLR` as presented when the program is called with an empty command line. The `hmmList` is a file containing a list of the names of all models. A similar file is required by several HTK1.5 tools.*

As the matrix that is to be inverted is likely to be close to singular for small amounts of adaptation data [Leggetter, 1995] it has been chosen to use a very robust inversion algorithm based on a Singular Value Decomposition (SVD) of the matrix.

The SVD technique decomposes a $m \times m$ matrix $\boldsymbol{A}$ into three matrixes as

$$\boldsymbol{A} = \boldsymbol{U}\ \boldsymbol{w}\ \boldsymbol{V}' = \boldsymbol{U}\ \mathrm{diag}[w_1, w_2, \ldots, w_m]\ \boldsymbol{V}' \tag{6.1}$$

where $\boldsymbol{U}(m \times m)$ and $\boldsymbol{V}(m \times m)$ are matrixes that satisfy

$$\boldsymbol{U}'\boldsymbol{U} = \boldsymbol{I}_m\ ;\ \boldsymbol{V}'\boldsymbol{V} = \boldsymbol{I}_m \tag{6.2}$$

The diagonal elements $\{w_1, w_2, \ldots, w_m\}$ are named the singular values of $\boldsymbol{A}$. $\boldsymbol{U}$ is the left singular matrix and $\boldsymbol{V}$ is the right singular matrix of $\boldsymbol{A}$.

The pseudo-inverse of $\boldsymbol{A}$ is defined as

$$\boldsymbol{A}^{-1} \simeq \boldsymbol{V}\ \mathrm{diag}[1/w_1, 1/w_2, \ldots, 1/w_m]\ \boldsymbol{U}' \tag{6.3}$$

When $\boldsymbol{A}$ is singular, the pseudo-inverse equals the right inverse. In the case where $\boldsymbol{A}$ is non-singular the pseudo-inverse represents the best solution in a least squared error sense [Johnson and Duggeon, 1993].

The C implementation used in the current project is taken from [Press *et al*]. It has been tested against the SVD function in Matlab.

Calculation of mixture component occupancy probability using the Forward-Backward algorithm

As outlined in chapter 5 the fundamental variable for the MLLR technique is the mixture component occupancy probability.

This variable can be calculated using an algorithm known as the Forward-Backward algorithm. It is often applied as a part of the re-estimation procedure of HMM parameters. The implementation used in the current project is the one used in the HTK1.5 tool `HERest` that performs a Baum-Welch re-estimation of HMM parameters. In appendix B the forward-backward algorithm is described briefly.

Chapter 7

Test results

The previous two chapters described the implementation of the MLLR system. This chapter deals with the tests that have been carried out using the MLLR system. Section 7.1 briefly describes the general test procedures, and sections 7.2 to 7.5 describe the tests in details.

7.1 General test procedures

The complete MLLR system is configured by specifying a number of different factors such as the topology of the hidden Markov models, the amount and type of the adaptation material, the speaker adaptation method etc. For each fixed configuration of the MLLR system, a complete test procedure consists of

1. Adapting the speaker independent models using `MLLR`
2. Measuring the performance of
   - The speaker independent models
   - The speaker dependent models trained from scratch
   - The adapted speaker dependent models
3. Comparing the obtained performances

The tests are carried out using the HTK1.5 tools, `HVite` and `HResults`. The output from the last tool is a correctness measure and an accuracy measure. The correctness measure is the percentage of correctly recognised phonemes

$$\%\text{Correct} = \frac{H}{N} \times 100\% \tag{7.1}$$

where H is the number of correct labels, and N is the total number of labels in the defining transcription files. The accuracy takes into account the number of phoneme insertions:

$$\text{Accuracy} = \frac{H - I}{N} \times 100\% \tag{7.2}$$

where I is the number of insertions.

For matter of clarity the test procedures are grouped in four test series as sketched below.

A preliminary test series where the number of re-estimations of the transformation matrix is fixed. The determination of four different model topologies that are used in further tests is also done in this test series. It is described further in section 7.2. In the second test series, the *amount of adaptation data* is varied and the effect upon the performance is investigated. The impact of factors like the type of transformation matrix, the offset term and the topology of the initial models are also tested in the second test series. The detailed description is given in section 7.3. Thirdly, different *regression class divisions* are investigated and the results are presented in section 7.4. The last test series focuses on the importance of *the phonetic contents of the adaptation data*. This series is described in section 7.5.

Finally the effect of varying the speaker is investigated in that all tests are carried out on each of the ten adaptation speakers. Section 7.6 summarises on the observed speaker variations.

7.2 Preliminary tests

The preliminary tests have all been carried out in order to choose a suitable configuration of the MLLR system for the remaining MLLR series. The preliminary tests have addressed the determination of two factors:

- The number of times that the transformation matrix has to be re-estimated with the `MLLR` program.
- The four model topologies that will be used in further tests. The choice of the four mixture numbers is based on the performances of the corresponding speaker independent models. The object is to choose a set of model topologies that covers a fairly wide range of performances.

7.2.1 Results

The effect of varying the number of iterations

The Maximum Likelihood Linear Regression defines a number of re-estimation equations for determining the transformation matrixes. The amount of adaptation data will typically be small, and it is therefore important to determine an appropriate number of re-estimations. Too many re-estimations can cause the transformation matrixes to become too well fitted to the adaptation data and fail to transform the test data well. On the other hand too few re-estimations might not exploit the potential of the algorithm very well. A number of test procedures were carried out, where, for a single speaker, the performance was measured for different number of re-estimations of the transformation matrix. The results are plotted in figure 7.1.

At the two left-hand plots a relatively constant performance is seen as the number of iterations is increases. However for the right-hand plot where only 2 seconds of adaptation material is used, the performance is seen to drop from when 4 iterations is used. Based on these tests it is therefore chosen to perform three re-estimations of the transformation matrixes.

The performance of the speaker independent and dependent systems

Both the speaker independent and the speaker dependent systems trained from scratch are trained using HMM's with different number of mixtures. As described in chapter 6 they vary from 2 to 10.

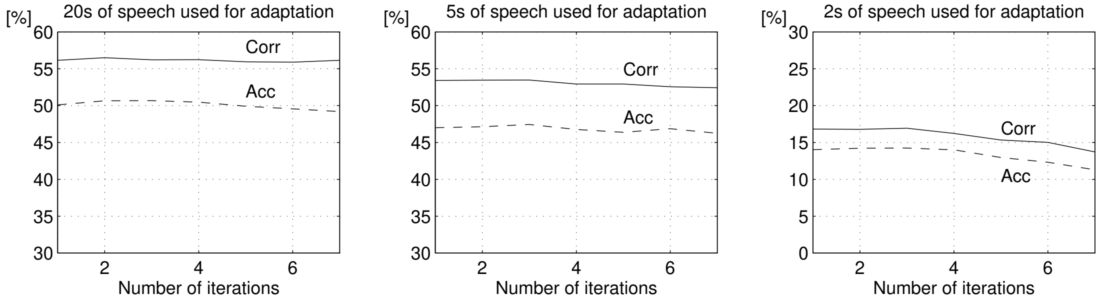

Figure 7.1: *Obtained performances in correctness and accuracy as a function of number of iterations used for estimation of the transformation matrix. Tests are carried out using for three different lengths of speech files. A global regression class, a full transformation matrix and a 5 mixture model is used. All results are for speaker 'jd'.*

The performance of these systems are tested in order to choose four model topologies that are used in those succeeding test procedures, where the effect of different qualities of the initial models are investigated. The four mixture numbers are chosen in such a way that they represent different initial model performances. The measured performances for both kinds of systems are illustrated in figure 7.2.

The two left-hand plots in figure 7.2 show the averaged results from testing the speaker independent systems with the reference speakers and the adaptation speakers respectively. The plots are almost identical which suggests that the models are indeed speaker independent. As expected the performance of the speaker independent system increases with growing model complexity. The largest improvements are seen for the models with a low number of mixtures. For the models with a relatively high number of mixtures the performance curve seems to flatten off at a level around 54% correctness. Using 10 mixtures is considered as a reasonable high number of mixtures, and it is therefore chosen as the upper limit.

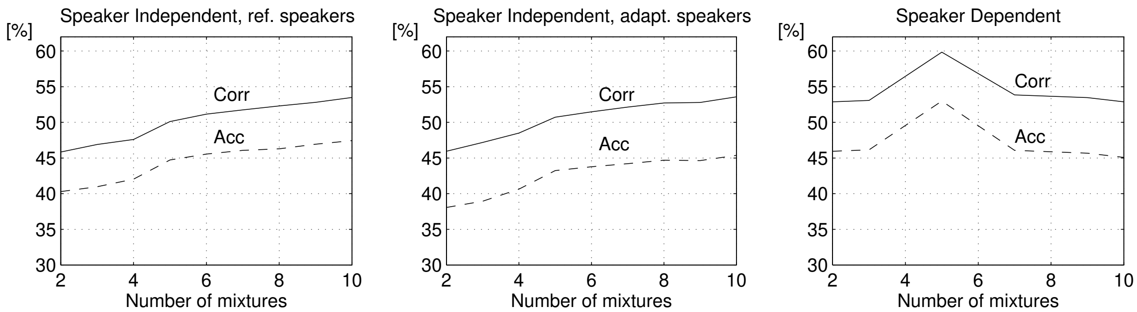

Figure 7.2: *Results of the speaker independent systems tested with the reference speakers and the adaptation speakers respectively, and of the speaker dependent systems tested with the adaptation speakers. All for varying number of mixtures in the Markov models. The results for the speaker dependent systems are averages over the ten adaptation speakers. Results are given in percentage correctness and percentage accuracy.*

The results of measuring the performance of the speaker dependent systems is shown in the right-hand plot in figure 7.2. As for the speaker independent system, the performance increases for low mixture numbers as the number of mixtures is increased. But after 5 mixtures the performance drops drastically. This tendency is caused by the amount of available training material. Each of the speaker independent models are trained with more than one hour's speech material, whereas the speaker dependent models only have up to ten minutes available per speaker. Consequently there is not enough material to estimate any of the high mixture speaker dependent models well enough, and the performance gets worse than what is obtainable with the low mixture HMM's.

Comparing the performance of the speaker independent systems with the low mixture speaker dependent models shows that the speaker dependent models give a better performance as is expected. The differences between the two types of models are in the range of 7% - 9%. Based on the results from figure 7.2 it is chosen to use 3, 5, 7 and 9 mixtures as the four topologies used in subsequent test series. The correctness of the speaker independent models with these mixture numbers vary from 47.2% to 52.8%.

7.3 Tests with varying amounts of adaptation data

The aim of this test series is to investigate the effect of varying the amount of data for a number of different configurations of the `MLLR` program. All tests are carried out for all adaptation speakers and using a global regression class. The following factors are varied

- The amount of adaptation data.
- The model topology; the four mixture numbers 3, 5, 7 and 9 mixtures are tried out.
- The type of transformation matrix; both diagonal and full transformation matrixes are used.
- The offset term; ($\omega = 0$ or $\omega = 1$).

Data

Speech files of different lengths are needed in this test series. The data are extracted from the available speech signals in the EUROM.1 database using the HTK1.5 tool `HCopy`. It enables the extraction of a file segment of a specified length, and there are options for creating a corresponding label file with the phoneme transcription for the extracted segment. A number of tests was done in order to find the relevant sizes of the segments. It was found that the performance as a function of the amount of adaptation material varies the most when the lengths of the adaptation files are in the range of $2s$ to $5s$ after which the performance stabilises. To reflect this performance function is has therefore been chosen to test with a set of 9 segment files of the following lengths: $1.5s$, $2s$, $2.5s$, $3s$, $4s$, $5s$, $6s$, $10s$ and $20s$. The segment files have been extracted from the speech files at an arbitrary point, but in such a way that the beginning and ending silence segments are avoided. To compensate for any unfortunate segment examples, three file segments of each of the above specified lengths are used. This makes a total of 27 speech file segments that are tested for each speaker, corresponding to 10 file segments of each of the specified lengths.

With the above described procedure it is assumed that the phoneme present in the segments are representative for the phoneme distribution found in the complete database. Table 7.1 lists the averaged number of labels in the segments.

Number of sec. in segments	1.5	2	2.5	3	4	5	6	10	20
Average number of labels	19	24	30	38	46	58	68	113	221

Table 7.1: *Average number of labels in each file segment.*

As `HCopy` cuts the file segments out on a per frame basis the leading and final phoneme might be truncated. The calculation of the mixture component occupancy probability for the present frames are unaffected by this truncation.

Test combinations

Combining the four topologies, the two transformation matrix types and the two offset term values gives 16 test combinations. Each of these involves testing with ten speakers and with the set of 9 times 3 different file segments. The computational overhead for a single of the 16 combinations is relatively large (in the order of two to four days with the computational resources available for the current project), and it is therefore chosen to limit the test series to include a total of 10 out of the 16 combinations. It is done under the assumption that the performed tests are representative enough to constitute a basis for the conclusions. Table 7.2 illustrates which of the test combinations that are carried out.

	3 mixtures	5 mixtures	7 mixtures	9 mixtures
Diagonal transformation matrix, $\omega = 0$	$\surd$	$\surd$	$\surd$	$\surd$
Diagonal transformation matrix, $\omega = 1$		$\surd$		
Full transformation matrix, $\omega = 0$	$\surd$	$\surd$	$\surd$	$\surd$
Full transformation matrix, $\omega = 1$		$\surd$		

Table 7.2: *The ticked test combinations are carried out. Each test combination is done for each of the ten adaptation speakers and for a varying amount of adaptation data.*

All test combinations using a zeroth offset term are performed. But it is chosen to limit the number of test combinations, where the offset term is utilised. This is done because all data used in the current work are taken from the same database. They are therefore expected to have very small variations in recording conditions, which justifies the assumption that the offset term is negligible. Furthermore the processing is faster when a zeroth offset term is used, especially in the case of using a diagonal transformation matrix.

The model topology with 5 mixtures is chosen when an offset term is used, because this is the highest number of mixtures for which enough training material is available to train the speaker dependent models from scratch. This is illustrated at the right-hand plot in figure 7.2, where the performance curve is seen to have a maximum for 5 mixtures.

7.3.1 Results

As described above the main object of this test series is to investigate the effect of varying the amount of data used in the adaptation process. To get a more detailed picture of how the MLLR technique performs with small amounts of adaptation material, a number of factors concerning the configuration of the technique are investigated too. This section describes the results obtained from the complete test series. First each of the factors are described separately. Finally a discussion of the effect of varying the amount of adaptation data is given based on the performance results from the complete test series.

The effect of using a full versus a diagonal transformation matrix

In figure 7.3 the results of the tests concerning the effect of using a full versus a diagonal transformation matrix are presented as well as the results of using the offset term or not. The figure illustrates four of the test combinations described above corresponding to the second column in table 7.2. Each of the four plots shows the correctness in percentage as a function of the number of seconds of adaptation material, and all results are averaged over the ten adaptation speakers. The model topology has 5 mixtures in all cases and a global regression class is used.

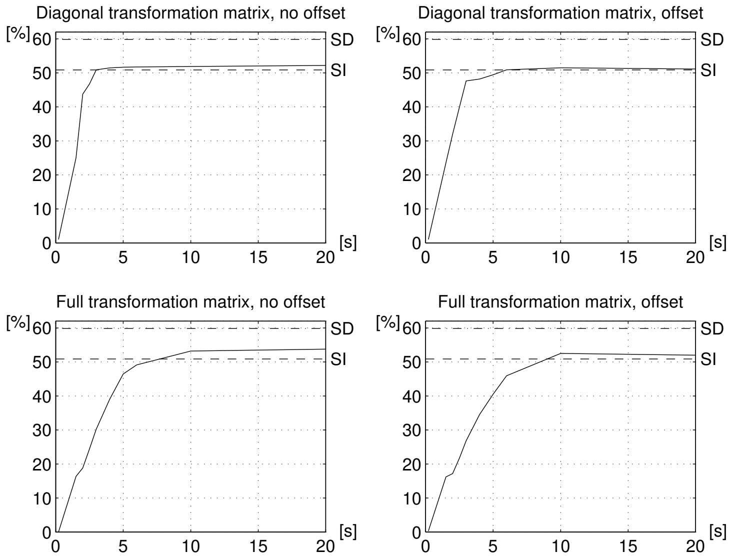

Figure 7.3: *Results obtained by testing combinations of diagonal versus full transformation matrix and using a zeroth offset term or not. A 5 mixture model topology is used together with a global regression class. In each plot the performance levels for the corresponding speaker independent models (SI) and speaker dependent models (SD) are shown too.*

The plots are organised in pairs. The two upper plots show results from when a diagonal transformation matrix is used in the adaptations. The lower plots similarly correspond to the usage of a full transformation matrix. In each pair the left-hand plot is for the case when the offset term is unused ($\omega = 0$), and the right-hand plot shows results from tests where the offset term is used ($\omega = 1$). In the next section the effect of the offset term is discussed further.

The four plots indicate the general shape of the performance as a function of the amount of adaptation data. This shape is found in all tests carried out in this test series. It shows a rapidly growing performance as the amount of adaptation data is increased. After a certain amount of data, the performance seems to stabilise at a relatively fixed level. For very few seconds of data the performance is much lower than the baseline performance for the initial speaker independent models. This reflects the fact that with only a few representations of phonemes in the adaptation data, the transformation matrix is estimated very poorly, which results in low performances. When enough adaptation data are available the effect of the speaker adaptation becomes clear, and the curve climbs above the baseline level. However increasing the amount of adaptation material further does in general not have any significant influence upon the performance obtained. This is due to the fact that once the transformation matrix has been estimated well enough, more data does not increase the accuracy of this estimation significantly.

From the plots in figure 7.3 it is seen that for a certain amount of adaptation data, the speaker adaptation does in all four test combinations improve the performance of the initial speaker independent models. Comparing the plots for the two types of transformation matrixes shows that by using a diagonal matrix the improved performance is reached, when using only 3 seconds of data. Whereas when the adaptation is done with the full matrix, it takes around 7 seconds of data to get a positive effect of the speaker adaptation . With the full matrix the performance seems to stabilise at a higher level, than obtainable with the diagonal transformation matrix.

The steepness of the performance curves for the two types of transformation matrixes reflects very well, that in the full matrix case far more parameters are to be estimated than in the diagonal case, why more data are needed before a well enough estimation is achieved. For the same reason the few parameters in the diagonal matrix are well estimated for less material. The final performance, when it stabilises for a higher amount of adaptation data, is better for the full matrix than for the diagonal matrix. This indicates that a diagonal transformation matrix is too restricted to account for general inter speaker variations.

The effect of using an offset term

The right-hand plots of figure 7.3 show results from tests performed, when the offset term is set to one ($\omega = 1$). The same shape of the performance function as described above is seen. The figures show that when using a diagonal matrix the performance curve crosses the baseline performance when in average 5 seconds of adaptation material is used, which is approximately two second more than what is observed, when no offset term is used. When a full transformation matrix is used the same situation is observed; the amount of adaptation material it takes for the performance curve to cross the baseline performance increases from around 7 seconds to around 8 seconds.

The increase in needed adaptation material can in both cases be explained by the extra number of parameters that an offset term introduces. For the case of a diagonal transformation matrix the number of parameters are doubled from n to $2n$, where n is the number of features. When a full transformation matrix is used the number of parameters to estimate increases with n from $n \times n$ to $n \times n + 1$.

The effect of using different qualities of the initial models

Four different model topologies are tested in order to investigate, whether the quality of the baseline performance obtainable for the initial models is reflected in the achievable effect of doing speaker adaptation. Figure 7.4 illustrates the results from the test combinations corresponding to the first row in table 7.2, and figure 7.5 corresponds similarly to the third row in the same table.

In both figures the plots show performance curves with the same shape as described above. In each figure the four plots represent one of the four topologies tested: 3, 5, 7 and 9 mixtures and for all cases the speaker adaptation is seen to have a positive effect.

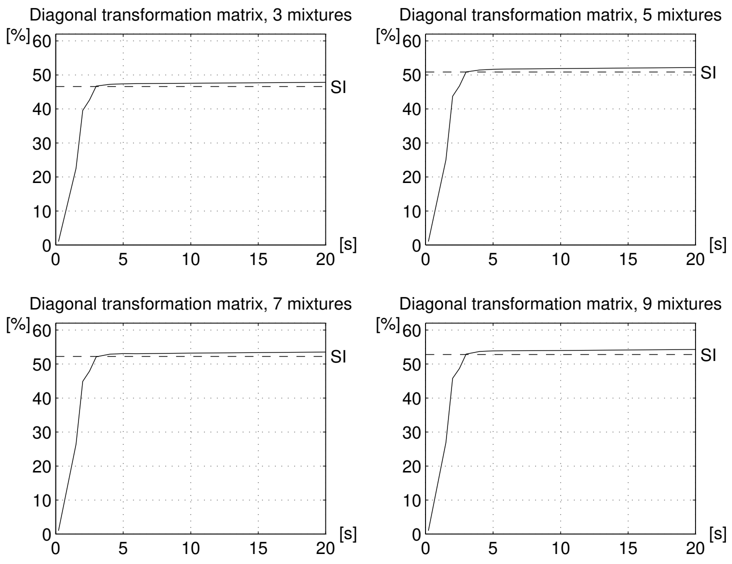

Figure 7.4: *Measuring the effect of varying the amount of adaptation data using initial models with different mixture numbers. A* ***diagonal transformation matrix*** *is used with an offset term set to zero. Each recognition result is averaged over all ten adaptation speakers and a global regression class is used.*

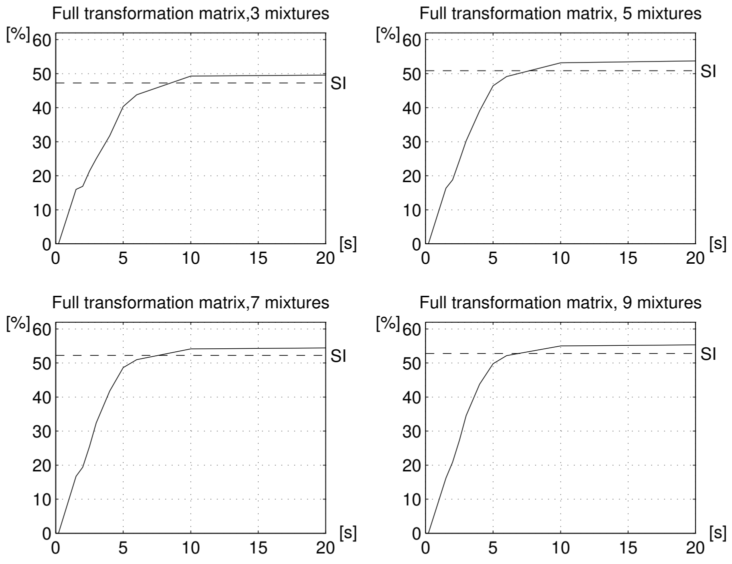

Figure 7.5: *Measuring the effect of varying the amount of adaptation data using initial models with different mixture numbers. A* ***full transformation matrix*** *is used with an offset term set to zero. Each recognition result is averaged over all ten adaptation speakers and a global regression class is used.*

When a diagonal transformation matrix is used, the amount of adaptation data it takes to make the adapted models perform as well as the initial models is fairly constant over the tested topologies. Around 3 seconds of material is needed. A variation is however seen when the effects of the speaker adaptation are compared. For the 3 mixture models the increase in correctness is only around 0.5% whereas for the 5, 7 and 9 mixture models the correctness increases with approximately 1.5%. The largest effect of the speaker adaptation is achieved with the 9 mixture models, where the phoneme correctness improves with 2.5%.

For the full transformation tests that are shown in figure 7.5 a comparison of the plots for the four different mixture numbers shows that the amount of data it takes to get a positive effect of the speaker adaptation is in the range of 7 to 8 seconds, where the 3 mixture models seem to require the most adaptation data. For all topologies the performance of the speaker adapted models stabilises at a level approximately 2.5% higher than the corresponding baseline for the initial models. A maximum of a 8.7% relative improvement is achieved for the 5 mixture models.

Comparing the results from the tests with the two types of transformation matrixes shows that a positive effect of performing a speaker adaptation is achievable for all the tested model topologies. The amount of data needed before this effect becomes clear does not vary significantly with the mixture numbers. For the diagonal case the final performance level is slightly lower for the 3 mixture models than for the 5, 7 and 9 mixture models. For the full transformation matrix the increase in recognition performance is similar for all the tested models.

7.3.2 Discussion

In this test series the effect of doing speaker adaptation using the MLLR technique was investigated for very small amounts of adaptation material. The results described above are all similar to those reported in [Leggetter, 1995]. Based on the results the following can be concluded:

Type of transformation matrix The tests in this test series have shown that several things must be taken into consideration when choosing the type of transformation matrix in a given application. Apart from the characteristics of the performance curve as described above, the processing time involved is important. The re-estimation equations for the diagonal case, especially when the offset term is set to zero, involve considerably fewer computations than for the full case, as described in chapter 5.

Offset term The general shape of the performance curve is preserved as expected, when the offset term is set to one compared to when it is set to zero. The tests also show that it takes a small amount more of adaptation material before the adaptation is seen to have any effect.

Model topology No clear tendency is seen when testing with different model topologies of the initial models. The resulting effect of the speaker adaptation does not change significantly with the number of mixtures in the initial models.

7.4 Tests with different regression class divisions

This test series is carried out in order to investigate the effect of using different regression class divisions. In all tests described above a single, global regression class has been used. However an even better performance for the MLLR method can be obtained if the mono-phones are divided into a set of regression classes. The assumption when using a global regression class is that all mono-phones undergo approximately the same transformation when they are adapted to a new speaker. Obviously in the general case this will be too restrictive, and better results can be obtained if more regression classes are allowed. The upper limit of the number of regression classes is of course regulated by the available amount of adaptation data. When too few representations are found in a specific regression class the transformation matrix is estimated poorly and the result will often be that the adapted models turn out to be worse than the initial models. For a given set of adaptation data a performance optimal regression class division exists.

The object of this test series is not to define such an optimal division, but rather to investigate different divisions based on the phonetic classes as described in chapter 6. Additionally an analysis of transformation matrixes is carried out. The transformation matrixes used are those needed to make an individual transformation of each mono-phone. The hypothesis is that these individual phoneme specific transformation matrixes implicitly incorporate information about which transformation is needed for each phoneme. Comparing these transformation matrixes might therefore then lead to a suggestion of which phonemes undergo similar transformations, resulting in a data driven regression class division.

7.4.1 Results

Using a phonetic based regression class division

In figure 7.6 the performance results from doing speaker adaptation using four different regression class divisions based upon phonetic classes are shown. In table 6.3 on page 49 the different regression class divisions are listed.

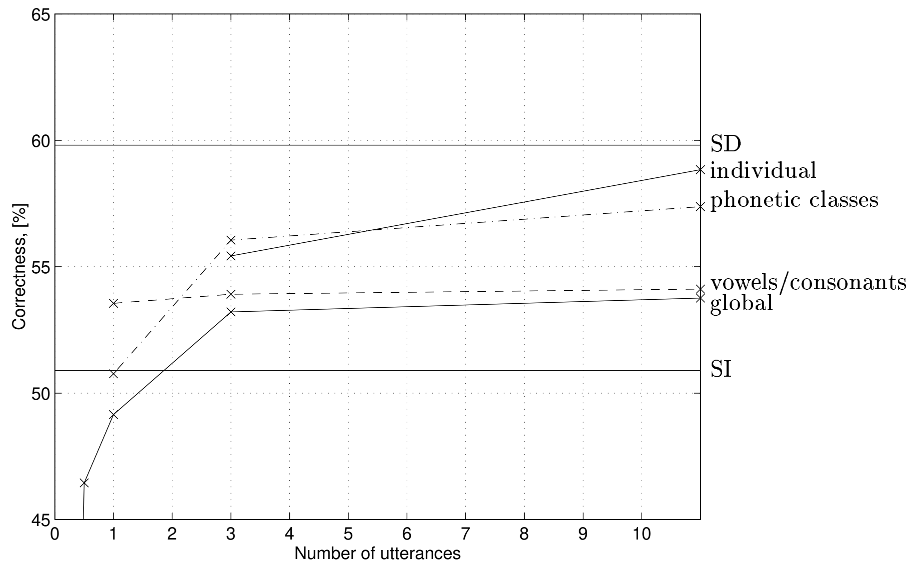

Figure 7.6: *Results from testing with different regression class divisions given in percentage phoneme correctness. A full transformation matrix and a 5 mixture model are used. The performances of the speaker independent (SI) and the speaker dependent (SD) models trained from scratch are given as baselines.*

To test the performance for different amounts of adaptation data the tests are carried out for one, three and eleven utterances. Each utterance has a length of approximately $20s$. As the amount of adaptation material is reduced, the effect of the adaptation decreases. At some point the algorithm fails because eventually there will not be any material assigned to one of the regression classes. For the highest amount of adaptation material, the tendency is as expected that the higher number of regression classes used, the better the performance. The best effect is for the individual regression classes, where the phoneme correctness is improved relatively by 15.6%. The division into ten broad phonetic classes does also perform very well with a relatively improvement of 12.8%. In the case where the phonemes are only divided into vowels and consonants, the effect is seen to be only a little better than what is achieved with a global regression class. This indicates that the differences between phonemes are only slightly reduced when separating vowels from consonants.

Analysis of transformation matrixes

The above used regression class divisions are all based upon a split of the phonemes into a number of broad phonetic classes. However it is likely that this division is not the ideal. In the current work it has been investigated whether transformation matrixes estimated using individual regression classes can be utilised to determine a data driven regression class division. The underlying idea is that comparing the transformation matrixes estimated for each of the individual phonemes might lead to a data driven grouping of them. The used transformation matrixes are estimated using the training data for the adaptation speakers. They are saved in separate files by the `MLLR` program and further processed in Matlab. In the following the transformation matrix for phoneme p, named $\boldsymbol{W}_p$ has elements w_{ij}^p and is an $(n \times n+1)$ matrix.

The following three ways of comparing the transformation matrixes have been investigated:

1. The Euclidean distance between two transformation matrixes for phonemes p and p':

$$\mathrm{dist}(p,p') = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n+1}\left(w_{ij}^{p} - w_{ij}^{p'}\right)^2} \tag{7.3}$$

2. The stress as a measure of the correspondence between two matrices [Lindberg, 1994]:

$$\mathrm{stress}(p,p') = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n+1}\left(w_{ij}^{p} - w_{ij}^{p'}\right)^2}{\sum_{i=1}^{n}\sum_{j=1}^{n+1} {w_{ij}^{p}}^{2}}} \tag{7.4}$$

3. The correlation between two vectors [Ross, 1987]:

$$\mathrm{corr}(p,p') = \frac{\mathrm{cov}(\boldsymbol{w}_p, \boldsymbol{w}_{p'})}{\sqrt{\mathrm{cov}(\boldsymbol{w}_p, \boldsymbol{w}_p)\cdot \mathrm{cov}(\boldsymbol{w}_{p'}, \boldsymbol{w}_{p'})}} \tag{7.5}$$

Here $\boldsymbol{w}_p = \boldsymbol{W}_p \boldsymbol{I}$ where $\boldsymbol{I}$ is the unitary vector of size $(n+1)$ and $\mathrm{cov}(\boldsymbol{w}_p, \boldsymbol{w}_{p'})$ is the covariance of the vectors $\boldsymbol{w}_p$ and $\boldsymbol{w}_{p'}$.

The aim of the analysis of the transformation matrixes has been to find out whether some phonemes have more similar transformation matrixes than others, and if those similarities can be generalised over the ten adaptation speakers for which the analysis is carried out. First the simple Euclidean distance was calculated. However since some of the transformation matrixes tend to have occasionally very high peak values this simple distance measure is not useful. Therefore a second attempt was done using the stress measure as described above. It is a distance measure, where the size of the matrix to which the distance is measured is taken into account. The undesirable strong effects from the peaks in the transformation matrixes disappears when the stress is measured. The result of the analysis shows that no clear definitions of data driven regression class division can be done using the stress measure. The same results occur, when the correlation between transformation matrixes are computed. No strong correlations between any pair of transformation matrixes are seen generally over all ten adaptation speakers.

Based on the comparison of individually estimated transformation matrixes carried out in this work, no suggestion can be made of a data driven regression class division.

7.4.2 Discussion

In this test series the effect of using different regression classes has been investigated. Furthermore several attempts have been done in order to utilise individual estimated transformation matrixes to determine some general data driven regression class division. The results of the test series can be summarised to:

Broad phonetic definition of regression class division. For all tested regression class divisions the performance of the adapted models are improved considerably. As expected best effects are seen when each phoneme is transformed individually. The result was a relative improvement of 15.6%. Using ten broad regression classes also gives a very good improvement. In the case of just dividing the phonemes into vowels and consonants only a small improvement is seen. Therefore in a specific application the regression class division should be chosen as fine as possible given the amount of available adaptation material. In [Leggetter, 1995] similar observations are reported both when phonetic division classes are used and when the individual mixture components are allowed to go into different regression classes.

Data driven definition of regression class division. None of the investigated ways of comparing the individually estimated transformation matrixes can lead to a suggestion of a data driven transformation matrix. With the chosen methods no clear tendency was observable over the ten adaptation speakers. However it is believed that the transformation matrixes still contain useful information, as these matrixes are indeed those that have given the very high performance improvements described above, where they are used in the individual regression class division.

7.5 Tests with varying phonetic contents of adaptation data

When speaker adaptation is carried out in a supervised manner, the speaker is prompted for the speech which is to be used as adaptation material. Using this method it is possible to direct the phonetic contents of the data. It is therefore highly relevant to find out what the importance of the phonetic content of the adaptation material is. That is, whether some phonemes can be said to possess a potentially bigger amount of speaker specific information than others. This would naturally imply that such phonemes should be preferred in the speech used for adaptation. This test series addresses the problem in two ways

- The mean vectors for the speaker dependent models trained from scratch are compared in Matlab. This is done in order to investigate whether the mean vector parameters for some phonemes tend to vary more from speaker to speaker than for other phonemes. It has been chosen not to include the variances in the considerations.

- In turn a group of phonemes are ignored when the transformation matrix is estimated. The implementation of the `MLLR` program enables this functionality. The performances of the models adapted using these specially estimated transformation matrixes are then measured and compared.

7.5.1 Results

Variability of phoneme modelling between speakers

The point of view for the first analysis method is that speaker specific information can be defined as information that characterises the difference between the speech from two speakers. The idea is that this will be reflected in HMM's trained for the speakers in question. This analysis therefore compares the speaker dependent HMM's trained from scratch for each of the ten adaptation speakers. In figure 7.7 the results from the analysis is shown. The upper plot shows when the Euclidean distance between mean vectors is used as a measure of difference between HMM's. In the lower plot the correlation coefficient between two mean vectors is used. Both measures are calculated as specified in section 7.4.1. The calculation is carried out as following: For a given speaker and a given phoneme all mean vectors in the specific HMM are summed. This summed mean vector is then compared to the summed mean vectors from the corresponding HMM's for the other nine speakers; either by calculating the distance or the correlation coefficient. For the specific phoneme the inter speaker differences are summed. This procedure is carried out for all phonemes and plotted in figure 7.7.

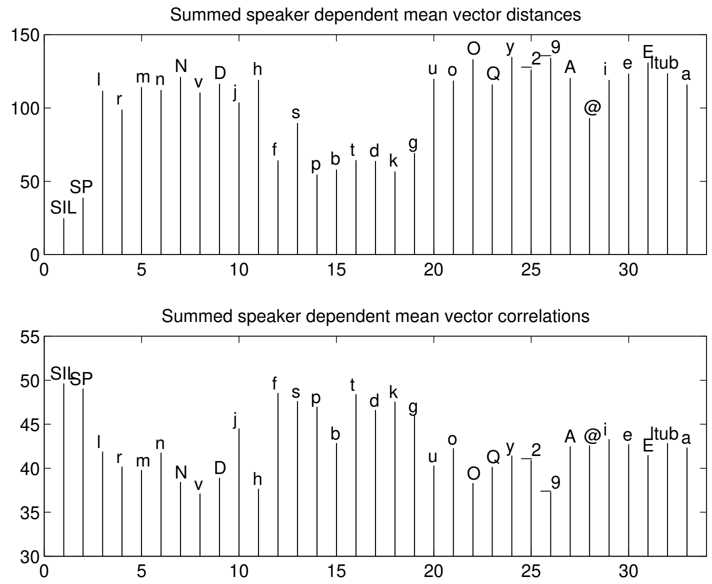

Figure 7.7: *Difference between mean vectors in speaker dependent HMM's averaged over all adaptation speakers. Upper plot shows summed Euclidean distance. Lower plot shows summed correlation coefficients. Description of the phonetic symbols are given in table C.1 on page 92.*

Together the two figures indicate how much the phonemes tend to vary between the ten tested speakers. It is clear that some phonemes does indeed vary more from speaker to speaker than others. As expected the silence model ('SIL') varies the least and the inter word silence model ('SP') has very little variance too. Also the group of plosives ('p','b','t','d','k','g') and the group of fricatives ('f','s') are seen to vary less than other phonemes. This can be explained by the fact that these are all phonemes pronounced with a relatively small cavity. Therefore it is likely that physiological speaker differences are less prominent for these phonemes. This is supported by the fact that all vowels have high mean vector distances and low correlation. The remaining consonants (liquids, nasals and approximants) are seen to vary almost as much as the vowels.

To give a rough estimate of the variations, the phonemes are ordered according to the results shown in figure 7.7. A ranking is carried out for both the distance and the correlation case, and based on these two ordered lists of phonemes a final ranking is done. The results is the following list with the most varying phonemes first:

_9, O, y, N, h, _2, E, u, D, Q, m, ltub, e, A, o, i,
v, r, n, a, l, @, j, g, b, s, d, t, k, p, f, SP, SIL.

and specified in broad phonetic classes:

Unrounded front vowels, *Back vowels*, *Rounded front vowels*, *Approximants*,
Nasals, *Plosives*, *Liquids*, *Central vowels*, *Silence models*.

Importance of different phonemes in estimation of transformation matrixes

The second approach to investigating the importance of the phonetic content of the adaptation material was to study what happens if a certain group of phonemes are ignored, when the transformation matrix is estimated. What is the performance of the resulting adapted models compared to the performance achieved if information from all phonemes is incorporated in the calculations?

Table 7.3 shows the measured performance when one of the phoneme classes are ignored during the estimation of the transformation matrix. The results are averaged over all adaptation speakers. A global regression class, a 5 mixture model and a full transformation matrix have been used.

Group of ignored mono-phones	Correctness	Relative improvement
None	53.8 %	0.0 %
Liquids	54.1 %	0.6 %
Nasals	53.2 %	-1.0 %
Approximants	54.0 %	0.5 %
Fricatives	53.8 %	0.4 %
Plosives	54.0 %	0.5 %
Back vowels	54.5 %	1.3 %
Rounded front vowels	53.8 %	0.0 %
Central vowels	53.8 %	0.0 %
Unrounded front vowels	54.2 %	0.7 %

Table 7.3: *Results in correctness from adapting with a transformation matrix estimated without using the data for the specified group of mono-phones.*

The last column in the table shows the improvement in performance relatively to what is obtained, when no phonemes are ignored. Only exclusion of the nasals from the calculations are seen to actually reduce the performance. When one of the remaining phonetic groups are ignored, the performance is either unaffected or improved. This might seem surprisingly but is an effect of the usage of a single global regression class. The increase in performance when a specific phonetic group is ignored indicates that the remaining phonemes are badly represented by the information which the specific phonetic class contributes with. It is seen from the table that most of the phonetic groups have a bad influence on the transformation of the other phonemes. A result which is not surprising and which is similar to what was concluded from the test series that investigated different regression class divisions described in section 7.4. There is was concluded that the more the mono-phonemes are separated out in regression classes the better the performance.

To further investigate to what degree the regression class division has influenced the above described results, another analysis was carried out. The results are shown in figure 7.8. In this test each single phoneme was ignored in turn. The plot shows performance results from the resulting adapted models, when the broad phonetic classes are used as regression class division. Because the test is very time consuming it was carried out for a single speaker (*an*) only. However the inter speaker differences observed for the above tests were small, and the results are therefore assumed to give a reasonable suggestion about the tendencies for most speakers.

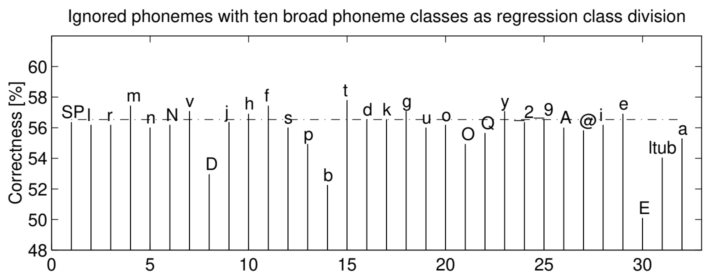

Figure 7.8: *Results from tests where a single phoneme is ignored during the estimation of the used transformation matrix. The tests are carried out for speaker 'an'. Phonemes are divided into ten regression classes, namely the ten broad phonetic classes. A full transformation matrix and a 5 mixture model topology is used. The horizontal line indicates the performance achieved, when all phonemes are included in the estimation of the transformation matrix and the same regression class division is used.*

Figure 7.8 shows more differentiated performance levels for the individual phonemes. Most of the phonemes vary around the performance level achieved for the case when all phonemes are included in the calculation of the transformation matrix. A few of the phonemes are still above this level indicating that they might still have a bad influence on the other phonemes in their regression group. This is especially clear for the phonemes: 't', 'f' and 'm'. The opposite tendency is observed for phonemes that lower the performance considerably when they are ignored in the re-estimation algorithm. Very low performances are obtained for the phonemes 'E', 'b', 'D', and 'ltub'. These phonemes do probably contribute with a relatively large amount of speaker specific information to the estimation of the transformation matrix. But it is important to note that the influences of the regression class division and the influence of the missing phonetic information when a single phoneme is ignored are not completely separated out in this test either. Therefore further investigations need to be done in order to make any definite conclusions. However the results obtained in this test series show some indications and are promising for further studies of the importance of the phonetic content of the adaptation material.

7.5.2 Discussion

This test series has included several approaches to the study of which phonemes that are likely to contribute the most with speaker specific information to the estimation of the transformation matrix. From the tests described above the following can be concluded:

Differences in speaker dependent HMM's Big variations in the speaker dependent models were especially seen for all types of vowels. The least varying group of phonemes was the fricatives and silence models. In order to be able to compensate for inter speaker differences it can therefore be concluded that especially vowels should be widely represented in the adaptation material. But also consonants like nasals and approximants are likely to prove important for a good adaptation.

Importance of phonemes in estimation of transformation matrix For the tests carried out in this work it can be concluded that there is indeed differences in the importance of the different phonemes in the estimation of the transformation matrix. The results of the tests reported here are all strongly dependent upon the type of regression class division used. Therefore further studies are needed to definitely determine the relatively importance of the phonemes.

7.6 Inter speaker differences

In order to investigate the variances between speakers all tests have in general been carried out on ten speakers. Where the results reported so far have been averaged over all speakers, this section will focus on the inter speaker differences that has been observed.

7.6.1 Results

In table 7.4 the relative improvements observed when testing with different regression class divisions are specified for the individual speakers. Also the relative improvements achieved when increasing the number of regression classes from one global to the 33 individual classes are included.

It is seen that for all the tested speakers, there is always a positive effect of performing the speaker adaptation. However the achieved relative improvements vary a lot. From 0.2% for speaker *an* to 12.9% for speaker *bl*, when a global regression class is used and the same tendency is seen for the individual regression classes. When taking into consideration, what the performance of the initial models is, it seems that higher relative improvement is achieved for small initial performances, but within the ten speakers there are a few exceptions. Comparing relative improvements for the ten speakers when going from a global to an individual regression class division shows that somewhat more equal results are observed except for the speaker *jo*. A plausible explanation could be that *jo* is the speaker with the highest initial performance. The smaller improvement could therefore be an indication of that speaker 'jo' is close to some upper performance limit, and the potential effect of doing speaker adaptation therefore naturally is restricted. That the relative improvement depends on the performance of the initial models seems reasonable as speakers that perform badly on the initial models have potentially more to gain from the adaptation, than speakers that are already fairly well modelled with the initial models. As mentioned in section 7.5.1 the tests where one of the phonetic classes was ignored in turn during estimation of the transformation matrix showed very few inter speaker variances. The levels of the results were of course highly speaker dependent, but the effects of ignoring different phoneme groups were very uniform for all speakers.

Speaker	Initial correctness	Relative improvement		
		Global	Individual	Global → Individual
an	52.43 %	0.2 %	11.9 %	11.7 %
ba	48.82 %	2.6 %	15.4 %	12.5 %
bl	49.14 %	12.9 %	22.0 %	8.0 %
eb	50.95 %	1.5 %	10.8 %	9.1 %
hm	45.90 %	12.5 %	20.7 %	7.3 %
il	56.27 %	0.4 %	11.1 %	10.6 %
jd	52.23 %	7.6 %	19.4 %	10.9 %
jo	59.42 %	9.4 %	12.6 %	2.8 %
js	52.14 %	1.6 %	10.1 %	8.4 %
ld	41.49 %	9.4 %	26.2 %	15.4 %

Table 7.4: *The effect of doing speaker adaptation specified for each of the adaptation speakers. The correctness of the initial models when tested with the specific speaker is shown together with the relative improvements when using a global and a individual regression class division. Also the relative improvement when going from using global regression classes instead of individual classes is shown. The adaptation is done with a total of 11 utterances.*

7.6.2 Discussion

Comparing results from the different speakers shows that for all the tested speakers adapting with the MLLR method can improve the performance for a sufficient amount of adaptation material. The relative improvement is however varying considerably between speakers. There seems to be a small tendency towards that speakers which perform badly on the initial models gain relatively more from speaker adaptation with the MLLR method. The tests that are carried out in order to investigate the importance of the phonetic contents of the adaptation material show very small inter speaker differences.

Chapter 8
Conclusion

This thesis has investigated speaker adaptation of hidden Markov models using the Maximum Likelihood Linear Regression technique. In the introduction the main object of speaker adaptation techniques was identified as to maximise the performance of the resulting speaker dependent system using as little data as possible.

The work presented in this report has addressed the problem of how the adaptation data can be utilised most effectively for a specific speaker adaptation method. A number of speaker adaptation approaches and techniques have been briefly described in chapter 2. Based on this study it was chosen to perform the investigations in the current project within a framework implementation of the Maximum Likelihood Linear Regression technique. It is a recently proposed technique that transforms mixture components of hidden Markov models with a transformation matrix estimated using the maximum likelihood principle. By introducing the idea of regression classes as sets of mixture components, the technique enables several mixture components to be transformed with the same transformation matrix. This allows transformation of all phonemes, even those for which no examples are present in the adaptation data. The project definition specifies a system which has been implemented in order to investigate the effect of doing speaker adaptation under influence of a number of factors.

Previous studies dealing with speaker adaptation techniques have focused mainly on two factors likely to influence the data effectiveness: the usage of adaptation material and the configuration of the speaker adaptation method. In this project a new approach has been followed as well; namely to investigate whether the phonetic contents of the adaptation material has any influence on the effect of the adaptation. Specifically the following three issues have been addressed

- The effect of using different amounts of adaptation data.
- The effect of using different regression class divisions.
- The importance of the phonetic content of the adaptation material.

Furthermore different parameters specifying the Maximum Likelihood Linear Regression technique as well as the inter speaker variations have been investigated.

The data are provided by the Danish EUROM.1 database that contains speech from a total of 60 speakers. In general adaptation tests were carried out for ten speakers. The database is transcribed at phoneme level why all conclusions are based on phoneme recognition performances.

The Maximum Likelihood Linear Regression technique has been implemented in a C program named `MLLR`. It constitutes the main part of the MLLR system. HTK1.5 tools were used for training and testing hidden Markov models within the system.

The underlying principle of all tests carried out within the implemented MLLR system is to vary one of the numerous factors that specify the exact adaptation that takes place. This way the data-effectiveness of the MLLR technique can be investigated.

The amount of adaptation data

An extensive test series has been carried out in which the amount of adaptation material was varied together with factors such as: the type of transformation matrix, the offset term and the model topology. When using a global regression class only a few seconds of adaptation material is necessary to obtain a positive effect. With a diagonal transformation matrix only 3 seconds are needed, and for a full transformation matrix 7 seconds are necessary. This difference reflects the fact that when using a full transformation matrix far more parameters are to be estimated than with the diagonal matrix, and therefore more adaptation material is needed. The tests also showed that as the amount of adaptation data is increased, once the transformation matrix is well enough estimated, the performance does not increase further even though the amount of adaptation material is increased.

By introducing the offset term inter speaker variations such as for instance different recording conditions can be modelled. However in this project all data were taken from the same database. As expected tests also showed that enabling usage of an offset in the transformation made no significant differences to the achievable performance. For both the diagonal and the full transformation matrix the amount of adaptation data needed to increase the performance from the initial level increased to 5 and 8 seconds respectively. A likely effect of having more parameters to estimate.

Based on the tests carried out, where different initial model qualities were used as the amount of data was varied, it can be concluded that no direct relation was observed between the initial model performance and the obtainable effect of doing speaker adaptation.

The regression class division

The different types of regression class divisions were all based on divisions of the phonemes into broad phonetic classes. The results were very promising and showed that the finer a regression class division it is possible to apply, the better a performance is achievable. Phoneme recognition was improved relatively with 15.8% for the individual classes and with 12.8% when 10 broad phonetic classes were used. The tests also showed that the differences between phonemes within a regression class are not reduced significantly even though vowels and consonants are adapted in different regression classes. This was seen by the fact that only a small increase in the effect of doing speaker adaptation was observed, when vowels were separated from consonants compared to when no separation was done using the global regression class.

A comparison of individual transformation matrixes was carried out. The matrixes were compared in three ways; measuring distance, stress or correlation between any two matrixes. The analysis showed no clear tendency in similarity of matrixes over the tested speakers.

However it is believed that the approach of analysing the transformation matrixes to get an alternative data-driven regression class division might still prove useful, as these matrixes are indeed those that give the very high performance improvements when used with an individual regression class division.

The phonetic contents of the adaptation data

Several approaches were tried out in order to investigate, whether the phonetic contents of the adaptation material is of any importance for the obtainable effect of the speaker adaptation. The mean vectors in the speaker dependent systems trained from scratch were compared. It showed a high difference in inter speaker variances. Fricatives were shown to vary considerably less than the other phonemes. Especially vowels were seen to vary a lot from speaker to speaker.

Another approach used the `MLLR` program to study the importance of different phoneme groups in the estimation of transformation matrixes. The tests were carried out by ignoring a group of phonemes in the estimation of the transformation matrix, and observing the effect on the performance of the adapted models. However when concluding on the results obtained using this test procedure the influence of the used regression class division has to be taken into considerations. The results showed that there is indeed a difference between the impact of the different phonemes on the estimation of the transformation matrix. The performance of the adapted models dropped significantly when specifically one of the phonemes 'E', 'b', 'D', 'ltub' were ignored during the estimation of the transformation matrix. This shows that these phonemes are likely to have a high influence for the estimation of the transformation matrix. In similar tests, ignoring the phonemes 't', 'f' and 'm' resulted in an increase in the performance, indicating that these phonemes contribute with little usable information and possibly even affect the adaptation of other phonemes in their regression class badly. The results obtained in this work are very motivating for further studies in the importance of the phonetic contents of the adaptation material.

Inter speaker differences

All tests were carried out using ten speakers. As expected inter speaker differences are observed in achievable performance. For all speakers a positive effect of doing speaker adaptation with a full transformation matrix was obtained. Comparing initial performances with performances of adapted models showed that most of the speakers that performed badly on the initial models had a relatively high performance improvement.

The tests where phonemes were ignored in turn, when the transformation matrixes were estimated, showed very small inter speaker differences.

Summary

Very promising results were obtained when adapting a set of speaker independent models using the Maximum Likelihood Linear Regression technique. The method is very data efficient and needs in general only a few seconds of adaptation material before a positive effect of the adaptation is seen. A larger improvement in performance can be achieved if phonemes are divided into more regression classes. It can also be concluded that an increase in performance is obtainable if the phonetic contents of the adaptation material is chosen carefully. Studies done in the current work suggest that especially vowels and consonants like nasals and approximants contain more speaker specific information than other phoneme groups. Results obtained in the current work are indeed promising and encouraging for further studies in ways to improve data effectiveness of speaker adaptation techniques.

Part IV

Appendix

Appendix A
Theory of MLLR

In this appendix the theory behind the Maximum Likelihood Linear Regression (MLLR) method is derived. The speaker adaptation technique has been presented in [Leggetter, 1995]. The main object of this speaker adaptation technique is to transform the mean vectors from a set of hidden Markov models into a set of speaker dependent models. An amount of adaptation data from the new speaker is used to estimate a transformation matrix that is multiplied by the mean vectors of the initial models yielding the mean vectors of the adapted models.

Fundamental for the theory is the concept of regression classes. A regression class is a set of mixture components that share the same transformation matrix. The number of mixture components in a regression class can be varied. The two outer limit cases are, when all models are in the same regression class (global regression class), and when each model has its own individual regression class. The optimal choice of regression class division is application dependent and is not discussed any further here. The advantage of introducing regression classes is that they enable the estimation of a transformation for a model even though there are not any adaptation data available for the model.

The adaptation with the transformation matrix, $\boldsymbol{W}_s$ is carried out as

$$\hat{\boldsymbol{\mu}}_s = \boldsymbol{W}_s \cdot \boldsymbol{\xi}_s \tag{A.1}$$

where $\boldsymbol{W}_s$ is the $(n \times n+1)$ transformation matrix for a mixture component s, n is the number of features, $\hat{\boldsymbol{\mu}}_s$ is the adapted mean vector of the model and $\boldsymbol{\xi}_s$ is the extended mean vector

$$\boldsymbol{\xi}_s = [\omega, \mu_{s_1}, \ldots, \mu_{s_n}]' = [\omega : \boldsymbol{\mu}_s]' \tag{A.2}$$

where $\boldsymbol{\mu}_s$ is the original mean vector and ω is an offset term. Incorporating the offset term enables the possibility to model, when the mean vectors in the models of a new speakers differ from the initial models by an additive term. This could typically be a results of different recording conditions etc. The derivation of the estimation equations for $\boldsymbol{W}_s$ is done in steps. In the first case single-mixture Gaussian distributions are used, only a single sequence is observed and the transformation matrix is estimated using a single regression class for every state in every model. This is described in section A.1. In the following sections the theory is extended to also apply to the case of:

- Multiple regression classes. (Section A.2).
- Multiple mixture components. (Section A.3).
- Multiple observation sequences. (Section A.4).

The final section, A.5 derives the re-estimation equations for the case, when the transformation matrix is restricted to a diagonal matrix.

A.1 Derivation of the general estimation equations

A set of re-estimation equations are determined for the transformation matrix, $\boldsymbol{W}_s$ introduced in equation A.1. When single-mixture models are used, each state has only one mixture component, and s therefore refers to a state as well.

A maximum likelihood approach is applied to the problem. The estimated transformation matrix, $\hat{\boldsymbol{W}}_s$ is chosen as the transformation matrix which maximises the probability that the model adapted using $\hat{\boldsymbol{W}}_s$ has generated the adaptation data.

$$\hat{\boldsymbol{W}}_s = \max_{\boldsymbol{W}_s} P(\boldsymbol{O}|\hat{\boldsymbol{\lambda}}) \tag{A.3}$$

where $\boldsymbol{O} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_T\}$ is the set of observed feature vectors, and $\hat{\boldsymbol{\lambda}}$ defines the parameters of the adapted model.

The procedure of solving equation A.3 is similar to the one often applied, when re-estimation equations used in the Baum-Welch algorithm for training of hidden Markov models are found. The main idea is to define an auxiliary function

$$Q(\boldsymbol{\lambda}, \hat{\boldsymbol{\lambda}}) = \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} P(\boldsymbol{O}, \boldsymbol{\theta}|\boldsymbol{\lambda}) \cdot \log(P(\boldsymbol{O}, \boldsymbol{\theta}|\hat{\boldsymbol{\lambda}})) \tag{A.4}$$

where $\boldsymbol{\theta}$ is a state sequence and $\boldsymbol{\Theta}$ is the set of all possible state sequences with length T, which is the number of observed feature vectors. The auxiliary function depends on both the initial model $\boldsymbol{\lambda}$ and the adapted model $\hat{\boldsymbol{\lambda}}$. Using this definition it can be shown that by successively defining a new model $\hat{\boldsymbol{\lambda}}$ which maximises $Q(\cdot)$, the auxiliary function has the property that the value of $P(\boldsymbol{O}|\hat{\boldsymbol{\lambda}})$ will not decrease, which was the original objective. The auxiliary function is maximised in the standard way by differentiating and equating to zero. This procedure is described below.

The combined probability of the set of observed feature vectors $\boldsymbol{O}$ and a state sequence $\boldsymbol{\theta}$ given a model $\boldsymbol{\lambda}$ can be expressed in terms of the model parameters

$$P(\boldsymbol{O}, \boldsymbol{\theta}|\boldsymbol{\lambda}) = a_{\theta_T N} \cdot \prod_{t=1}^{T} a_{\theta_{t-1}\theta_t} b_{\theta_t}(\boldsymbol{o}_t) \tag{A.5}$$

where $a_{\theta_{t-1}\theta_t}$ is the transition probability of moving from state θ_{t-1} to state θ_t in a hidden Markov model with N states including the non-emitting entry and exit states, θ_0 and θ_N. $b_{\theta_t}(\boldsymbol{o}_t)$ is the probability density function of emitting the feature vector $\boldsymbol{o}_t$ when being in state θ_t. When using Gaussian distributions in the models, a probability density function for state s can be written:

$$b_s(\boldsymbol{o}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\boldsymbol{\Sigma}_s|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{o} - \boldsymbol{\mu}_s)' \boldsymbol{\Sigma}_s^{-1} (\boldsymbol{o} - \boldsymbol{\mu}_s)\right) \tag{A.6}$$

where $\boldsymbol{\Sigma}_s$ is the covariance matrix and $\boldsymbol{\mu}_s$ the corresponding mean vector for state s.

Similarly

$$\log(P(\boldsymbol{O}, \boldsymbol{\theta}|\hat{\boldsymbol{\lambda}})) = \log\left[\hat{a}_{\theta_t N} \prod_{t=1}^{T} \hat{a}_{\theta_{t-1}\theta_t} \hat{b}_{\theta_t}(\boldsymbol{o}_t)\right] \tag{A.7}$$

$$= \log \hat{a}_{\theta_t N} + \sum_{t=1}^{T} \log \hat{a}_{\theta_{t-1}\theta_t} + \sum_{t=1}^{T} \log \hat{b}_{\theta_t}(\boldsymbol{o}_t) \tag{A.8}$$

where, for an adapted model, the probability density function can be defined as:

$$\begin{aligned}
\hat{b}_s(\boldsymbol{o}) &= \frac{1}{(2\pi)^{\frac{n}{2}}|\boldsymbol{\Sigma}_s|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\boldsymbol{o}-\hat{\boldsymbol{\mu}}_s)'\boldsymbol{\Sigma}_s^{-1}(\boldsymbol{o}-\hat{\boldsymbol{\mu}}_s)\right) && \text{(A.9)}\\
&= \frac{1}{(2\pi)^{\frac{n}{2}}|\boldsymbol{\Sigma}_s|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\boldsymbol{o}-\boldsymbol{W}_s\boldsymbol{\mu}_s)'\boldsymbol{\Sigma}_s^{-1}(\boldsymbol{o}-\boldsymbol{W}_s\boldsymbol{\mu}_s)\right) && \text{(A.10)}
\end{aligned}$$

Substituting equation A.7 in the auxiliary function in equation A.4 yields:

$$\begin{aligned}
Q(\boldsymbol{\lambda},\hat{\boldsymbol{\lambda}}) &= \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} P(\boldsymbol{O},\boldsymbol{\theta}|\boldsymbol{\lambda})\left[\log \hat{a}_{\theta_t N} + \sum_{t=1}^{T}\log \hat{a}_{\theta_{t-1}\theta_t} + \sum_{t=1}^{T}\log \hat{b}_{\theta_t}(\boldsymbol{o}_t)\right] && \text{(A.11)}\\
&= \sum_{t=1}^{N} Q_{a_i}[\boldsymbol{\lambda},\{\hat{a}_{is}\}_{s=1}^{N}] + \sum_{s=1}^{N} Q_b(\boldsymbol{\lambda},\hat{b}_s) && \text{(A.12)}
\end{aligned}$$

where T is the number of observed feature vectors, N is the number of states in the models, and $Q_{a_i}[\boldsymbol{\lambda},\{\hat{a}_{is}\}_{s=1}^{N}]$ and $Q_b(\boldsymbol{\lambda},\hat{b}_s)$ are defined as

$$\begin{aligned}
Q_{a_i}[\boldsymbol{\lambda},\{\hat{a}_{is}\}_{s=1}^{N}] &= \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} P(\boldsymbol{O},\theta_T = i|\boldsymbol{\lambda})\log \hat{a}_{iN} && \text{(A.13)}\\
&\quad + \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\sum_{t=1}^{T}\sum_{s=1}^{N} P(\boldsymbol{O},\theta_{t-1}=i,\theta_t = s|\boldsymbol{\lambda})\log \hat{a}_{is} && \text{(A.14)}
\end{aligned}$$

and

$$Q_b(\boldsymbol{\lambda},\hat{b}_s) = \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\sum_{t=1}^{T} P(\boldsymbol{O},\theta_t = s|\boldsymbol{\lambda})\log \hat{b}_s(\boldsymbol{o}_t) \qquad \text{(A.15)}$$

The object is to maximise the auxiliary function, $Q(\boldsymbol{\lambda},\hat{\boldsymbol{\lambda}})$ with regard to the transformation matrix, $\hat{\boldsymbol{W}}_s$. However only the part termed $Q_b(\boldsymbol{\lambda},\hat{b}_s)$ depends on $\hat{\boldsymbol{W}}_s$ through the probability density function, $\hat{b}_s(\boldsymbol{o})$. Therefore only this part will be maximised in the following, as the remaining part of the auxiliary function can be viewed as a constant.

The total *a posteriori* probability of occupying state s at time t, given that the set of feature vectors, $\boldsymbol{O}$ is observed, is defined as

$$\gamma_s(t) = \frac{1}{P(\boldsymbol{O}|\boldsymbol{\lambda})}\sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} P(\boldsymbol{O},\theta_t = s|\boldsymbol{\lambda}) \qquad \text{(A.16)}$$

Using this variable in equation A.15 yields

$$Q_b(\boldsymbol{\lambda},\hat{b}_s) = P(\boldsymbol{O}|\boldsymbol{\lambda})\sum_{s=1}^{S}\sum_{t=1}^{T}\gamma_s(t)\log \hat{b}_s(\boldsymbol{o}_t) \qquad \text{(A.17)}$$

where S is the number of states in the system. Inserting the expression for the density distribution function from equation A.10 gives

$$\begin{aligned}
\log \hat{b}_s(\boldsymbol{o}_t) &= -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}_s| - \frac{1}{2}(\boldsymbol{o}_t - \hat{\boldsymbol{W}}_s\boldsymbol{\xi}_s)'\boldsymbol{\Sigma}_s^{-1}(\boldsymbol{o}_t - \hat{\boldsymbol{W}}_s\boldsymbol{\xi}_s) && \text{(A.18)}\\
&\Downarrow\\
Q_b(\boldsymbol{\lambda}|\hat{b}_s) &= -\frac{1}{2}P(\boldsymbol{O}|\boldsymbol{\lambda})\sum_{s=1}^{S}\sum_{t=1}^{T}\gamma_s(t)\\
&\quad \times\left[n\cdot\log(2\pi) + \log|\boldsymbol{\Sigma}_s| + (\boldsymbol{o}_t - \hat{\boldsymbol{W}}_s\boldsymbol{\xi}_s)'\boldsymbol{\Sigma}_s^{-1}(\boldsymbol{o}_t - \hat{\boldsymbol{W}}_s\boldsymbol{\xi}_s)\right] && \text{(A.19)}
\end{aligned}$$

Using a standard maximisation method this expression is then differentiated with respect to $\hat{\boldsymbol{W}}_s$, the estimated transformation matrix, and equated to zero.

$$\frac{d}{d\hat{\boldsymbol{W}}_s}Q(\boldsymbol{\lambda},\hat{\boldsymbol{\lambda}}) = \frac{d}{d\hat{\boldsymbol{W}}_s}Q_b(\boldsymbol{\lambda},\hat{b}_s) \tag{A.20}$$

$$= -\frac{1}{2}P(\boldsymbol{O}|\boldsymbol{\lambda})\frac{d}{d\hat{\boldsymbol{W}}_s}\sum_{s=1}^{S}\sum_{t=1}^{T}\gamma_s(t)\left[n\cdot\log(2\pi)+\log|\boldsymbol{\Sigma}_s|+h(\boldsymbol{o}_t,s)\right] \tag{A.21}$$

where

$$h(\boldsymbol{o}_t,s) = (\boldsymbol{o}_t - \hat{\boldsymbol{W}}_s\boldsymbol{\xi}_s)'\boldsymbol{\Sigma}_s^{-1}(\boldsymbol{o}_t - \hat{\boldsymbol{W}}_s\boldsymbol{\xi}_s) \tag{A.22}$$

Now

$$\frac{d}{d\hat{\boldsymbol{W}}_s}Q(\boldsymbol{\lambda},\hat{\boldsymbol{\lambda}}) = 0 \tag{A.23}$$

$$\Updownarrow$$

$$-\frac{1}{2}P(\boldsymbol{O}|\boldsymbol{\lambda})\frac{d}{d\hat{\boldsymbol{W}}_s}\sum_{s=1}^{S}\sum_{t=1}^{T}\gamma_s(t)h(\boldsymbol{o}_t,s) = 0 \tag{A.24}$$

$$\Updownarrow$$

$$\sum_{t=1}^{T}\gamma_s(t)\boldsymbol{\Sigma}_s^{-1}(\boldsymbol{o}_t - \hat{\boldsymbol{W}}_s\boldsymbol{\xi}_s)\boldsymbol{\xi}_s' = 0 \tag{A.25}$$

$$\Updownarrow$$

$$\sum_{t=1}^{T}\gamma_s(t)\boldsymbol{\Sigma}_s^{-1}\boldsymbol{o}_t\boldsymbol{\xi}_s' = \sum_{t=1}^{T}\gamma_s(t)\boldsymbol{\Sigma}_s^{-1}\hat{\boldsymbol{W}}_s\boldsymbol{\xi}_s\boldsymbol{\xi}_s' \tag{A.26}$$

Equation A.26 is the general equation for estimating the transformation matrix, $\hat{\boldsymbol{W}}_s$. So far no assumptions have been made about the nature of the covariance matrix, $\boldsymbol{\Sigma}_s$, but in the remaining part of the appendix, only the case of having a diagonal covariance matrix is dealt with. There is no closed form solution for the full covariance case [Leggetter, 1995].

Equation A.26 is extended to use multiple regression classes and solved for the case of a diagonal covariance matrix in section A.2.

A.2 Extension to multiple regression classes

When there is only a small amount of adaptation data available, one solution is to tie several states in the system together in a regression class. All the data corresponding to states within regression class is then used to estimate a shared transformation matrix, $\hat{\boldsymbol{W}}_s$ for the regression class. If R states, $\{s_1, s_2, \ldots, s_R\}$, are shared in a given regression class, the general equation A.26 can be written:

$$\sum_{t=1}^{T}\sum_{r=1}^{R}\gamma_{s_r}(t)\boldsymbol{\Sigma}_{s_r}^{-1}\boldsymbol{o}_t\boldsymbol{\xi}_{s_r}' = \sum_{t=1}^{T}\sum_{r=1}^{R}\gamma_{s_r}(t)\boldsymbol{\Sigma}_{s_r}^{-1}\hat{\boldsymbol{W}}_s\boldsymbol{\xi}_{s_r}\boldsymbol{\xi}_{s_r}' \tag{A.27}$$

$$\sum_{t=1}^{T}\sum_{r=1}^{R}\gamma_{s_r}(t)\boldsymbol{\Sigma}_{s_r}^{-1}\boldsymbol{o}_t\boldsymbol{\xi}_{s_r}' = \sum_{r=1}^{R}\boldsymbol{V}^{(r)}\hat{\boldsymbol{W}}_s\boldsymbol{D}^{(r)} \tag{A.28}$$

where $\boldsymbol{V}^{(r)}$ and $\boldsymbol{D}^{(r)}$ are two matrixes defined as

$$\boldsymbol{V}^{(r)} = \sum_{t=1}^{T} \gamma_{s_r}(t) \boldsymbol{\Sigma}_{s_r}^{-1} \tag{A.29}$$

and

$$\boldsymbol{D}^{(r)} = \boldsymbol{\xi}_{s_r} \boldsymbol{\xi}'_{s_r} \tag{A.30}$$

A matrix, $\boldsymbol{Y}$ is now defined as the right-hand side of equation A.28. Denoting the elements of the matrixes $\boldsymbol{Y}, \boldsymbol{V}^{(r)} \boldsymbol{W}_s$ and $\boldsymbol{D}^{(r)}$ by $y_{ij}, v_{ij}^{(r)}, w_{ij}$ and $d_{ij}^{(r)}$ respectively yields

$$\left[\begin{array}{c} y_{ij} \end{array}\right]_{(n \times n+1)} = \sum_{r=1}^{R} \left[\begin{array}{c} v_{ij}^{(r)} \end{array}\right]_{(n \times n)} \left[\begin{array}{c} w_{ij} \end{array}\right]_{(n \times n+1)} \left[\begin{array}{c} d_{ij}^{(r)} \end{array}\right]_{(n+1 \times n+1)}$$

Written on an element basis and with terms re-arranged

$$y_{ij} = \sum_{p=1}^{n} \sum_{q=1}^{n+1} w_{pq} \left[\sum_{r=1}^{R} v_{ip}^{(r)} d_{qj}^{(r)} \right] \tag{A.31}$$

Where the last summation term has a simplified solution for two reasons. The covariance matrix is diagonal which makes $\boldsymbol{V}^{(r)}$ diagonal, and $\boldsymbol{D}^{(r)}$ is symmetric because it is the outer product of two equal vectors. Therefore

$$\sum_{r=1}^{R} v_{ip}^{(r)} d_{qj}^{(r)} = \left\{ \begin{array}{lll} \sum_{r=1}^{R} v_{ii}^{(r)} d_{jq}^{(r)} & \text{when} & i = p \\ 0 & \text{when} & i \neq p \end{array} \right. \tag{A.32}$$

Combining with equation A.31 and introducing a new matrix variable, $\boldsymbol{G}^{(i)}$ with elements $g_{jq}^{(i)}$ results in

$$\begin{aligned} y_{ij} &= \sum_{q=1}^{n+1} w_{iq} \sum_{r=1}^{R} v_{ii}^{(r)} d_{jq}^{(r)} && \text{(A.33)} \\ &= \sum_{q=1}^{n+1} w_{iq} g_{jq}^{(i)} && \text{(A.34)} \end{aligned}$$

Where

$$\boldsymbol{G}^{(i)} = \left[\begin{array}{c} g_{jq}^{(i)} \end{array}\right] = \left[\begin{array}{cccc} \sum_{r=1}^{R} v_{ii}^{(r)} d_{11}^{(r)} & \sum_{r=1}^{R} v_{ii}^{(r)} d_{12}^{(r)} & \cdots & \sum_{r=1}^{R} v_{ii}^{(r)} d_{1\,n+1}^{(r)} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{r=1}^{R} v_{ii}^{(r)} d_{n+1\,1}^{(r)} & \sum_{r=1}^{R} v_{ii}^{(r)} d_{n+1\,2}^{(r)} & \cdots & \sum_{r=1}^{R} v_{ii}^{(r)} d_{n+1\,n+1}^{(r)} \end{array}\right] \tag{A.35}$$

Equation A.34 corresponds to the right-hand side of equation A.27. Denoting the left-hand side by the matrix, $\boldsymbol{Z}$ with elements z_{ij} yields two equations with one unknown variable. Written on element basis

$$z_{ij} = y_{ij} = \sum_{q=1}^{n+1} w_{iq} g_{jq}^{(i)} \tag{A.36}$$

The $\boldsymbol{G}$ matrix is a three dimensional structure $(n+1 \times n+1 \times n)$, however solving equation A.36 on a row basis can be done as following:

$$\begin{aligned} \boldsymbol{z}_i &= \hat{\boldsymbol{w}}_i \, \boldsymbol{G}^{(i)} && \text{(A.37)} \\ \hat{\boldsymbol{w}}_i' &= \boldsymbol{G}^{(i)-1} \, \boldsymbol{z}_i' && \text{(A.38)} \\ \Big[\ \hat{w}_i\ \Big]_{(n+1\times 1)} &= \Big[\ g_{jq}\ \Big]_{(n+1\times n+1)} \Big[\ z_i\ \Big]_{(n+1\times 1)} \end{aligned}$$

Using equation A.38, the transformation matrix for a given regression class, $\hat{\boldsymbol{W}}_s$ can be calculated row by row. The following two sections lead to a derivation of a corresponding equation for the case of multiple mixture components and multiple observation sequences.

A.3 Extension to multiple mixture components

The symbol probability distributions in a hidden Markov model are modeled as a mixture of a finite number of probability density functions

$$b_s(\boldsymbol{o}) = \sum_{k=1}^{M} c_{sk} b_{sk}(\boldsymbol{o}) \tag{A.39}$$

where M is the number of mixtures and c_{sk} is the mixture coefficient for the kth mixture in state s. Similar to equation A.5 for the single mixture case, the probability of a set of observed feature vectors, $\boldsymbol{O}$ and a state sequence, $\boldsymbol{\theta}$ given a model $\boldsymbol{\lambda}$ is defined

$$\begin{aligned} P(\boldsymbol{O},\boldsymbol{\theta}|\boldsymbol{\lambda}) &= a_{\theta_T N} \prod_{t=1}^{T} a_{\theta_{t-1}\theta_t} \left[\sum_{k=1}^{M} c_{\theta_t k} b_{\theta_t k}(\boldsymbol{o}_t)\right] && \text{(A.40)} \\ &= \sum_{k_1=1}^{M}\sum_{k_2=1}^{M}\cdots\sum_{k_T=1}^{M}\left[a_{\theta_T N}\prod_{t=1}^{T} a_{\theta_{t-1}\theta_t} c_{\theta_t k_t} b_{\theta_t k_t}(\boldsymbol{o}_t)\right] && \text{(A.41)} \\ &= \sum_{k_1=1}^{M}\sum_{k_2=1}^{M}\cdots\sum_{k_T=1}^{M} P(\boldsymbol{O},\boldsymbol{\theta},K|\boldsymbol{\lambda}) && \text{(A.42)} \end{aligned}$$

where K denotes a given sequence of mixture components, $K = (k_1, k_2, \ldots, k_T)$. Letting Ω_b be the set of all possible mixture component sequences, the auxiliary function is given as

$$\begin{aligned} Q(\boldsymbol{\lambda},\hat{\boldsymbol{\lambda}}) &= \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\sum_{K\in\Omega_b} P(\boldsymbol{O},\boldsymbol{\theta},K|\boldsymbol{\lambda}) \cdot \log(P(\boldsymbol{O},\boldsymbol{\theta},K|\hat{\boldsymbol{\lambda}})) && \text{(A.43)} \\ &= \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\sum_{K\in\Omega_b} P(\boldsymbol{O},\boldsymbol{\theta},K|\boldsymbol{\lambda}) \\ &\quad \times \left[\log \hat{a}_{\theta_T N} + \sum_{t=1}^{T}\log \hat{a}_{\theta_{t-1}\theta_t} + \sum_{t=1}^{T}\log \hat{b}_{\theta_t k_t}(\boldsymbol{o}_t) + \sum_{t=1}^{T}\log \hat{c}_{\theta_t k_t}\right] && \text{(A.44)} \end{aligned}$$

Again, since only the term with $\hat{b}_{\theta_t k_t}(\boldsymbol{o}_t)$ depends on $\hat{\boldsymbol{W}}_s$ the rest of the terms in the auxiliary function can be viewed as a constant

$$Q(\boldsymbol{\lambda},\hat{\boldsymbol{\lambda}}) = \text{constant} + \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\sum_{K\in\Omega_b}\sum_{t=1}^{T} P(\boldsymbol{O},\theta_t = j, k_t = k|\boldsymbol{\lambda}) \log \hat{b}_{\theta_t k_t}(\boldsymbol{o}_t) \tag{A.45}$$

The total occupation probability of mixture component k of state s at time t given that the set of feature vectors, $\boldsymbol{O}$ is observed is given as

$$\gamma_{sk}(t) \quad = \quad \frac{1}{P(\boldsymbol{O}|\boldsymbol{\lambda})} \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sum_{K\in\Omega_b} P(\boldsymbol{O}, \theta_t = s, k_t = k|\boldsymbol{\lambda}) \tag{A.46}$$

Substituting into equation A.45 yields

$$Q(\boldsymbol{\lambda}, \hat{\boldsymbol{\lambda}}) \quad = \quad \text{constant} + P(\boldsymbol{O}|\boldsymbol{\lambda}) \sum_{t=1}^{T} \gamma_{sk}(t) \log \hat{b}_{\theta_t k_t}(\boldsymbol{o}_t) \tag{A.47}$$

As for the single mixture case the procedure is now to differentiate the auxiliary function with respect to $\hat{\boldsymbol{W}}_s$ and equate to zero in order to find the maximum likelihood solution. This leads to the following equation for the multiple mixture and multiple regression class case:

$$\sum_{t=1}^{T} \sum_{r=1}^{R} \gamma_{s_r k}(t) \boldsymbol{\Sigma}_{s_r k}^{-1} \boldsymbol{o}_t \boldsymbol{\xi}'_{s_r k} \quad = \quad \sum_{t=1}^{T} \sum_{r=1}^{R} \gamma_{s_r k}(t) \boldsymbol{\Sigma}_{s_r k}^{-1} \hat{\boldsymbol{W}}_s \boldsymbol{\xi}_{s_r k} \boldsymbol{\xi}'_{s_r k} \tag{A.48}$$

where a regression class is a set of R mixture components; $\{s_1 k, s_2 k, \ldots, s_R k\}$.

Equation A.48 is now extended to the case of having multiple observation sequences in the adaptation data.

A.4 Extension to multiple observation sequences

In the most general version of the MLLR technique, the case of multiple observation sequences is included. Assume that there are P observation sequences in the adaptation data, $\boldsymbol{O}^{(1)} \ldots \boldsymbol{O}^{(P)}$, each consisting of an individual number of frames; the observation sequence $\boldsymbol{O}^{(p)}$ has T_p number of frames ($\boldsymbol{O}^{(p)} = \boldsymbol{o}_1^{(p)} \ldots \boldsymbol{o}_{T_p}^{(p)}$). Equation A.48 then becomes

$$\sum_{p=1}^{P} \sum_{t=1}^{T_p} \sum_{r=1}^{R} \gamma_{s_r k}^{(p)}(t) \boldsymbol{\Sigma}_{s_r k}^{-1} \boldsymbol{o}_t^{(p)} \boldsymbol{\xi}'_{s_r k} \quad = \quad \sum_{p=1}^{P} \sum_{t=1}^{T_p} \sum_{r=1}^{R} \gamma_{s_r k}^{(p)}(t) \boldsymbol{\Sigma}_{s_r k}^{-1} \hat{\boldsymbol{W}}_s \boldsymbol{\xi}_{s_r k} \boldsymbol{\xi}'_{s_r k} \tag{A.49}$$

This equation is then solved following the procedure used in section A.2, where new matrixes are defined and the transformation matrix finally is calculated on a per row basis. The derivations are given below.

Introducing the matrixes $\boldsymbol{V}^{(r)}$ and $\boldsymbol{D}^{(r)}$ defined as

$$\boldsymbol{V}^{(r)} = \sum_{p=1}^{P} \sum_{t=1}^{T_p} \gamma_{s_r k}^{(p)}(t) \boldsymbol{\Sigma}_{s_r k}^{-1} \tag{A.50}$$

and

$$\boldsymbol{D}^{(r)} = \boldsymbol{\xi}_{s_r k} \boldsymbol{\xi}'_{s_r k} \tag{A.51}$$

brings equation A.49 on the form

$$\sum_{p=1}^{P} \sum_{t=1}^{T_p} \sum_{r=1}^{R} \gamma_{s_r k}^{(p)}(t) \boldsymbol{\Sigma}_{s_r k}^{-1} \boldsymbol{o}_t^{(p)} \boldsymbol{\xi}'_{s_r k} \quad = \quad \sum_{r=1}^{R} \boldsymbol{V}^{(r)} \hat{\boldsymbol{W}}_s \boldsymbol{D}^{(r)} \tag{A.52}$$

A matrix, $\boldsymbol{Y}$ is defined as the right-hand side of equation A.52 and the elements can be calculated as

$$y_{ij} = \sum_{p=1}^{n}\sum_{q=1}^{n+1} w_{pq}\left[\sum_{r=1}^{R} v_{ip}^{(r)} d_{qj}^{(r)}\right] \tag{A.53}$$

$$y_{ij} = \sum_{q=1}^{n+1} w_{iq}\sum_{r=1}^{R} v_{ii}^{(r)} d_{jq}^{(r)} \tag{A.54}$$

where the properties of the $\boldsymbol{V}^{(r)}$ being diagonal and $\boldsymbol{D}^{(r)}$ being symmetric has been utilised according to equation A.32. Denoting the left-hand side of equation A.49 with the matrix $\boldsymbol{Z}$ with elements z_{ij} and using the matrix $\boldsymbol{G}^{(i)}$ with elements $g_{jq}^{(i)}$ yields

$$z_{ij} = y_{ij} = \sum_{q=1}^{n+1} w_{iq} g_{jq}^{(i)} \tag{A.55}$$

and solving for the transformation matrix, $\hat{\boldsymbol{W}}_s$

$$\hat{\boldsymbol{w}}_i' = \boldsymbol{G}^{(i)-1}\, \boldsymbol{z}_i' \tag{A.56}$$

A.5 Equations for a diagonal transformation matrix

The computational overhead involved in re-estimating the transformation matrix using the above equations is large mainly due to the high number of matrix inversions that has to be carried out. This can be greatly reduced by restricting the transformation matrix, $\boldsymbol{W}_s$ to a diagonal matrix

$$\boldsymbol{W}_s = \begin{bmatrix} w_{1,1} & w_{1,2} & 0 & \cdots & 0 \\ w_{2,1} & 0 & w_{2,3} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ w_{n,1} & 0 & \cdots & 0 & w_{n,n+1} \end{bmatrix} \tag{A.57}$$

For implementational reasons the non-zero elements of this matrix is rewritten to a transformation vector, $\mathbf{w}_s$

$$\mathbf{w}_s = \begin{bmatrix} w_{1,1} \\ \vdots \\ w_{n,1} \\ w_{1,2} \\ \vdots \\ w_{n,n+1} \end{bmatrix} \tag{A.58}$$

Equations for re-estimating this transformation vector is found following the same procedure as applied in the case of a full transformation matrix.

Defining the matrix $\boldsymbol{D}$ as

$$\boldsymbol{D}_{s_r k} = \begin{bmatrix} \omega & 0 & \cdots & 0 & \mu_{s_r k1} & 0 & \cdots & 0 \\ 0 & \omega & \ddots & \vdots & 0 & \mu_{s_r k2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \omega & 0 & \cdots & 0 & \mu_{s_r kn} \end{bmatrix} \tag{A.59}$$

it can be shown that the non-constant term of the auxiliary function for the case of a single regression class, a single mixture and a single observation sequence, can be written [Leggetter, 1995]

$$Q_b(\boldsymbol{\lambda}|\hat{b}_s) = -\frac{1}{2}P(\boldsymbol{O}|\boldsymbol{\lambda})\sum_{s=1}^{S}\sum_{t=1}^{T}\gamma_s(t) \tag{A.60}$$

$$\times\left[n\cdot\log(2\pi)+\log|\boldsymbol{\Sigma}_s|+(\boldsymbol{o}_t-\boldsymbol{D}_s\hat{\mathbf{w}}_s)'\boldsymbol{\Sigma}_s^{-1}(\boldsymbol{o}_t-\boldsymbol{D}_s\hat{\mathbf{w}}_s)\right] \tag{A.61}$$

Differentiating with respect to the transformation vector gives

$$\frac{d}{d\hat{\mathbf{w}}_s}Q(\boldsymbol{\lambda},\hat{\boldsymbol{\lambda}}) = \frac{d}{d\hat{\mathbf{w}}_s}Q_b(\boldsymbol{\lambda},\hat{b}_s) \tag{A.62}$$

$$= -\frac{1}{2}P(\boldsymbol{O}|\boldsymbol{\lambda})\frac{d}{d\hat{\mathbf{w}}_s}\sum_{s=1}^{S}\sum_{t=1}^{T}\gamma_s(t)\left[n\cdot\log(2\pi)+\log|\boldsymbol{\Sigma}_s|+h(\boldsymbol{o}_t,s)\right] \tag{A.63}$$

where

$$h(\boldsymbol{o}_t,s)=(\boldsymbol{o}_t-\boldsymbol{D}_s\hat{\boldsymbol{w}}_s)'\boldsymbol{\Sigma}_s^{-1}(\boldsymbol{o}_t-\boldsymbol{D}_s\hat{\boldsymbol{w}}_s) \tag{A.64}$$

Now

$$\frac{d}{d\hat{\mathbf{w}}_s}Q(\boldsymbol{\lambda},\hat{\boldsymbol{\lambda}}) = 0 \tag{A.65}$$

$$\Updownarrow$$

$$\sum_{t=1}^{T}\gamma_s(t)\boldsymbol{D}_s'\boldsymbol{\Sigma}_s^{-1}(\boldsymbol{o}_t-\boldsymbol{D}_s\hat{\boldsymbol{w}}_s) = 0 \tag{A.66}$$

$$\Updownarrow$$

$$\sum_{t=1}^{T}\gamma_s(t)\boldsymbol{D}_s'\boldsymbol{\Sigma}_s^{-1}\boldsymbol{o}_t = \sum_{t=1}^{T}\gamma_s(t)\boldsymbol{D}_s'\boldsymbol{\Sigma}_s^{-1}\boldsymbol{D}_s\hat{\mathbf{w}}_s \tag{A.67}$$

Extending equation A.67 to the case of multiple regression classes, multiple mixture components and multiple observation sequences give the following equation which corresponds to equation A.49 for the case of a full transformation matrix

$$\sum_{p=1}^{P}\sum_{t=1}^{T_p}\sum_{r=1}^{R}\gamma_{s_rk}(t)\boldsymbol{D}_{s_rk}'\boldsymbol{\Sigma}_{srk}^{-1}\boldsymbol{o}_t = \sum_{p=1}^{P}\sum_{t=1}^{T_p}\sum_{r=1}^{R}\gamma_{srk}(t)\boldsymbol{D}_{srk}'\boldsymbol{\Sigma}_{srk}^{-1}\boldsymbol{D}_{srk}\hat{\mathbf{w}}_{srk} \tag{A.68}$$

The transformation vector, $\hat{\mathbf{w}}_s$ can now be found as

$$\hat{\mathbf{w}}_s=\left[\sum_{p=1}^{P}\sum_{t=1}^{T_p}\sum_{r=1}^{R}\gamma_{srk}(t)\boldsymbol{D}_{srk}'\boldsymbol{\Sigma}_{srk}^{-1}\boldsymbol{D}_{srk}\right]^{-1}\left[\sum_{p=1}^{P}\sum_{t=1}^{T_p}\sum_{r=1}^{R}\gamma_{s_rk}(t)\boldsymbol{D}_{s_rk}'\boldsymbol{\Sigma}_{srk}^{-1}\boldsymbol{o}_t\right] \tag{A.69}$$

Equation A.56 and A.69 are both implemented in `MLLR`. The mathematical algorithms are given in section 5.4

Appendix B
Forward-Backward algorithm

The mixture component occupancy probability, $\gamma_{sk}^{(p)}(t)$ can be calculated using the Forward-Backward algorithm. $\gamma_{sk}^{(p)}(t)$ is defined as the probability of being in mixture component k of state s given the observed set of feature vectors $\boldsymbol{O}$ and the current model parameters $\boldsymbol{\lambda}$:

$$\gamma_{sk}(t) = \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sum_{K\in\Omega_b} P(\theta_t = s, k_t = k|\boldsymbol{O}, \boldsymbol{\lambda}) \tag{B.1}$$

$$= \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sum_{K\in\Omega_b} \frac{P(\boldsymbol{O}, \theta_t = s, k_t = k|\boldsymbol{\lambda})}{P(\boldsymbol{O}|\boldsymbol{\lambda})} \tag{B.2}$$

where $\boldsymbol{\theta}$ is a state sequence and $\boldsymbol{\Theta}$ denotes the set of all possible state sequences of the right length. Similarly K denotes a given sequence of mixture components and Ω_b is the set of all possible mixture component sequences. $\gamma_{sk}^{(p)}(t)$ can be calculated using two variables called the forward variable (α) and the backward variable (β). They are further described in sections B.1 and B.2 respectively. In section B.3 it is shown how these two variables combine into the mixture component occupancy probability.

B.1 Forward procedure

The algorithm for the forward procedure is given in figure B.1. The fundamental variable is the forward variable $\alpha_t^{(p)}(i)$ defined as:

$$\alpha_i^{(p)}(t) = P(\mathbf{o}_1^{(p)}\mathbf{o}_2^{(p)}\cdots\mathbf{o}_t^{(p)}, \theta_t = i|\boldsymbol{\lambda}) \tag{B.3}$$

$\alpha_t^{(p)}(i)$ represents the probability of having observed the pth set of feature vectors up until the time t and being in state i at time t for a given model $\boldsymbol{\lambda}$.

The overall probability of having observed the set of feature vectors with the given model is calculated as the forward variable for the last frame, T as stated in equation B.13.

B.2 Backward procedure

Analogous to the forward algorithm the backward algorithm has been developed. Its primary variable is defined as:

$$\beta_i^{(p)}(t) = P(\mathbf{o}_{t+1}^{(p)}\mathbf{o}_{t+2}^{(p)}\cdots\mathbf{o}_{T_p}^{(p)}|\theta_t = i, \lambda) \tag{B.4}$$

determining the probability of observing the observation sequence p from $t+1$ to T_p given that the model is in state i at time t. The complete algorithm is described in steps in figure B.2.

B.3 Calculation of mixture component occupancy probability

The α and β variables can be combined to give the probability of being in state i and mixture component k at time t and having observed the sequence $\boldsymbol{O}^{(p)}$ given the model.

$$P(\boldsymbol{O}^{(p)}, \theta_t = i, k_t = k|\boldsymbol{\lambda}) = U_i^{(p)} \; c_{ik} \; b_{ik}(\mathbf{o}_t^{(p)}) \; \beta_i^{(p)}(t) \tag{B.5}$$

where

$$U_i^p = \begin{cases} a_{1i} & \text{if } \; t=1 \\ \sum_{j=2}^{N-1} \alpha_j^{(p)}(t-1)a_{ji} & \text{otherwise} \end{cases} \tag{B.6}$$

where N is the number of states. The forward variable $\alpha_i^{(p)}(t-1)$ accounts for the partial observation sequence $\mathbf{o}_1^{(p)}, \mathbf{o}_2^{(p)}, \ldots, \mathbf{o}_{t-1}^{(p)}$ and being in state i, a_{ji} is the transition probability from state j to state i and $b_{ik}(\mathbf{o}_t^{(p)})$ the probability of emitting the observation $\mathbf{o}_t^{(p)}$ from mixture component k of state i. Finally $\beta_i^{(p)}(t)$ gives the probability of observing the remaining feature vectors $\mathbf{o}_{t+1}^{(p)}, \mathbf{o}_{t+2}^{(p)}, \ldots, \mathbf{o}_T^{(p)}$ given that the state is i.

By summing the probability in equation B.5 over all states and mixture components, an expression for the mixture component occupancy probability is given as

$$\gamma_{sk}(t) = \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sum_{K\in\Omega_b} \frac{P(\boldsymbol{O}, \theta_t = s, k_t = k|\boldsymbol{\lambda})}{P(\boldsymbol{O}|\boldsymbol{\lambda})} \tag{B.7}$$

$$= \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sum_{K\in\Omega_b} \frac{U_s^{(p)} \; c_{sk} b_{sk}(\mathbf{o}_t^{(p)}) \; \beta_s^{(p)}(t)}{\alpha_N^{(p)}(T_p)} \tag{B.8}$$

1. Initialisation

$$\alpha_1^{(p)}(1) = 1 \tag{B.9}$$

$$\alpha_i^{(p)}(1) = a_{1i}\ b_i(\boldsymbol{o}_1^{(p)}) \qquad \text{for} \quad 1 < i < N \tag{B.10}$$

2. Induction

$$\alpha_i^{(p)}(t) = \left[\sum_{j=2}^{N-1} \alpha_j^{(p)}(t-1) a_{ji}\right] b_i(\boldsymbol{o}_t^{(p)}) \qquad \text{for} \quad 1 < i < N \ ; \ 1 < t \leq T_p \tag{B.11}$$

3. Termination

$$\alpha_N^{(p)}(T_p) = \sum_{j=2}^{N-1} \alpha_j^{(p)}(T_p) a_{jN} \tag{B.12}$$

$$P(\boldsymbol{O}^{(p)}|\boldsymbol{\lambda}) = \alpha_N^{(p)}(T_p) \tag{B.13}$$

Figure B.1: *Forward procedure.*

1. Initialisation

$$\beta_i^{(p)}(T_p) = a_{iN} \qquad \text{for} \quad 1 < i < N \tag{B.14}$$

2. Induction

$$\beta_i^{(p)}(t) = \sum_{j=2}^{N-1} a_{ij}\ b_j(\boldsymbol{o}_{t+1}^{(p)})\ \beta_j^{(p)}(t+1) \qquad \text{for} \quad 1 < i < N \ ; \ T_p > t \geq 1 \tag{B.15}$$

3. Termination

$$\beta_i^{(p)}(1) = \sum_{j=2}^{N-1} a_{1j}\ b_j(\boldsymbol{o}_1^{(p)})\ \beta_j^{(p)}(1) \tag{B.16}$$

$$P(\boldsymbol{O}^{(p)}|\boldsymbol{\lambda}) = \beta_1^{(p)}(1) \tag{B.17}$$

Figure B.2: *Backward procedure.*

Appendix C
The EUROM.1 database

The Danish EUROM.1 Database is recorded at Institute of Electronic Systems at Aalborg University and constitutes the Danish part of the SAM EUROM.1 database. The recordings took place during spring 1990 and spring 1991. The database consists of three separate parts: A Many Talker Corpus, a Few Talker Corpus and a Very Few Talker Corpus.

In total 25 female and 35 male speakers participated in the recordings. In age they are distributed between 16 and up to above 56 at the recording time. This appendix gives a short introduction to the database. Further details are given in [Lindberg and Christensen, 1995].

C.1 Data material

All recordings for the database took place in an acoustically damped and soundproof room and all recordings were text prompted. Three types of data were recorded: Sentences and passages, numbers and CVC words. Each passages consisted of 5 sentences which were task related. The additional sentences were recorded in order to compensate for the uneven phoneme distribution in the passages. The division of speakers and data types for the different parts of the database is given below.

Many Talker Corpus 60 speakers.

- 4 passages each containing 5 sentences.
- 5 compensating sentences.
- 100 numbers.

Few Talker Corpus 10 speakers.

- 15 passages each containing 5 sentences.
- 25 compensating sentences.
- 100 numbers (each spoken 5 times)
- 114 CVC words (each spoken 5 times)

Very Few Talker Corpus 2 speakers

- 114 CVC words embedded in 5 different carrier phrases.
- 10 carrier words (each spoken 5 times)

C.2 Transcriptions

EUROM.1 is transcribed at phoneme level. Table C.1 lists the used phonetic transcriptions as defined in [Wells *et al*, 1992]. The indicated phonetic classes are those used as regression classes.

Phoneme class	Phonetic symbol	Danish word	Transcription
Plosives	p	pande	pan@
	b	bande	ban@
	t	tand	tan
	d	dan	dan
	k	kalde	kal@
	g	galde	gal@
Fricatives	f	finde	fen@
	s	stand	sdan
Approximants	v	vinde	ven@
	D	bide	biD@
	j	jul	jul
	h	hest	hEsd
Nasals	m	mile	mil@
	n	ny	ny
	N	lunge	loN@
Liquids	l	land	lan
	R	ride	RiD@
Unrounded front vowels	i	ville	vil@
	e	visse	ves@
	E	tække	tEk@
	ltub ({)	male	m{l@
	a	malle	mal@
Central vowels	A	pakken	pAgn
	@	*unstressed final e*	
Rounded front vowels	y	hylde	hyl@
	_2	kølle	k_2l@
	_9	hønse	h_9ns@
Back vowels	u	kulde	kul@
	o	foto	fodo
	O	munde	mOn@
	Q	kors	kQs

Table C.1: *The phonetic symbols used in EUROM.1 [Wells et al, 1992]*

C.3 Statistics

Table C.2 shows the division of sentences and passages into training and testing parts for the adaptation speakers.

Table C.3 shows the division of age and gender of the speakers in EUROM.1. In table C.4 to C.6 the number of occurances of the different phonemes in the reference and adaptation part of EUROM.1 is given.

Speaker ID	Training part	Testing part
ba	O0..O3,O5..O8,P0..P3,F0..F4,Z0..Z4	O4,O9,P4
an	P5..P8,Q0..Q3,Q5..Q8,F5..F9,Z5..Z9	P9,Q4,Q9
ld	Q5..Q8,R0..R3,R5..R8,F5..F9,Z5..Z9	Q9,R4,R9
bl	P5..P8,Q0..Q3,Q5..Q8,F5..F9,Z5..Z9	P9,Q4,Q9
hm	O0..O3,O5..O8,P0..P3,F0..F4,Z0..Z4	O4,O9,P4
jo	O5..O8,P0..P3,P5..P8,F5..F9,Z5..Z9	O9,P4,P9
jd	Q0..Q3,Q5..Q8,R0..R3,F0..F4,Z0..Z4	Q4,Q9,R4
eb	O0..O3,O5..O8,R5..R8,F5..F9,Z5..Z9	O4,O9,R9
il	O0..O3,R0..R3,R5..R8,F0..F4,Z0..Z4	O4,R4,R9
js	P0..P3,P5..P8,Q0..Q3,F5..F9,Z5..Z9	P4,P9,Q4

Table C.2: *Division of passages and sentences from the EUROM.1 database into training and testing parts for the adaptation and reference speakers respectively.*

Age Category	Reference speakers				Adaptation speakers			
	Talker ID (Male)		Talker ID (Female)		Talker ID (Male)		Talker ID (Female)	
16 - 25	14	CC CD EJ FK	7	BC DA GJ LC	1	BA	2	AN LD
		HC KH KT LA		LN UM VN				
		LB LM MR NR						
		PA TK						
26 - 35	9	BF BR HS JM	4	JA KM LO ML	1	BL	0	
		JT LJ LL OA						
		SD						
36 - 45	1	SP	4	IJ KJ LE MS	2	HM JO	1	JD
46 - 55	5	FC HE JN PD	4	EC HG ID IV	1	EB	2	IL JS
		UH						
56 -	1	EH	1	AV	0		0	
TOTAL	30		20		5		5	

Table C.3: *Distribution of age and gender*

Label	**ba**	**an**	**ld**	**bl**	**hm**	**jo**	**jd**	**eb**	**il**	**js**	**Total**
SIL	265	510	526	219	460	160	261	372	95	85	2953
SP	163	256	275	224	371	374	292	195	225	268	2643
l	176	222	202	217	240	200	211	190	195	211	2064
R	78	107	109	107	111	101	102	91	94	98	998
m	132	140	145	140	184	144	140	153	148	130	1456
n	285	334	346	335	393	316	334	347	334	306	3330
N	24	46	52	46	34	38	40	44	41	32	397
v	96	84	81	86	135	77	90	101	97	88	935
D	110	116	103	116	150	111	121	113	101	117	1158
j	69	76	65	76	99	83	74	65	72	93	772
h	56	62	79	62	84	49	80	66	81	60	679
f	85	101	91	101	120	105	94	86	87	107	977
s	243	321	34	321	364	322	275	302	282	300	2764
p	55	44	46	44	67	31	69	40	67	62	525
b	79	103	101	103	117	88	93	98	81	76	939
t	69	104	93	104	96	84	94	84	90	94	913
d	228	270	248	268	333	294	236	272	238	263	2650
k	60	58	52	58	86	65	70	49	62	73	575
g	116	158	172	158	178	153	151	153	150	142	1531
u	76	124	108	123	114	108	90	113	90	94	1040
o	51	71	76	71	67	56	80	55	6	62	595
O	62	74	74	74	91	71	71	71	63	67	718
Q	275	323	296	321	372	316	307	296	295	323	3121
y	35	25	23	25	43	23	37	22	39	42	314
_2	35	45	45	45	53	40	48	39	43	44	437
_9	30	34	24	34	38	35	31	26	24	38	314
A	121	157	163	157	187	160	152	160	151	157	1565
@	160	200	174	186	240	205	201	178	202	210	1956
i	177	189	205	189	253	200	206	208	208	186	2021
e	145	211	223	211	227	198	173	219	185	165	1957
E	124	152	134	152	180	138	151	142	145	151	1469
ltub	32	37	33	34	45	33	36	37	30	29	346
a	205	215	209	218	293	232	207	223	218	223	2243
Total	3917	4969	4607	4625	5765	4610	4617	4610	4239	4396	46355

Table C.4: *Number of occurrences per speaker of the phonemes in the training part of the material from the adaptation speakers.*

Label	**ba**	**an**	**ld**	**bl**	**hm**	**jo**	**jd**	**eb**	**il**	**js**	**Total**
SIL	5	60	43	11	36	11	20	33	2	9	230
SP	25	37	36	25	42	50	37	27	30	44	353
l	26	27	36	26	34	29	26	37	43	29	313
R	15	18	19	18	16	18	15	20	17	18	171
m	22	22	16	22	30	30	19	26	23	29	239
n	40	43	42	43	46	44	46	39	42	54	439
N	4	6	7	6	4	6	4	6	5	4	52
v	13	16	19	16	19	16	16	21	13	9	158
D	18	10	10	8	21	15	16	18	20	17	153
j	2	15	17	15	2	7	16	9	14	11	108
h	15	12	8	12	19	20	12	10	10	22	140
f	8	14	20	14	9	13	12	18	15	13	136
s	39	30	48	30	47	42	32	55	53	35	411
p	3	6	2	6	4	7	3	1	0	7	39
b	17	12	17	12	19	14	17	17	22	14	161
t	7	15	18	15	9	9	15	14	12	7	121
d	40	37	37	36	50	51	29	53	44	40	417
k	9	6	4	6	8	8	6	6	6	8	67
g	18	15	14	15	22	22	12	19	20	22	179
u	12	9	14	9	10	14	12	10	11	11	112
o	5	10	8	10	5	5	9	6	7	7	72
O	7	7	7	7	9	12	4	10	9	11	83
Q	42	46	78	46	50	46	52	71	67	43	541
y	6	3	4	3	7	3	4	7	6	4	47
_2	3	10	7	10	5	4	10	4	3	5	51
_9	6	4	16	4	6	5	5	16	15	5	82
A	22	29	27	29	25	32	26	22	31	33	276
@	28	24	24	23	33	33	20	35	30	32	282
i	29	29	36	29	34	26	34	36	38	28	319
e	33	34	30	34	39	48	25	40	28	30	341
E	12	16	27	16	19	18	18	24	25	15	190
ltub	4	6	2	6	5	8	4	2	1	8	46
a	24	28	24	28	31	30	28	32	24	28	277
Total	585	656	717	590	715	696	601	711	686	652	6609

Table C.5: *Number of occurrences per speaker of the phonemes in the testing part of the material from the adaptation speakers.*

Labels	Train	Test
SIL	808	98
SP	2561	577
l	2363	458
R	1130	244
m	1666	334
n	3639	790
N	411	116
v	1080	192
D	1335	208
j	844	203
h	766	215
f	1105	203
s	3316	726
p	550	89
b	1070	201
t	997	214
d	303	545
k	692	114
g	1572	405
u	1139	235
o	680	160
O	749	163
Q	3562	811
y	342	58
_2	486	73
_9	389	79
A	1735	499
@	2197	415
i	2301	516
e	2191	546
E	1632	353
ltub	400	49
a	2502	495
Total	46513	10384

Table C.6: *Number of observed occurrences of the phonemes in the training and testing part of the material from the reference speakers.*

Appendix D
The HTK1.5 Toolkit

D.1 Description of used tools

HCode

HCode has been used for parametising the speech data into MFCC's.

HHEd

HHEd is a tool for editing HMM definition files. It has been used for increasing the number of mixtures in the HMM definition files between running the estimation tools HInit, HRest and HERest. See section D.2 for a more detailed description of where in the training process HHEd has been used.

HInit

The initial training of the HMM models is done with HInit that recomputes means and variances of a given model. The computation is based on a standard Viterbi algorithm for aligning of the data vectors. The output files of HInit are used as input files for the HRest tool.

HRest

HRest has been used for doing a Baum-Welch re-estimation of all parameters in a given HMM definition file. The output files of HRest are used as input files for the HERest tool.

HERest

The HERest tool is used for re-estimating the parameters of a set of HMM definition as opposed to HInit and HRest, where only parameters of a single HMM definition is re-estimated. HERest uses an embedded training version of the Baum-Welch algorithm, where the parameters in the set of models corresponding to a complete utterance are re-estimated at the same time.

`HVite`

Once a set of fully trained HMM's has been generated using the above mentioned tools, the performance of the speech recognition system can be measured using `HVite`. The tool performs a Viterbi recognition of the test data and outputs label files with the recognised label sequences. These label files can be used as input to `HResults`.

`HResults`

`HResults` compares the label files from `HVite` with the correct label files and outputs the result using the correctness measure and the accuracy measure as defined below.

Correctness:

$$\%\text{Correct} = \frac{H}{N} \times 100\% \tag{D.1}$$

Accuracy:

$$\text{Accuracy} = \frac{H - I}{N} \times 100\% \tag{D.2}$$

where H is the number of correct labels, I is the number of insertions and N is the total number of labels in the defining transcription files.

D.2 Training procedure

```
for all models
    HInit
    HRest
    HRest
end
while num_mix < max_mix
    HHEd (num_mix = num_mix+1)
    HERest
end
HERest
```

Appendix E
Program Design

This appendix documents the `MLLR` program. The functional design and function hierarchy is described in section E.1. Finally in section E.2 the function descriptions and flowcharts for the major functions are given. The enclosed floppy disk contains source code files and a compiled version of the program `MLLR`.

E.1 Functional design

The program is controlled via the command line in a way similar to the HTK1.5 tools to facilitate the interactions with those. In the functional design of the program a number of smaller functional units are defined. This is illustrated in figure E.1.

Each of the functional modules has been implemented with a number of program functions. As stated before the final program is a mixture of slightly modified functions from the HTK1.5 tool `HERest`, (in the file `HERest_M.c` at the floppy disk) and own implemented functions (in the file `MLLR.c with the header file MLLR.h`. The `HERest_M` performs an alignment of an input speech sequence using the Forward-Backward algorithm. Using this alignment the mixture component occupancy variable is calculated, and this variable is used in the further computations of the transformation matrixes.

In figure E.2 the function hierarchy is sketched.

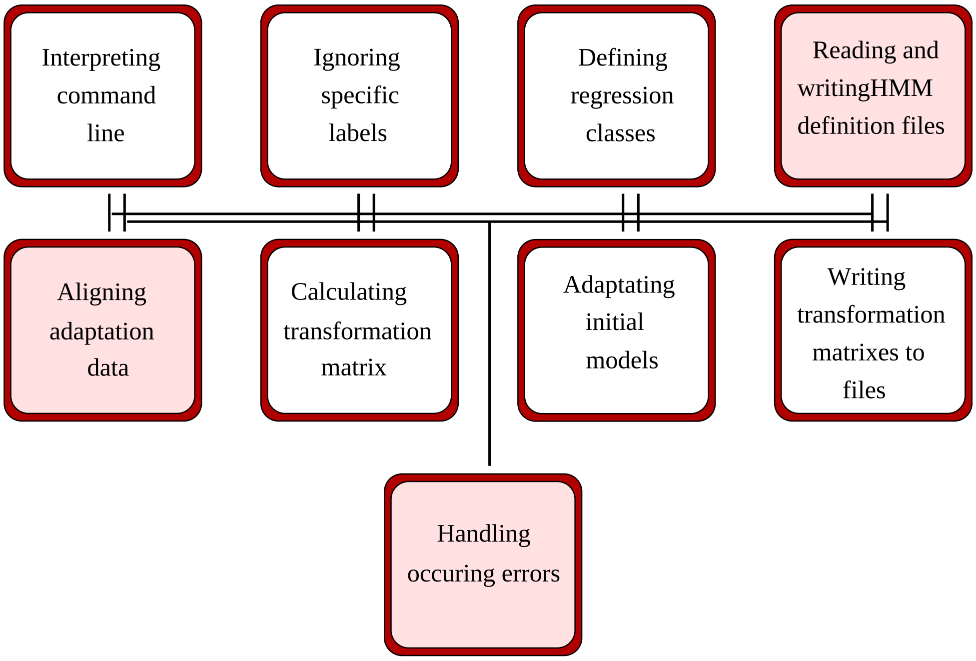

Figure E.1: *Functional design for* `MLLR`*. The program is divided into 8 smaller function modules. The grey boxes are based primarily on HTK1.5 functions. The white modules are implemented as part of the current project.*

Figure E.2: *Function hierarchy.*

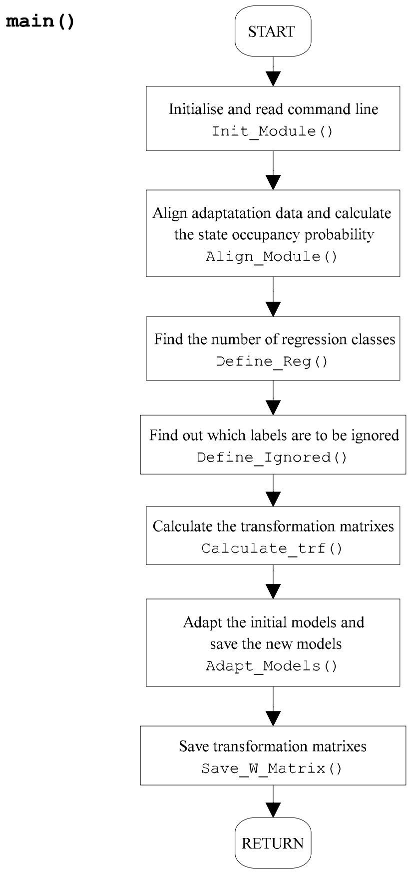

Figure E.3: *Flowchart for the mainloop of `MLLR`.*

E.2 Function descriptions

In figure E.3 a flowchart for the mainloop of `MLLR` is given.

Name: `void Adapt_Models()`
Input: none
Output: none
Description: All SI HMM's are stepped through. For each model the model structure is read and the means are updated by a call to **`AdaptMean()`**.
A flowchart is shown in figure E.4

Name: `void AdaptMean(int px, HMMDef hmm)`
Input: **`int px`** the number of the model.
`HMMDef hmm` a pointer to the hmm structure.
Output: none
Description: The current regression class for the hmm number **`px`** is found by a call to **`GetRegClass()`**. The means are updated as $\hat{\boldsymbol{\mu}} = \hat{\boldsymbol{W}}_r \cdot \boldsymbol{\xi}$, where $\hat{\boldsymbol{\mu}}$ is the adapted mean vector, $\hat{\boldsymbol{W}}_r$ is the transformation matrix for regression class r and $\boldsymbol{\xi}$ is the extended mean vector for the original system.

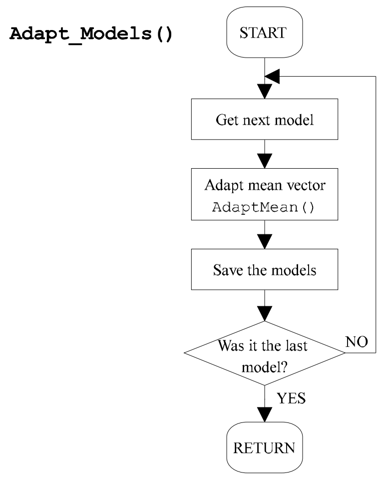

Figure E.4: *Flowchart for the function* `Adapt_Models`.

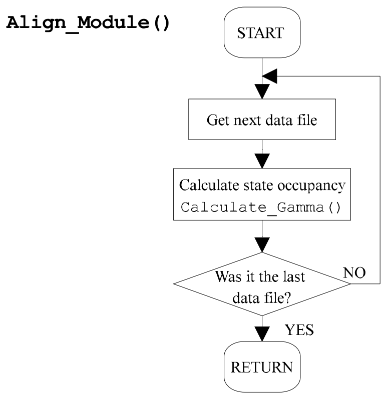

Figure E.5: *Flowchart for the function* `Align_Module`.

Name: `void Align_Module()`
Input: none
Output: none
Description: For all the data files specified at the command line or in the scriptfile an forward-backward alignment is performed where the state occupancy variable is calculated. This is done by a call to `Calculate_gamma()`. A flowchart is shown in figure E.5.

Name: `void Calculate_gamma(char *datafn)`
Input: `char *datafn` Pointer to data file name.
Output: none
Description: Called by `Align_Module()` to perform a forward-backward alignment for the current data file, and calculate the state occupancy variables. These variables are stored as a part of the hmm structures.

Name: `void Calculate_trf()`
Input: none
Output: none
Description: Is called by `main` after the data has been aligned and the regression classes have been defined. It

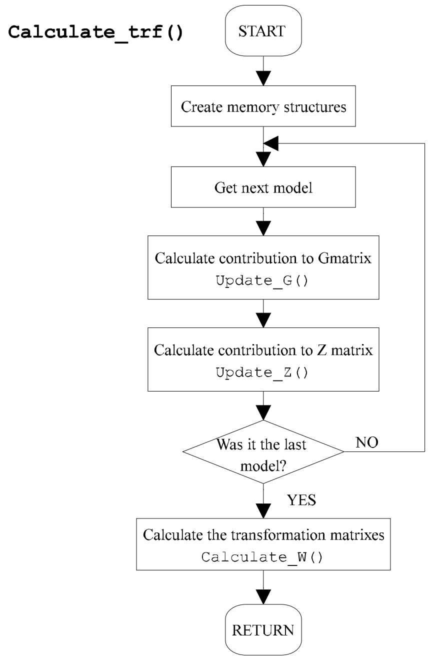

Figure E.6: *Flowchart for the function `Calculate_trf`.*

calculates a transformation matrix for every regression class. It runs through all data files and updates the corresponding matrixes (`Update_G(), Update_Z()`). Finally the transformation matrixes are calculated (`Calculate_W()`).
A flowchart is shown in figure E.6.

Name: `void Calculate_trf_diag()`
Input: none
Output: none
Description: Is called by `main` after the data has been aligned and the regression classes have been defined. It calculates a transformation matrix for every regression class. It runs through all data files and updates the corresponding matrixes (`Update_A(), Update_B()`). Finally the transformation matrixes are calculated (`Calcualte_W()`).

Name: `void Calculate_W()`
Input: none
Output: none
Description: For all regression classes the transformation matrix, W is calculated using equation 5.16.

Name: `void Calculate_W_diag()`
Input: none
Output: none
Description: For all regression classes the transformation matrix, W is calculated using equation 5.24.

Name: `void Check_Reg_Class()`
Input: none
Output: none
Description: Checks whether there are adaptation examples in all regression classes. Exits if there are zero examples in a class.

Name: `void CreateA()`
Input: none
Output: none
Description: Allocates memory for A[numReg][vSize*2][vSize*2]

Name: `void CreateB()`
Input: none
Output: none
Description: Allocates memory for B[numReg][vSize*2]

Name: `void CreateG()`
Input: none
Output: none
Description: Allocates memory for G[numReg][vSize][vSize+1][vSize+1]

Name: `void CreateGinv()`
Input: none
Output: none
Description: Allocates memory for Ginv[vSize+1][vSize+1]

Name: `void CreateInvMat()`
Input: none
Output: none
Description: Allocates memory for U[vSize+1][vSize+1], w[vSize+1], V[vSize+1][vSize+1] and tmp[vSize+1][vSize+1].

Name: `void CreateW()`
Input: none
Output: none
Description: Allocates memory for W[numReg][vSize][vSize+1]

Name: `void CreateZ()`
Input: none
Output: none
Description: Allocates memory for Z[numReg][vSize][vSize+1]

Name: `void Define_Ignored()`
Input: none
Output: none
Description: Determines whether a label is to be ignored according to the value of `Minus`.

Name: `int Define_Reg_div2()`
Input: none
Output: `int numReg` The number of regression classes
Description: Divides into two regression classes: vowels and consonants.

Name: `int Define_Reg_div10()`
Input: none
Output: `int numReg` The number of regression classes
Description: Divides into 10 regression classes: liquids, nasals, approximants, fricatives, plosives, back vowels, rounded front vowel, central vowel, unrounded front vowel, silence.

Name: `void Define_Reg_local()`
Input: none
Output: `int numReg` number of regression classes.
Description: Divides into 33 regression classes: one for each label.

Name: `int GetRegClass(int px)`
Input: `int px` The number of the model.
Output: none
Description: The number of the regression class that `px` belongs to.

Name: `void Init_Module(int argc, char *argv[])`
Input: `int argc` The number of input parameters to the overall program.
Output: `char *argv[]` An array containing the input parameters.
Description: If `MLLR` is called with an empty commandline an usage message is printed (`ReportUsage()`), otherwise the command line options and files are read and the corresponding variables initialised.

Name: `void Invert_a(int Reg)`
Input: `int Reg` The current regression class.
Output: none
Description: The current A matrix is inverted by making a singular value decomposition. (`svdcmp()`).

Name: `void Invert_G(int Reg, int i)`
Input: `int Reg` The current regression class.
`int i` The current feature
Output: none
Description: The current G matrix is inverted by making a singular value decomposition. (`svdcmp()`).

Name: `int NotIgnored(int px)`
Input: `int px` The current HMM.
Output: `int Not` =1 if current HMM is not ignored.
Description: The current G matrix is inverted by making a singular value decomposition. (`svdcmp()`).

Name: `void ReportUsage()`
Input: none
Output: none
Description: Prints the following text when `MLLR` is called without any arguments:

```
USAGE: MLLR [options] hmmList adaptFiles

Option                                  Default

-R r   Use multiple regression classes  global
       r=0: global
       r=1: euromdiv2
       r=2: euromdiv10
       r=3: local
-D     Use diagonal transf. matrix       full
-M m   Group of labels to ignore         none
       m=x: phonetic class x, 0<x<9
       m=1000: none
       m=100: vowels
       m=200: consonants
 -i s   Extension for SI models           none
 -I d   Dir to find SI models            current
 -l s   Extension for label files          lab
 -L d   Dir to find label files          current
 -m t   Offset term for W                   0
 -o s   Extension for SD models          as src
 -O d   Dir to save SD models            current
 -F fmt Set data file format to fmt        HTK
 -G fmt Set label file format to fmt       HTK
 -P f   Set the pruning threshold to f     0.0
 -S s   Set script file to s              none
 -T t   Trace level                         0
 -w s   Set W-file to s                   none
 -W d   Dir to save W files             current
```

Name: `void svdcmp()`
Input: `float ** a` Pointer to array that are to be SVD compositioned
`int m` First dimension of `a` array.
`int n` Second dimension of `a` array.
`float *w` Pointer to memory where `w` will be saved.
`float **v` Pointer to memory where `v` will be saved.
Output: none
Description: Calculates the singular value decomposition of the array `a` into the matrixes, `U`, `w` and `v`. `U` is returned in `a`.

Name: `void Update_A(int px, HMMDef *hmm)`
Input: `int px` The current regression class.
`HMMDef *hmm` A HMMDef structure containing the current model.
Output: none
Description: Updates the `A` structure according to equation 5.21.

Name: `void Update_B(int px, HMMDef *hmm)`
Input: `int px` The current regression class.
`HMMDef *hmm` A HMMDef structure containing the current model.
Output: none
Description: Updates the `B` structure according to equation 5.22.

Name: `void Update_G(int px, HMMDef *hmm)`
Input: `int px` The current regression class.
`HMMDef *hmm` A HMMDef structure containing the current model.
Output: none
Description: Updates the `G` structure according to equation 5.13.

Name: `void UpdateZ(int px, HMMDef *hmm)`
Input: `int px` The current regression class.
`HMMDef *hmm` A HMMDef structure containing the current model.
Description: Updates the `Z` structure according to equation 5.14

Bibliography

[Ahadi and Woodland, 1995] SM Ahadi, PC Woodland. Rapid Speaker Adaptation Using Model Prediction. *Proceedings ICASSP-95*, Michigan, 684-687, 1995.

[Choukri *et al*, 1986] K Choukri, G Chollet, Y Grenier. Spectral Transformation Through Canonical Correlation Analysis for Speaker Adaptation in ASR. *Proceedings ICASSP-86*, Tokyo, 2659-2662, 1986.

[Class *et al*, 1990] F Class, A Kaltenmeier, P Regel, K Trottler. Fast Speaker Adaptation for Speech Recognition Systems. *Proceedings ICASSP-90*, Alberquerque, 133-136, 1990.

[Cox, 1993] SJ Cox. Speaker Adaptation Using a Predictive Model. *Proceedings EUROSPEECH-93*, Berlin, 2283-2286, 1993.

[Cox, 1995] SJ Cox. A Speaker Adaptation Technique Using Linear Regression. *Proceedings ICASSP-95*, Michigan, 700-703, 1995.

[Cox and Bridle, 1989] SJ Cox, JS Bridle. Unsupervised Speaker Adaptation by Probabilistic Spectrum Fitting. *Proceedings ICASSP-89*, Glasgow, 294-297, 1989.

[Cox and Bridle, 1990] SJ Cox, JS Bridle. Simultaneous Speaker Normalisation and Utterance Labelling Using Bayesian/Neural Net Techniques. *Proceedings ICASSP-90*, Alberquerque, 161-164, 1990.

[Gauvain and Lee, 1994] JL Gauvain, CH Lee. Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains. *IEEE Trans on Speech and Audio Processing*, Vol. 2, No. 2, April 1994.

[Gong, 1995] Y Gong. Speech Recognition in noisy environments: A survey. *Speech Communication* 16:261-91.1995.

[Huang *et al*, 1990] XD Huang, Y Ariki, MA Jack. *Hidden Markov Models for Speech Recognition*. Edinburgh University Press, 1990.

[Huang and Lee, 1991] XD Huang, KF Lee. On Speaker-Independent, Speaker-Dependent, and Speaker-Adaptive Speech Recognition. *Proceedings ICASSP-91*, Toronto, 877-880, 1991.

[Johnson and Duggeon, 1993] DH Johnson, DE Dudgeon. *Array Signal Processing* Prentice Hall. 1993.

[Leggetter and Woodland, 1994] CJ Leggetter, PC Woodland. *Speaker Adaptation of HMM's Using Linear Regression.* Cambridge University, Technical Report, TR.181, June 1994.

[Leggetter and Woodland, 1995a] CJ Leggetter, PC Woodland. Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models. *Computer Speech and Language* Vol 9, p 171-185, 1995.

[Leggetter and Woodland, 1995b] CJ Leggetter, PC Woodland. Flexible Speaker Adaptation for Large Vocabulary Speech Recognition. *Proceedings EUROSPEECH95* 1155-1158, 1995.

[Leggetter, 1995] CJ Leggetter. *Improved Acoustic Modelling for HMM's using Linear Transformations.* Ph.D. thesis, Cambridge University, 1995.

[Lindberg and Christensen, 1995] B Lindberg, H Christensen. *Documentation of the Danish EUROM.1 Database.* Esprit Project 2589 (SAM) Multi-Lingual Speech Input/Output Assessment, Methodology and Standardisation, SAM-IES-68, 1995.

[Lindberg, 1994] B Lindberg. Recogniser Response Modelling From Testing on Series of Minimal Word Pairs. *Proceedings ICSLP-94*, Yokohama, Japan, 1994.

[Mariani, 1989] J Mariani. Recent Advances in Speech Recognition. *Proceedings ICASSP-89*, Glasgow, 429-440, 1989.

[Mermelstein and Davis, 1980] P Mermelstein, SB Davis. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing* Vol. ASSP-28, No. 4, August 1980.

[Moore, 1989] BCJ Moore. *An Introduction to the Psychology of Hearing*, Academic Press Limited. 1989.

[Press *et al*] WH Press, BP Flannery, SA Teukolsky, WT Vetterling. *Numerical Recipes in C*, Cambridge University Press. 1990

[Rabiner, 1989] L Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *Proceedings of the IEEE*, vol 12,no. 2, Feb. 1989.

[Rabiner and Juang, 1993] L Rabiner, BH Juang. *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

[Ross, 1987] SM Ross. *Introduction to Probability and Statistics for Engineers and Scientists.* John Wiley & Sons. 1987.

[Shikano *et al*, 1986] K Shikano, KL Lee, R Reddy. Speaker Adaptation through Vector Quantization. *Proceedings ICASSP-86* Tokyo, 2643-2646, 1986.

[Wells *et al*, 1992] J Wells, W Barry, M Grice, A Fourcin, D Gibbon. *Standard Computer-Compatible Transcription* SAM STAGE REPORT Sen.3. ESPRIT Project 2589 (SAM) Multi-Lingual Speech Input/Output Assessment, Methodology and Standardisation. SAM-UCL-037. 28. Feb. 1992.

[Young *et al*, 1993] SJ Young, PC Woodland, WJ Byrne. *HTK: Hidden Markov Model Toolkit V1.5* Cambridge University Engineering Department, Speech Group and Entropic research Laboratories Inc. 1993.

[Zavaliagkos *et al*, 1995] G Zavaliagkos, R Schwartz, J Makhoul. Batch, Incremental and Instantaneous Adaptation Techniques for Speech Recognition. *Proceedings ICASSP-95*, Michigan, 676-679, 1995.

[Zavaliagkos, 1995] G Zavaliagkos. *Maximum a posteriori Adaptation Techniques for Speech Recognition.* Ph.D. thesis. Northeastern University, Boston, Massachusetts, 1995.

[Zhao, 1994] Y Zhao. An Acoustic-Phonetic-Based Speaker Adaptation Technique for Improving Speaker-Independent Continuous Speech Recognition. *IEEE Trans on Speech and Audio Processing*, Vol. 2, No. 3, July 1994.

Nomenclature list

Abbreviations

DTW	Dynamic Time Warping
HMM	Hidden Markov Model
MFCC	Mel Frequency Cepstral Coefficient
MLLR	Maximum Likelihood Linear Regression
SD	Speaker Dependent System
SI	Speaker Independent System
SVD	Singular Value Decomposition

Notation

$\boldsymbol{\lambda}$	Set of model parameters
μ	Mean vector
Σ	Covariance matrix
M	Number of mixtures
N	Number of states in a model
R	Number of regression classes
S	Number of states in the system
n	Number of features
$\boldsymbol{W}$	Transformation matrix
$\boldsymbol{\xi}$	Extended mean vector
ω	Offset term
P	Number of utterances
T_p	Number of frames in observation p
$\boldsymbol{O}_P$	Set of P utterances
$\boldsymbol{O}^{(p)}$	Observation sequence p
$\boldsymbol{o}_t^{(p)}$	Observation vector for time t and observation p
θ	State
$\boldsymbol{\theta}$	State sequence
$\boldsymbol{\Theta}$	Set of all possible state sequences
K	Mixture component sequence
Ω_b	Set of all possible mixture component sequences
$\gamma(t)$	Mixture component occupancy probability
$Q(\cdot)$	Auxiliary function
$P(\cdot)$	Probability function
α	Forward variable
β	Backward variable