Analysis and Implementation of the Speaker Adaptation Techniques: MAP, MLLR and MLED

Robert M. Fanner

December 2002

Study Leader: Prof. J.A. du Preez

Thesis presented in partial fulfilment of the requirement for the degree of **Master of Electronic Engineering** at the *University of Stellenbosch*

I, the undersigned, do hereby declare that the work contained in this thesis is my own original work except where otherwise indicated, and has not previously in its entirety or in part been submitted at any university for a degree.

Abstract

The topic of this thesis is speaker adaptation, whereby speaker-independent speech models are adapted to more closely match individual speakers by utilising a small amount of data from the targeted individual. Speaker adaptation methods — specifically, the MAP, MLLR and MLED speaker adaptation methods — are critically evaluated and compared.

Two novel extensions of the MLED adaptation method are introduced, derived and evaluated. The first incorporates the explicit modelling of the mean speaker model in the speaker-space into the MLED framework. The second extends MLED to use basis vectors modelling inter-class variance for classes of speech models, instead of basis vectors modelling inter-speaker variance.

An evaluation of the effect of two different types of feature vector — PLP-cepstra and LPCCs — on the performance of speaker adaptation is made, to determine which feature vector is optimal for speaker-independent systems and the adaptation thereof.

Opsomming

Die onderwerp van hierdie tesis is spreker-aanpassing, dit wil sê, die verandering van 'n spreker-onafhanklike spraakmodel om nader aan 'n spreker-afhanklike model vir 'n individu te wees, gegewe 'n klein hoeveelheid spraakdata van die individu. Die volgende sprekeraanpassing-metodes word geëvalueer: MAP, MLLR en MLED.

Twee nuwe uitbreidings vir die MLED-metode word beskryf, afgelei en geëvalueer. Die eerste inkorporeer die eksplisiete modellering van die gemiddelde sprekermodel van die sprekerruimte in die MLED metode. Die tweede uitbreiding maak gebruik van basis-vektore vir MLED wat vanaf die interklas-variansie tussen 'n stel sprekerklasse in plaas van die interspreker-variansie afgelei is.

Die effek van twee tipes kenmerk-vektore — PLP-kepstra en LPCC's — op die prestasie van sprekeraanpassings-metodes word ondersoek, sodat die optimale tipe kenmerk-vektor vir spreker-onafhanklike modelle en hul aanpassing gevind kan word.

Acknowledgements

First and foremost, I wish to thank my supervisor, prof Johan du Preez, for giving me the opportunity to study, as well as for the guidance and help without which this thesis would have been impossible.

A big thank-you my family, friends and girlfriend for understanding, support (moral and financial), and simply being there.

To my fellow lab-rats: thank-you for creating a wonderful working environment, for borrowed pens, staples and knowledge, and for the camaraderie. Special mention must be made of Lude Schwardt, for being a walking repository and fount of knowledge and his willingness to discuss new and old ideas.

Finally, I wish to thank the NRF for their financial support.

Contents

List of Figures

List of Tables

List of Abbreviations

CAT Cluster adaptive training

CBKLT Class-based Karhunen-Loéve transform

CD Compact disk

CDHMM Continuous density hidden Markov model

EM Expectation maximisation

GM Gaussian model

GMM Gaussian mixture model

HMM Hidden Markov model

iid Independent and identically distributed

KLT Karhunen-Loéve transform (another term for PCA)

LDA Linear discriminant analysis

LP Linear prediction

LPC Linear prediction coefficient

LPCC Linear prediction coefficient cepstrum

MAP Maximum *a posteriori*

ML Maximum likelihood

MLED Maximum likelihood eigenvoice decomposition

MLLR Maximum likelihood linear regression

PCA Principal Component Analysis

pdf Probability density function

PLP Perceptual linear prediction

RMP Regression-based model prediction

SD Speaker dependent

SI Speaker independent

WP Weighted projection

List of Symbols

$l(O|\lambda)$ Likelihood for the observation sequence O given the current set of parameters λ

$L(O|\lambda)$ Log-likelihood for the observation sequence O

$Q\left(\lambda|\lambda^{[k]}\right)$ Auxiliary function for EM algorithm implementations, where $\lambda^{[k]}$ is the current set of model parameters and λ is the new set of model parameters to be estimated

$Q(\lambda, \hat{\lambda})$ Auxiliary function for EM algorithm implementations, where λ is the current set of parameters and $\hat{\lambda}$ is the new set of parameters that will be estimated

λ A set of model parameters

$\tilde{\lambda}$ The ML estimate of a set of model parameters

$\hat{\lambda}$ The new (and not necessarily ML) estimate of a set of model parameters

$\mathbf{o}_t$ The observation for time t

T The length of an observation sequence

O An observation sequence

π The initial state distribution for an HMM

π_i The probability of beginning in state i of an HMM

A The state transition probability matrix

a_{ij} The state transition probability for going from state i to state j

$b_m^s(\mathbf{o})$ Gaussian pdf for mixture component number m of the GMM for state s of an HMM evaluated for vector $\mathbf{o}$

μ_m^s Mean vector of a Gaussian pdf for mixture component number m of the GMM for state s of an HMM

C_m^s Covariance matrix of a Gaussian pdf for mixture component number m of the GMM for state s of an HMM

$\gamma_m^{(s)}(t)$ Mixture occupation probability for time t

$\gamma^{(s)}(t)$ State occupation probability for time t

ξ_m^s An extended mean vector for the mean vector of a Gaussian pdf for mixture component number m of the GMM for state s of an HMM, where an extended mean vector consists of a mean vector with a one or zero prepended to it

W_α MLLR transformation matrix for regression class α

μ An MLED supervector, or a mean vector, depending on the context

$\mathbf{e}$ An MLED eigenvoice

$\mathbf{e}_m^{(s)}$ An MLED eigenmean for mixture component m of a GMM for state s of an HMM

w An MLED eigenweight

$\mathcal{D}$ The original speaker space for weighted projection

$\mathcal{T}$ The mapped speaker space for weighted projection

$\mathcal{K}$ The original eigenspace for weighted projection

S_w Intra-class dispersion matrix for the CBKLT

S_b Inter-class dispersion matrix for the CBKLT

Chapter 1

Introduction

1.1 Preamble

Speaker adaptation techniques originated due to the needs of the speech recognition milieu of the late 1970s and early 1980s. Most speech recognition systems rely on the analysis of short frames of speech in order to form feature vectors, and then model these feature vector sequences using statistical models. Now — as in the 1980s — the state of the art in speech recognition is to use hidden Markov models as the statistical models for feature vector sequences. Typically LPC (linear prediction coefficients), LPCC (linear prediction coefficient cepstra), PLP (perceptual linear prediction) or MFCC (Mel-frequency warped cepstral coefficients) are used as feature vectors.

Initially, two kinds of system were available to cope with speech recognition problems: speaker-independent (SI) systems and speaker-dependent (SD) systems. Speaker-independent systems are trained using the data from many speakers, and the resulting system has to cope with the task of recognising the speech of a great variety of individual speakers. Speaker-dependent systems, however, are trained using speech data of the target speaker it is designed to recognise. SD systems generally outperform their SI counterparts — typically having error rates that are two to three times lower [20, 33]. The better performance of SD models is due to the fact that they only need to model the intra-speaker variability of a single speaker, whereas SI models model the intra-speaker and inter-speaker variability of a set of speakers. This performance is not without cost, as speech recognition systems require a great deal of data to train, and all the data for

an SD system have to be obtained from the target speaker. Definite benefits could be reaped if the recognition performance of an SI system could be improved for individual speakers, or if the exorbitant amount of data required from a single individual to train an SD system could be reduced.

Enter speaker adaptation techniques — techniques originating due to the necessity to improve the performance of an SI system for individual speakers, and operating by utilising a small amount of observation training data from an individual speaker with which to modify the recognition system to better recognise the particular individual — referred to as the *target speaker*. Speaker adaptation techniques are not only used to adapt for differences in the target speaker, as they are often employed to adapt a recognition system to compensate for varying acoustic and noise conditions. Speaker adaptation techniques can be roughly categorised into two groups: speaker normalisation and speaker (model) adaptation.

Speaker normalisation occurs when the SI model is left unchanged, but the features of the speaker are normalised to match the features used to train the SI (reference) model. Such methods attempt to map the input vectors of all speakers to the reference model using a single transform. Examples of speaker normalisation methods include VTLN (vocal tract length normalisation) [45, 50], dynamic frequency warping [46] and the use of self-organising feature maps [26]. Speaker normalisation methods have been found to be ineffective, yielding only small (if any) improvements in recognition [33, 3], due to the complex mappings required and the fact that the same mapping is applied to the input features of different new speakers — allowing little modelling for the differences between speakers.

Speaker adaptation occurs when the SI model is adapted in order to better model a specific speaker. Results attained using these methods are generally superior to those obtained using speaker normalisation approaches, so that most research today focuses on the development of new speaker adaptation methods. Speaker adaptation methods include maximum *a posteriori* reestimation (MAP) [18], regression-based model prediction (RMP) [1], maximum likelihood linear regression (MLLR) [34], cluster adaptive training (CAT) [14], maximum likelihood eigenvoice decomposition (MLED) [28, 40], and weighted projection (WP) [51].

We focus on speaker adaptation techniques that reestimate HMM model parameters — specifically the MAP, MLLR, MLED and WP adaptation methods. Subsequent introductory sections consist of an introduction into speaker adaptation concepts, a chronologically ordered overview of selected speaker adaptation techniques, the objectives and contributions of this work and an overview of the treatment of topics in this thesis.

1.2 Speaker Adaptation Concepts

The following concepts are of importance when considering speaker adaptation: inter-speaker and intra-speaker variation, the adaptation mode, the choice of parameters to adapt and adaptation performance.

1.2.1 Inter- and Intra-Speaker Variation

Intra-speaker variation refers to the variations in the speech signal produced by one speaker between different utterances of the same text. Inter-speaker variation refers to the variations between speech signals produced by different speakers when uttering the same text. Both of these variations associated with human speech production have an impact on the performance of speech models and speaker adaptation methods. For instance, SI models have to cope with modelling inter-speaker variation and intra-speaker variation for many speakers. SD models, on the other hand, need only model the intra-speaker variation of a single individual. The intra-speaker variation of one individual is normally much less than the combined inter- and intra-speaker variation of several speakers. Therefore, it is to be expected that variances of SD model parameters are less than the variances of corresponding SI model parameters. The resulting increase in the precision of model parameter estimates thus affords SD models more accurate modelling capability than SI models, observable in the lower word error rates achieved when using SD models.

Intra- and inter-speaker variance also influence the design and performance of speaker adaptation systems. For instance, should we wish to use adaptation to train an SD model for a speaker given the SI model and a some adaptation data,[1] we are attempting

[1]The term *adaptation data* refers to the available observation data for a speaker that can be used to modify a recognition system to better model and recognise the speaker in question.

to remove the inter- and intra-speaker variability of the speakers used to train the SI model and replace them with the intra-speaker variability of the target speaker.[2] When modelling the speech of an individual, other problems arise as speech characteristics of the person may vary due to factors such as illness or emotion. We might then adapt an initial SD model to better recognise the individual's speech under varying circumstances. In effect, we attempt to change the intra-speaker variability represented in our model to better match the current attributes of the individual's speech.

1.2.2 Inter-Speaker Variation

We use classification of inter-speaker variation proposed by Ahadi and Woodland [2] for this discussion.[3] In this model, inter-speaker differences originate from two main sources: anatomical differences between speakers, and speaker habits.

Anatomical differences refer to the characteristics of the vocal apparatus, such as vocal tract length and shape. One of the greatest sources of inter-speaker variation is the person's gender. Female speakers tend to have higher fundamental frequencies, and female vocal tracts are shorter, resulting in wider formant bandwidths than that of male speakers.

Speaker habit refers to the way in which a speaker has learned to speak. The speech rate is such a habit, and it is normally due to a personal preference. Accent is another source of learned behaviour, and accents can arise due to the speaker's socio-economical environment, the ethnic group to which the speaker belongs, whether the language is native of non-native to the speaker, and the region in which the speaker was raised.

1.2.2.1 Intra-Speaker Variation

Sources of intra-speaker variation include differences arising from the physical and mental conditions of the speaker. The emotion anger may make a person's speech louder and increase the speech rate. A physical condition such as tiredness may cause slower, slurred speech. A cold, causing a blocked nose, may remove the nasal component of speech.

[2]The term *target speaker* refers to the speaker from whom adaptation data is obtained, and for whom the recognition system is to be adapted.

[3]More accurate classifications of sources of speaker variability exist, i.e. [42]. For our needs, however, the simpler classification will suffice.

Obviously, these short-term effects will cause a degradation in recognition performance, as the speech model must now cope with very different speech data compared to that on which it was trained. According to Leggetter [33], an ideal speech adaptation scheme should be able to compensate for such short-term fluctuations in speech. However, he notes that such schemes are not possible at present due to the poor understanding of these short-term effects.

1.2.3 Modes of Speaker Adaptation

When performing speaker adaptation, we are faced with the choice of mode to use. We can choose between supervised vs. unsupervised, static vs. dynamic and adaptive vs. non-adaptive adaptation.

- **Supervised vs. Unsupervised:**
  - **Supervised adaptation** occurs when the identity of the observed speech is known, i.e. a transcription of the observed adaptation data is available.
  - **Unsupervised adaptation** occurs when the identity of the observed speech is unknown, i.e. no transcription is available for the observed adaptation data. Supervised adaptation is preferred over unsupervised adaptation, as it does not require a segmentation of the received speech signal based on the recognition ability of the initial SI model (or set of SD models).
- **Static vs. Dynamic:**
  - **Static** or batch adaptation occurs when all of the available adaptation data is gathered before the model is reestimated.
  - **Dynamic** or incremental adaptation occurs when the model is reestimated as soon as an adequate amount of data has been observed, and then further refined as more adaptation data become available.
- **Adaptive vs. Non-Adaptive:** Here the term adaptive does not refer to the actual speaker adaptation process, but to the method of training used in large tasks involving the training of SD models for many speakers. It should be noted that not all adaptation methods use an SI model as the initial model on which adaptation

is performed. Many methods, mostly adaptive, use a set of SD models from which an initial model is generated. Such a set of SD models is often referred to as a canonical model set.

- **Adaptive** training is used when the same adaptation method was used to create the SD models present in the speaker database as the adaptation method used to create an SD model for a new speaker. Often these methods use information extracted from the well-trained models in the database to serve as a prior when adapting a model for a new speaker. Adaptation methods that facilitate adaptive training, such as RMP and CAT, allow the rapid generation of large sets of SD models.
- **Non-adaptive** training is used when the adaptation method cannot incorporate a set of SD models as prior, or if the adaptation method requires prior SD models and cannot train new SD models without it. Such methods include the eigenvoice methods MLED and WP, which completely rely on a well-defined eigenspace extracted from a large set of SD models before a model for a new speaker can be estimated.

The choice of adaptation mode is dependent on the task at hand. For instance: should a large set of SD models be trained, with new models being added to the system on a regular basis with moderate amounts of available observation data, an adaptive method such as CAT would be appropriate. If a task requires the once-off training of an SD model for a new speaker, it would be desirable to have the speaker utter a phonetically representative sentence, with known transcription, so that static, supervised adaptation can be used. On the other hand, should an SD model for a person be trained online while the person is using the recognition system, a dynamic unsupervised mode of adaptation may be needed.

Other issues that impact the choice of adaptation scheme include the relative improvement in the adapted speaker models given various amounts of data, and the computational requirements of the adaptation. Should a great deal of data be available, MAP adaptation would be a suitable choice, as it is a computationally lightweight adaptation and it converges to the ML estimate as the amount of adaptation data increases. However, if

very little adaptation data are available, one of the eigenvoice methods should be used, as they have the ability to form robust model estimates using very little observation data.

As this thesis involved the incorporation of well-known speaker adaptation methods into a software toolkit and the development and testing of new speaker adaptation methods, experiments were done using the most forgiving adaptation mode: supervised, static adaptation.

1.3 A Brief Semi-Chronological Overview of Speaker Adaptation Methods

It is our opinion that the best way to rapidly attain an understanding of speaker adaptation topologies and their capabilities is to be shown examples. Thus an abridged history of speaker adaptation techniques is traced here by describing a select few of the possible model adaptation techniques. Not all methods described here are treated in this thesis. However, they are mentioned as they give some indication of other possible speaker adaptation schemes, and because each of them represents a broad class of speaker adaptation techniques. The following broad classes of speaker adaptation topologies may be identified:

- Bayesian adaptation methods, such as ML reestimation of HMM parameters and MAP adaptation, as well as simple extensions of these methods. The original ML reestimation and MAP adaptation methods (and most of the methods based on them) are not rapid[4] adaptation techniques, as they cannot adapt parameters for which no data was observed. Though ML reestimation of HMM parameters is not normally viewed as a speaker adaptation technique, one can sometimes refer to it as such, because it may be employed to generate SD models given copious amounts of adaptation data and an initial SI model.

- Spectral transformation approaches, where the input speech of a new speaker is mapped (using a transform estimated for the new speaker) to match the speaker

[4] A speaker adaptation method is only considered to be rapid if it can reestimate parameters to which no observation data is assigned, by relying on relationships with other model parameters for which data was observed.

on which the system was trained. Mappings are either applied to the input vector, or to the entire model. Spectral transformation methods are not equivalent to speaker normalisation methods — though there are some resonances between the two — as the latter only maps the input speech, and use the same mapping for all new speakers. The first spectral transformation methods were developed in the early 1980s for use with recognisers based on spectral templates. Later methods focus on the adaptation of VQ codebooks for HMMs. These methods will receive no further treatment, as this thesis focuses solely on speaker adaptation techniques that reestimate HMM model parameters. An excellent summary containing descriptions and references for a multitude of spectral transformation techniques may be found in [33].

- Methods relying on linear regression and linear transformations to adapt the HMMs of a speaker model. These methods all attempt to find linear relationships between model parameters, so that the linear relationships may by exploited to adapt parameters for which no data were observed from parameters for which data were observed. We will refer to parameters to which no observation data was assigned as unseen parameters, and to parameters to which observation data was assigned, as seen parameters.

- Speaker clustering techniques. Instead of using an SI model as initial model, a set of SD models are trained. These SD models are then clustered, and a single cluster model is estimated for all the SD models represented by the cluster. It is then assumed that a new speaker may be represented by one of the cluster models. Adaptation consists of estimating to which cluster the target speaker belongs given the adaptation data obtained from the target speaker.

- Eigenvoice methods: methods inspired by the concept in human face recognition of using a weighted sum of eigenfaces to model an individual's face. A set of eigenvoices are determined in an off-line pre-adaptation phase, and it is assumed that a new speaker may be represented as a weighted sum of the eigenvoices. Adaptation consists of estimating the optimal set of weights given the adaptation data obtained from the target speaker.

We now proceed with our semi-chronological overview and treat the following adaptation methods: ML, MAP, MLLR, RMP, CAT, MLED and WP. It should be noted that all of these methods are applied to recognition systems making use of HMMs (or sets of HMMs) with GMM state pdfs. Furthermore, a speaker (dependent or independent) model for these methods is defined as the set of HMMs — where each HMM represents a phoneme or word — used to model speech for the speaker. A brief description of each is given, as well as the reestimation formula for a mean vector in a GMM in an HMM of a speaker model.

1.3.1 ML adaptation

Maximum likelihood (ML) estimation is the standard method used to train hidden Markov models, and it has been used since the inception of HMMs in speech modelling. Since part of the data needed to form ML estimates for HMM parameters is hidden, the EM algorithm (see Section 2.3.3) is used to estimate the hidden data (the state sequence or state occupation probabilities) and determine the optimal parameter set. The ML estimate $\tilde{\mu}_m^{(s)}$ of a mean vector is given by[5]

$$\tilde{\mu}_m^{(s)} = \frac{\sum_{t=1}^{T} \gamma_m^{(s)}(t)\mathbf{o}_t}{\sum_{t=1}^{T} \gamma_m^{(s)}(t)}, \tag{1.1}$$

where $\mu_m^{(s)}$ is the original mean in state mixture component m of state s in the SI model, $\gamma_m^{(s)}(t)$ is the mixture occupation probability (see Section 2.4.4) and $\{\mathbf{o}_t\}_{t=1,\ldots,T}$ is the sequence of observation vectors.

Robust ML reestimation, where only those parameters with an adequate amount of observation data assigned to them are reestimated, may be seen as an early form of speaker adaptation. Alternatively (or additionally) deleted interpolation [47] may be used. Here the new model estimate is formed using a weighted sum of the SI model and ML estimate, so that

$$\text{New model estimate} = (1-\epsilon)(\text{SI model}) + \epsilon(\text{ML estimate}), \tag{1.2}$$

[5]In this thesis we distinguish between a Bayesian (ML or MAP) estimate of a parameter and an adaptation method estimate of a parameter by using tildes and hats. If λ is the parameter, then the ML or MAP estimate is denoted as $\tilde{\lambda}$, and any adaptation method estimate is denoted as $\hat{\lambda}$

where $\epsilon, 0 \leq \epsilon \leq 1$, is essentially a confidence measure placed on the ML estimate. The less data are available, the less robust the ML estimate will be and the smaller ϵ should be made.

The new estimate $\hat{\mu}_m^{(s)}$ for a mean vector in a GMM in an HMM of the speaker model using deleted interpolation is

$$\hat{\mu}_m^{(s)} = (1-\epsilon)\mu_m^{(s)} + \epsilon \frac{\sum_{t=1}^{T} \gamma_m^{(s)}(t)\mathbf{o}_t}{\sum_{t=1}^{T} \gamma_m^{(s)}(t)}. \tag{1.3}$$

Both these implementations of ML estimation attempt to make the ML estimate more robust in sparse data conditions. It should be noted that they cannot adapt parameters for which no observation data was available, and are thus not rapid adaptation methods.

1.3.2 MAP adaptation

Maximum *a posteriori* (MAP) adaptation of HMM parameters was introduced by Gauvain and Lee [18] in 1994. MAP estimation includes the prior pdf of the parameters to be estimated into the estimation process. In this way, any prior knowledge we have of the parameters is utilised, so that less data is necessary to form a robust MAP estimate of a parameter than the amount of data required for a robust ML estimate. When only the mean vectors are adapted, the MAP reestimation formula for a mean vector is

$$\tilde{\mu}_m^{(s)} = \frac{\tau_m^{(s)}\mu_m^{(s)} + \sum_{t=1}^{T} \gamma_m^{(s)}\mathbf{o}_t}{\tau_m^{(s)} + \sum_{t=1}^{T} \gamma_m^{(s)}}, \tag{1.4}$$

where $\tau_m^{(s)}, \tau_m^{(s)} \geq 0$ is a parameter originating from prior density for the mixture component m in state s.

Equation 1.4 has several interesting properties: As the number of observation vectors goes to infinity ($T \to \infty$), the MAP estimate approaches the ML estimate. Also, the prior parameter $\tau_m^{(s)}$ behaves as a confidence measure between the SI model and new ML estimate. As $\tau_m^{(s)} \to \infty$, all our confidence is placed in the original SI model, and as $\tau_m^{(s)} \to 0$, we trust that the new ML estimate is robust. It should be noted that the behaviour of MAP reestimation of mean vectors is very similar to that of deleted interpolation (Equation 1.3), save that the MAP estimate will converge to the ML estimate as the amount of available adaptation data becomes greater. Though MAP adaptation is more robust than ML estimation, it still shares the weakness of not being able to adapt parameters for which no data was available.

1.3.3 MLLR adaptation

Since its introduction in 1995 by Leggetter and Woodland [34, 33], maximum likelihood linear regression (MLLR) has become one of the most popular speaker adaptation methods. MLLR is a rapid speaker adaptation technique, i.e. it has the ability to adapt parameters for which there was no observed data, given that there was at least some observed data for other parameters in the speaker model. Like all the rapid speaker adaptation techniques treated, it relies on parameter reduction for robust estimation of unseen parameters. MLLR accomplishes parameter reduction by assuming that classes of mean vectors in the speaker model undergo similar linear transformations when a new speaker model is trained. The same linear transform is thus used to adapt the set of mean vectors belonging to the same class in the original SI model. For instance, mean vectors in GMMs belonging to HMMs representing similar phonemes in a speaker model may be grouped in a class, as it is expected that they will all undergo a similar change for a new target speaker. The MLLR reestimation formula for a mean vector is of the form

$$\hat{\mu}_m^{(s)} = T\mu_m^{(s)} + \mathbf{b}, \tag{1.5}$$

where T is a matrix and $\mathbf{b}$ is an offset vector calculated for the regression class to which $\mu_m^{(s)}$ belongs. T performs rotation and scaling of the original mean vector, and $\mathbf{b}$ adds an offset to map the mean vector of the initial model to a mean vector for the new target speaker model.

Today several adaptation methods exist that employ linear transformations to change the SI model to fit a new target speaker. Due to their superb parameter reduction capabilities, these methods can all yield robust parameter estimates in conditions where very little observation data are available.

1.3.4 RMP adaptation

Regression-based model prediction (RMP) was introduced a while after MLLR in 1995 by Ahadi and Woodland [1]. RMP uses a set of well trained SD models as prior information. Linear relationships between parameters in a speaker model may be found using the information of the set of speaker models. The linear relationships may be found in the following way: One parameter y is labelled as a target, and another parameter x to which

we wish to relate it, is labelled as the source parameter. We assume that there is a linear relationship between the source and the target such that

$$y = b_1 x + b_0 + \epsilon, \tag{1.6}$$

where b_1 and b_0 are the regression parameters to be estimated and ϵ is the error associated with the approximation. If there are K speakers in the prior SD model set, then the regression parameters may be found by minimising the sum of the squared errors for all K speakers, i.e. we minimise

$$\sum_{k=1}^{K} \epsilon_k^2 = \sum_{k=1}^{K} (y_k - b_1 x_k - b_0)^2, \tag{1.7}$$

to obtain the regression parameters. Alternatively, multiple regression relationships may be found such that linear relationships are found between multiple sources and the target parameter:

$$y = b_0 + \sum_{l=1}^{P} b_l x_l + \epsilon, \tag{1.8}$$

where P is the order of the regression.

For any target parameter y, we can also calculate the x source parameter with which it is most highly correlated.

When a new speaker model is to be estimated, a MAP estimate is formed for the parameters given the available adaptation data. Those parameters that were robustly estimated are now labelled as source parameters, and parameters that were adapted using little data or parameters for which there were no observed data are labelled as target parameters. For each of the target parameters, we determine the most correlated source parameter (or parameters, for multiple regression), determined from the set of prior models. A new regression-based estimate is then made for each target parameter y using the source parameter (or parameters, for multiple regression) with which it is most correlated. In this way the linear regression estimates for parameters with no data may be made using the robust MAP estimates of parameters with sufficient observation data.

In the final RMP step, the MAP estimates and regression estimates for the parameters are combined in a MAP framework. The resulting equation for RMP reestimation of a mean vector is given by

$$\hat{\mu} = \frac{\mu_{\text{MAP}}\, \sigma_{\text{LR}}^2 + \mu_{\text{LR}}\, \sigma_{\text{MAP}}^2}{\sigma_{\text{LR}}^2 + \sigma_{\text{MAP}}^2}, \tag{1.9}$$

where μ_{MAP} is the original MAP estimate, μ_{LR} is the linear regression estimate based on the other original MAP estimates, σ^2_{MAP} is the variance associated with the MAP estimate, σ^2_{LR} is the variance associated with the linear regression estimate and $\hat{\mu}$ is the new estimate of the mean vector.

As the amount of observation data for the variable increases, σ^2_{MAP} decreases, and σ^2_{LR} increases, so that the RMP estimate consists mostly of the more accurate MAP estimate. The converse holds when little data are observed, as σ^2_{LR} is small and σ^2_{MAP} is large, so that the RMP estimate consists mostly of the linear regression estimate μ_{LR}. For the limiting conditions of infinite data or no data, the following holds: when infinite data are available, the RMP estimate reduces to the MAP estimate, and when no data is available, the RMP estimate reduces to the linear regression estimate.

RMP is a rapid adaptation technique. It should be noted that RMP is outperformed (by a factor of almost two) by MLLR for conditions where observation data is very to moderately sparse. However, as the amount of observed data is increased, RMP begins to perform as well as MLLR [3]. When the observation data becomes sufficient for robust MAP estimates of many mean vectors, RMP outperforms MLLR. This is to be expected, as the RMP estimate converges to the MAP estimate, whereas the MLLR estimate does not.

1.3.5 CAT

Cluster adaptive training (CAT), introduced by Gales [14] in 1998, is an adaptation method that makes use of speaker clustering. First, a set of well-trained SD models are created. These models are then clustered into P clusters, and a single model is trained to represent all the SD models in a cluster. With initial speaker clustering methods, a new speaker is modelled by assigning one of the cluster models to represent the speaker. CAT, however, models a new speaker by using a weighted sum of the cluster models. In addition to the weighted sum of clusters, a mean vector $\mathbf{b}$ from a bias cluster may be included in the summation with weight 1. Essentially the bias cluster then models speaker-independent mean vector characteristics, and the bias cluster may be incorporated without increasing the number of parameters that need to be estimated for a new speaker.

Instead of using a set of different weights for the estimation of each mean vector,

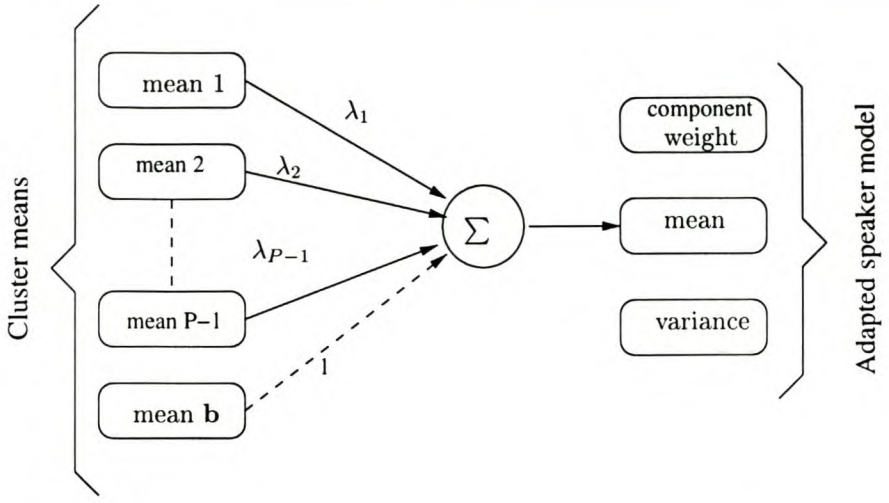

Figure 1.1: Mean vector modelling in CAT. $\lambda_1, \ldots, \lambda_{P-1}$ are the weight vectors, and $\mathbf{b}$ is the offset vector (from Gales [14]).

weight classes may be created. Any mean vector in a weight class is then estimated from the cluster means by applying the same set of class weights. Thus, for a mean vector belonging to weight class $r, m \in M^{(r)}$, the new estimate of a speaker mean $\hat{\mu}^{(m)}$ is given by

$$\hat{\mu}^{(m)} = M^{(m)}\lambda^{(r)}, \tag{1.10}$$

where $M^{(m)}$ is a matrix with the cluster means as columns, i.e.

$$M^{(m)} = [\mu^{(m1)} \quad \ldots \quad \mu^{(mP)}], \tag{1.11}$$

where $\mu^{(mp)}$ is the mean of Gaussian component m associated with cluster p, and the extended weight vector for weight class r, $\lambda^{(r)}$, is given by

$$\lambda^{(r)} = [\lambda_1^{(r)} \quad \ldots \quad \lambda_{P-1}^{(r)} \quad 1]'. \tag{1.12}$$

For a new speaker, the set of weight vectors for the speaker is estimated given the available adaptation data and applied to Equation 1.10 in order to obtain the new set of speaker mean vectors.

CAT facilitates *adaptive training*, i.e. both the SD models used to create the clusters as well as the new speakers modelled are trained using the same adaptation scheme — in this instance, CAT. CAT architectures may be iteratively trained using training data from a set of new speakers using two stages:

1. The values of the weight vectors for each speaker are estimated using the training data for that speaker, given the current cluster parameters.

2. The parameters for each cluster are reestimated given the estimated weight vectors. The parameters reestimated for each cluster include the mean vectors, covariance matrices and component weights of all the Gaussian pdfs represented in the cluster.

The above steps are iterated until some convergence criterion is met.

The above CAT implementation uses model-based clusters. Alternatively, transform-based clusters may be specified, i.e. the mean vectors for each cluster are not specified directly. Instead, each cluster mean vector is modelled as an MLLR transform of some canonical model.

According to Gales [14], CAT has roughly the same performance as other adaptive methods. It may also be used to yield small improvements when very little adaptation data is available.

1.3.6 MLED adaptation

Inspired by the use of "eigenfaces" used in human face recognition, Kuhn [27] introduced the concept of eigenvoices for speaker adaptation in 1997. Kuhn postulated that the inter-speaker variability in a set of SD models may be modelled in space with lower dimension than the space spanned by the parameters of the original SD models. In order to form the lower-dimensional space, "supervectors" are created for each SD model (every SD model comprises of several phonemes, and represents the entire speech model for the speaker) in the set, where each supervector consists of all the SD model parameters we wish to adapt. Eigenvectors (of the same dimension as the supervectors), referred to as eigenvoices, are then extracted from the set of SD supervectors. Typically very few eigenvoices are needed to accurately model inter-speaker variability. A supervector for a new speaker may now be approximated by using a weighted sum of the dominant eigenvoices. The weights are referred to as "eigenweights". An estimate of a mean vector using eigenvoice adaptation schemes is given by

$$\hat{\mu} = \sum_{j=1}^{E} w_j \bar{\mu}(j) \tag{1.13}$$

where $\bar{\mu}(j)$ is the section of eigenvector j modelling the mean vector in question, w_j is the estimated eigenweight associated with $\bar{\mu}(j)$, and E is the number of eigenvoices utilised.

A highly effective adaptation method using Kuhn's original postulate, MLED (maximum likelihood eigenvoice decomposition) was developed by P. Nguyen [40] in 1998. Nguyen states that adaptation for a new speaker may be achieved by simple projection, i.e. the ML estimate for the new speaker is constrained to lie in the eigenspace by projecting the ML estimate onto the eigenspace, and then reprojecting the constrained ML estimate back into the normal speaker space. Projection, however, is suboptimal for two reasons: One, parameters for which no new ML estimates are available (due to lack of data) are also projected into the eigenspace, and thus play a role in the adaptation. Two, the projection of the ML estimate onto the eigenspace does not necessarily represent the highest value of the likelihood function representable in the eigenspace. For this reason, Nguyen developed MLED, where MLED forms an optimal estimate of the eigenweights given the adaptation data directly in the eigenspace. An ML estimate is thus made taking the eigenspace constraint into mind, instead of first forming the ML estimate and then enforcing the eigenspace constraint.

Eigenvoice adaptation techniques are to date the adaptation methods that require by far the smallest amount of adaptation data for successful parameter adaptation. Even the data of one or two phonemes are enough to yield a small improvement in performance. Also, MLED adaptation is on par with most MLLR adaptation schemes where moderate amounts of adaptation data are available.

1.3.7 WP

Weighted projection is a newer eigenvoice adaptation method, and it was introduced by Westwood [51] in 1999. The extraction of eigenvoices and the modelling of a new mean vector (Equation 1.13) are identical to those used in MLED. The estimation of the eigenweights, however, is accomplished using a projection technique. Instead of using normal projection, discounted by Nguyen for good reasons, Westwood did the following:

An ML estimate of parameters is formed in the original full-dimensional speaker space. The ML-estimated parameters in the original space are then mapped to a new space, $\mathcal{T}$. The mapping is such that each parameter is divided by its variance and multiplied by

weighting dependent on the parameters' occupancy (the number of observation vectors assigned to it). Thus parameters with no observed data map to the null vector in $\mathcal{T}$, and parameters with lower variance receive preference over those with high variance. The closest point in the eigenspace to the point for the speaker model in the space $\mathcal{T}$ is then found, where the closest point then represents the estimate of the new set of eigenweights. These eigenweights are then used to estimate a mean vector for the target speaker using Equation 1.13.

The performance of WP is on par with that of MLED. This is no surprise, as the estimation equations for the eigenweights are very similar for both methods.

1.4 The Choice of Parameters to Adapt when HMMs are Employed

Ideally, when a speaker adaptation method is used to adapt an HMM-derived recognition system, the adaptation method in question should be capable of adapting all of the HMM parameters. Of the speaker adaptation methods mentioned in this thesis, only ML, MAP and CAT adaptation have the ability to reestimate all HMM parameters — given that enough observation data was observed for each parameter. Very few methods have the ability to adapt the state transition probabilities of HMMs. However, as Leggetter [33] states, the effect of state transition probabilities is small in a continuous density HMM, and thus not adapting state transition probabilities will not severely affect the performance of a speaker adaptation method. Similar arguments may be made for neglecting to adapt mixture component weights of GMMs, as GMMs may be expanded into HMMs, where the mixture component weights become state transition probabilities. Also, the best speaker recognition systems currently use GMMs, where temporal information of the observation sequence from a speaker is neglected. This seems to indicate that most of the differences between speakers lie in the distributions associated with the observation vectors, and not in the transitional information in the sequence of the observation vectors. Reestimation of state transition probabilities for a new speaker is thus not considered to be of paramount importance for successful speaker adaptation.

Many methods, including the basic MLLR scheme as well as the eigenvoice adapta-

tion techniques, rely solely on the adaptation of the mean vectors present in the state pdf GMMs present in the HMMs of a speaker. Though they do not adapt all HMM parameters, the success of these methods indicate that highly effective speaker adaptation is possible when focusing only on mean vectors and neglecting covariance matrices. It should also be noted that robust reestimation of covariance matrices requires more adaptation data than the robust reestimation of mean vectors — a fact that is of importance when employing Bayesian adaptation methods such as ML and MAP reestimation.

1.5 Objectives

This work was aimed at research in the field of speaker adaptation, specifically the adaptation of speaker models comprising continuous density hidden Markov models, where single or mixture Gaussian state probability density functions are used. The following objectives were set:

- A literature study of speaker adaptation, so that the most effective adaptation techniques may be selected.
- Detailed study of the available material on the selected adaptation methods.
- Coding of speaker adaptation routines to expand the in-house DSP and pattern recognition software package PatrecII.
- Experimentation, both for validation of the code and confirmation of the results of other research efforts.
- Further experimentation to determine behavioural aspects of the adaptation methods.
- Development of novel speaker adaptation approaches, or enhancements for existing speaker adaptation methods.

These objectives were met in the following way:

- A literature study of speaker adaptation techniques was completed. During the study trends in adaptation techniques were noted, so that speaker adaptation techniques could (for the most part) be classified as belonging to one of the following:

Bayesian adaptation schemes, spectral transformation schemes, speaker clustering schemes, linear transformation or regression schemes, and eigenvoice schemes.

- The adaptation methods selected for further study were MAP, MLLR and MLED. MAP was selected on merit of being a computationally lightweight algorithm and the fact that the MAP estimate converges to the ML model estimate as the observed data increase. MLLR was selected as a matter of course. It is a very effective and well-known adaptation method — so much so that it is often used as a benchmark against which the performance (both in terms of computational complexity and recognition performance) of other adaptation methods are tested. It also represents a wide field of adaptation techniques based on tying and linear transformations of the speaker model. The most effective adaptation methods for extremely sparse observation data conditions are the eigen-decomposition methods. MLED was selected to represent these methods, as it is the first eigen-decomposition method created and the body of literature for the method is consequently larger than that of other eigenvoice methods. Though newer eigen-decomposition methods such as WP exist, their performance is similar that of MLED.

- PatrecII is a very flexible software package, and it allows for the implementation of statistical structures in an almost Mobius-strip-like quality. An HMM, for instance, may have state pdfs comprised of HMMs, neural networks or Gaussian mixture models. A state pdf consisting of a mixture density model may in turn consist of an HMM, which in turn may have states consisting of mixture density models, etc. The speaker adaptation methods selected were for the most part very rigid, as they are only applicable to continuous density HMMs with mixture Gaussian state pdfs. MLLR and MLED also only adapt the mean vectors of Gaussian pdfs in the HMMs forming a speaker model.

  Due to the flexibility of PatrecII, and the relatively inflexible nature of the adaptation methods, they proved challenging to incorporate in the software package. Initially they could only be incorporated via pointer promiscuity. After changes to the package by prof. J. du Preez, however, a more elegant design was implemented.

- In order to test the performance of PatrecII using the newly incorporated speaker adaptation methods, the set of experiments performed by P. Nguyen on the ISOLET speech corpus was repeated. As the ISOLET speech corpus represents a relatively small recognition problem, many other experiments could be set up and executed speedily in order to attain a better feel for the behaviour of the adaptation methods.

- Further experiments were completed on the larger TIMIT speech corpus, in order to further test existing methods and to evaluate the performance of the MLED extensions designed in this thesis.

- Two novel enhancements of the eigenvoice methods were designed and implemented. One is a simple method which incorporates the direct modelling of the original mean of the set of SD models into the MLED framework. Incorporation of the original mean is essentially a better modelling of the speaker space using an eigenspace, and the result was improved recognition performance. The second enhancement extends eigen-decomposition methods via the class-based Karhunen-Loéve method. Here we have the ability to model classes of speaker models instead of modelling individual models. It was assumed that by using this adaptation method it would be possible to adapt for speakers belonging to different dialect classes.

1.6 Contributions

During the experiments, an interesting observation regarding the behaviour of speaker adaptation methods was made:

- Experiments were conducted on the use of PLP-cepstral and LPCC features for the speaker models. As found by other researchers, SI models trained using PLP-cepstral features outperform models using LPCC features by up to 3%. Thus, the pre-adaptation models (ML-estimated SI models) show better results when PLP-cepstral features are employed. However, the post-adaptation models for most methods tested show better results when LPCC features are used.

  PLP-cepstral and MFCC features outperform LPC and LPCC features for recognition systems that use a single SI model, therefore perceptual features were sub-

sequently used for systems using several SD models as well as for systems where speaker adaptation of models take place. The results of this thesis show that the commonly accepted practice of using perceptual features for all kinds of systems is not necessarily good, as the type of feature employed has an effect on the performance of speaker adaptation and the performance of SD models. It is our opinion that the information discarded when forming perceptual features gives these features some speaker normalisation attributes, thereby increasing the performance achieved with SI systems. The speaker normalisation, however, removes some of the inter-speaker variance that can be utilised by speaker adaptation methods, causing poorer adaptation. Furthermore, the larger the degree of inter-speaker variance, the greater the differentiation between SD models will be, so that an individual's speech is (possibly) more accurately modelled by his SD model.

This once more opens the issue of which type of feature vector is superior, as it now seems that the performance of the type of system is linked to the type of feature vector employed. It is hoped that this thesis will lead to detailed studies of the effect of the type of feature vector on the performance of different recognition systems.

Furthermore, two extensions to the eigenvoice decomposition methods were made:

- Including the average of the available SD models into the modelling method employed by the MLED technique is a simple yet effective extension to the adaptation method. It has the effect of adding the modelling power of a robustly estimated extra eigenvoice to the adaptation procedure at very little extra computational cost.

- Researchers working with eigenvoice speaker adaptation methods have postulated that other methods for extracting eigenvoices might lead to better performance. One method in particular — linear discriminant analysis (LDA), referred to in this thesis as the CBKLT — was cited by several researchers as a possibility [28, 40, 51]. MLED was therefore extended to use basis vectors — determined via LDA (CBKLT) — that model the inter-class variance between several speaker classes instead of the inter-speaker variance. The method is shown to be effective for adaptation according to the speaker's gender, but has slightly poorer performance than standard MLED for a limited recognition task with speakers originating from different dialect

regions. The results indicate that even though the CBKLT extracted eigenvoices resulted in satisfactory performance, their performance was not on par with eigenvoices extracted using the KLT (PCA). Further uses for a class-based MLED might be found — such as language or environmental adaptation — but in our opinion it is doubtful that CBKLT/LDA-based adaptation will outperform KLT/PCA-based speaker adaptation.

1.7 Overview of the Thesis

The treatment of speaker adaptation is divided into three phases: The material in the first phase (Chapter 2) is intended as a propadeutic treating the current speech modelling and recognition approach. Though it is by no means sufficient for a complete understanding, important concepts used throughout the thesis such as ML (*maximum likelihood*) estimation are touched upon. As it treats familiar concepts in speech recognition, it should aid the reader in understanding the form of notation employed in the rest of the thesis. A maximum likelihood framework from which all of the adaptation methods are derived is introduced at the end of the first phase, and it completes the primer for the study of the rest of the thesis.

Having laid the speech recognition background, the chosen speaker adaptation techniques MAP, MLLR and MLED are treated in Chapters 3, 4 and 5.

As it is possibly the most well known of the adaptation methods, and as it is the only non-rapid speaker adaptation method, MAP adaptation will receive only cursory treatment (see Chapter 3).

MLLR is afforded a more detailed treatment, with complete derivation of the basic adaptation equations and discussion on the various clustering schemes. As the volume of literature on MLLR is great, and given the large amount of MLLR variations and derived methods — all using linear transformations to adapt HMM parameters — only basic MLLR is treated. Brief mention is made of other adaptation methods employing linear transformations (see Chapter 4 for MLLR adaptation). The use of eigenvoices is relatively new, and the volume of literature on the subject is relatively small. As the implementation, analysis and extension of eigenvoice methods form the major drive of

the thesis, the derivation of the basic MLED adaptation equations receives full treatment (see Chapter 5).

WP (*weighted projection*) is a newer eigenvoice adaptation method, and a full derivation of the basic adaptation equations as well as a detailed description of the concepts involved are included. WP is identical to MLED in terms of computational complexity and recognition performance — so similar, in fact, that certain implementations of WP reduce to the same adaptation equations as those used for MLED. Due to the similarity between the methods, WP was not employed in the experimentation. However, WP represents a completely different approach and view on eigenvoice adaptation, and is thus treated in full (see Chapter 6).

Two novel extensions slotting into the MLED adaptation framework are treated in Chapter 7. All eigenvoice methods rely on a set of eigenvoices that are extracted via the Karhunen-Loéve transform (PCA) from a set of well trained SD models. The first extension includes the modelling of the mean of the set of SD models into the MLED framework, resulting in small changes in the final MLED adaptation equations. This simple enhancement results in improved performance, at very little extra computational cost.

The second extension makes use of the CBKLT (class-based Karhunen-Loéve transform) to extract a set of vectors spanning a subspace. These vectors are not eigenvectors — they do not even have the property of orthogonality; neither is the subspace an eigenspace. However, as the method fits into MLED adaptation equations, this thesis will still refer to the extracted vectors as "eigenvectors", though the term is not strictly applicable. The CBKLT-extended MLED was designed to cope with recognition tasks where moderately disparate accents (differences in pronunciation) are present in the expected target speakers. Instead of modelling differences between the models of individual speakers, CBKLT-extended MLED models the differences between classes of speaker models, where every class comprises the speakers with a certain accent or dialect.

Chapters 8 to 9 deal with the experiments. The initial experiments on the ISOLET speech corpus (for the most part a repetition of P. Nguyen's work) are treated in Chapter 8. Further experiments on the more complex TIMIT database are treated in Chapter 9.

Experiments on both the ISOLET and TIMIT corpora were aimed at highlighting the

strengths and weaknesses of each adaptation method. Some experiments showed that MAP is inferior to the other methods for conditions of extreme data scarcity. Others were designed to show how unbalanced data might adversely affect MLLR adaptation. Further experiments tested the conditions of extreme data scarcity under which only the eigenvoice methods could still yield good estimates. Experiments to examine how using different types of feature vector impacts on the adaptation methods were completed for both speech corpora. Two different types of feature vector were analysed, namely LPC features and PLP features. PLP features normally give recognition performance that is a few percent higher than that of LPC features. However, LPC features yield better post-adaptation improvement than PLP features for both the speech corpora. It is thus not always true that PLP features are superior to LPC features.

These experiments emphasised one of the findings of other researchers: each adaptation method has strengths and weaknesses, and most of the weaknesses are only encountered under very specific data conditions. The expected amount and kind of available adaptation data will thus play an important role in the selection of a suitable technique for every speaker adaptation task. Furthermore, new findings regarding the performance of the MLLR and MLED adaptation methods for different features and different speaker model complexities resulted from the experiments. Also, the experiments conducted prove that the novel extensions to the eigenvoice methods introduced in this thesis function as they were designed to, and that both of the methods are successful. The performance of the mean-preserving MLED implementation is such that it should be used in favour of the original MLED method when the amount of adaptation data is extremely little and only a single eigenvoice is used. The CBKLT-extended MLED implementation is shown to have the same performance as standard MLED when a large inter-class variance exists between several speaker (accent or gender) classes in the speaker set. For more detail on the resulting conclusions, refer to Chapter 10.

Chapter 2

Speech Modelling in Brief

2.1 Introduction

Speaker adaptation is but a small cog in the engine of speech recognition, and it is a relatively recent addition to the field of discrete-time processing of speech signals. In order to fully appreciate speaker adaptation it must be placed in the correct context within the recognition framework. Thus, this section introduces a few of the basic concepts of discrete-time speech recognition.

The end goal of modern speech recognition is the recognition of extracted features from the speech waveform using higher-order statistical models. In order to model speech using higher-order statistical models (such as HMMs), speech is assumed to be stationary over a short period of time — typically 10-30 ms. This assumption is known as the quasi-stationary assumption. When the analysis time frame is too short, too little data is available to determine the signal properties, and when the frame is too long, the speech signal varies too much during the time frame and the stationarity assumption becomes void. Speech signals are a combination of semi-periodic, aperiodic and stochastic signals. Using intimate knowledge of the properties of the speech signal it may be represented in a compact form by a sequence of feature vectors. Well-designed features summarise the most important information in the speech signal, while discarding unimportant information and suppressing noise. This focuses the higher-order statistical models on the most important information for the task at hand. Poorly designed features focus the statistical models on unimportant (and possibly performance degrading) signal characteristics.

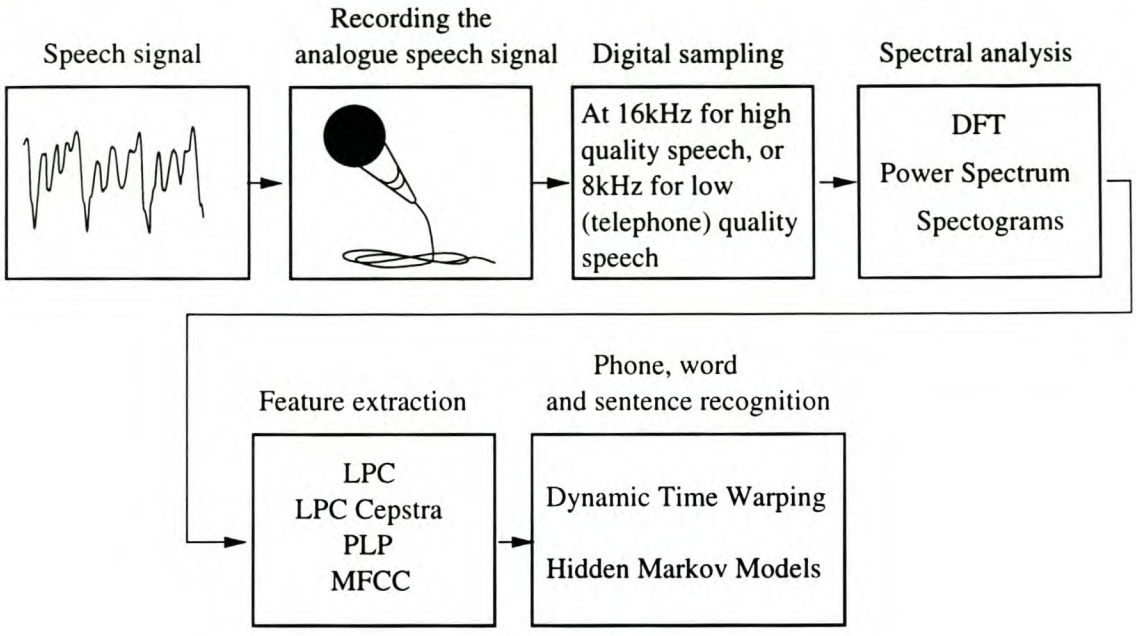

Figure 2.1: Block diagram of the speech recognition process.

Furthermore, the resulting compression reduces the processing and memory requirements for the recognition problem.

Speech recognition may be broken into the following sub-categories:

- **Recording the speech signal:** The goal here is to create a copy of the original speech waveform onto some storage format. The storage may be analogue, such as audio tape, or, if the signal has been sampled, digital, such as CD-ROM or hard disk.

- **Digital sampling of the recorded speech signal:** In order to perform analysis of the speech signal on a digital computer, the recorded speech signal is sampled at twice the Nyquist frequency of speech, and then quantised and stored in digital format. Speech signals may have frequency components as high as 15kHz, but typically the highest frequency components are lower than 8kHz. Sampling at 16kHz is thus sufficient to represent a high quality speech signal. Low quality speech, e.g. speech recorded over a telephone transmission line, may be sampled at a lower frequency. The bandwidth of telephone transmission lines is typically between 3-4 kHz, so sampling at 8kHz is considered to be adequate.

- **Spectral analysis of the digitised speech signal:** Spectral analysis is important for both understanding of the speech waveform as well as being a precursor to many feature extraction methods. Under the quasi-stationary assumption, the speech signal is segmented into short time frames. Spectograms may be calculated using these frames to gain more knowledge of the specific speech signal. Wideband spectograms using short time frames have high time resolution, and are useful for the accurate transcription (i.e. finding the begin and end times for the phonemes or words that were spoken in the digitised speech) of the speech signal. Narrowband spectograms use relatively long time frames and have high frequency resolution, and are useful for accurately determining the formants (resonances in spectrum caused by the shape of the vocal tract, and named formants because they tend to dominate and "form" the overall spectrum) of the time frames. The power spectra and DFTs computed for the time frames are also used for feature extraction purposes.

- **Feature extraction:** Several kinds of feature vectors are available: cepstral coefficients, linear prediction coefficients (LPCs), linear prediction coefficient cepstra (LPCC), perceptual linear prediction (PLP) and Mel-frequency warped cepstral coefficients (MFCCs). In the experiments described in this thesis LPCC and PLP features are exclusively. Therefore LPCC and PLP extraction, as well as the precursors of LPCCs — cepstral coefficients and LPCs — will be treated briefly in Sections 2.2.2—2.2.5.

- **Phoneme, word and sentence recognition:** Hidden Markov models (HMMs) are the current state-of-the-art in statistical modelling for speech recognition. HMMs are also the focus of the most successful speaker-model adaptation techniques, and are treated in Section 2.4.

- **Message understanding:** This constitutes the processing and understanding of the information the speaker intended the speech signal to convey. As it falls beyond the scope of speaker adaptation it will not receive any treatment in this thesis.

Selected topics in the speech recognition process will now be addressed.

2.2 Feature Extraction

Feature extraction is the process of creating compact representations of each frame of the segmented speech signal, so that these features may be used either as a method of storing the speech signal in a highly compressed form or as the sequence of vectors modelled by a higher-order statistical model.

The following feature extraction techniques will be treated: cepstral analysis, linear prediction, cepstral coefficients of the linear prediction spectrum and perceptual linear prediction. In terms of interpretation and derivation of the feature extraction methods, the source-filter model of speech production is of tantamount importance. Thus the feature extraction discussion begins with a description of the source-filter model.

2.2.1 The Source-Filter Model for Speech Production

There are two kinds of basic excitation for a speech signal [9]:

- **Voiced excitation:** A stream of quasi-periodic puffs of air due to the periodic movement of the vocal chords.
- **Unvoiced excitation:** A noise-like excitation caused by the turbulence of air-flow through a narrow constriction.

Along with these are combinations of voiced and unvoiced excitations that are categorised for modelling purposes:

- **Plosive excitation:** The buildup and subsequent release of air pressure behind a completely closed portion of the vocal tract. The release may be voiced or unvoiced.
- **Whisper:** Air is forced through the partially open glottis to excite an utterance.
- **Silence:** Short periods of silence occur between utterances, such as the pause before a plosive sound.

Both unvoiced and voiced sounds may be adequately modelled as the convolution of an excitation signal and the vocal tract impulse response. For voiced sounds, the excitation signal is modelled by a periodic pulse train, and the vocal tract impulse response is

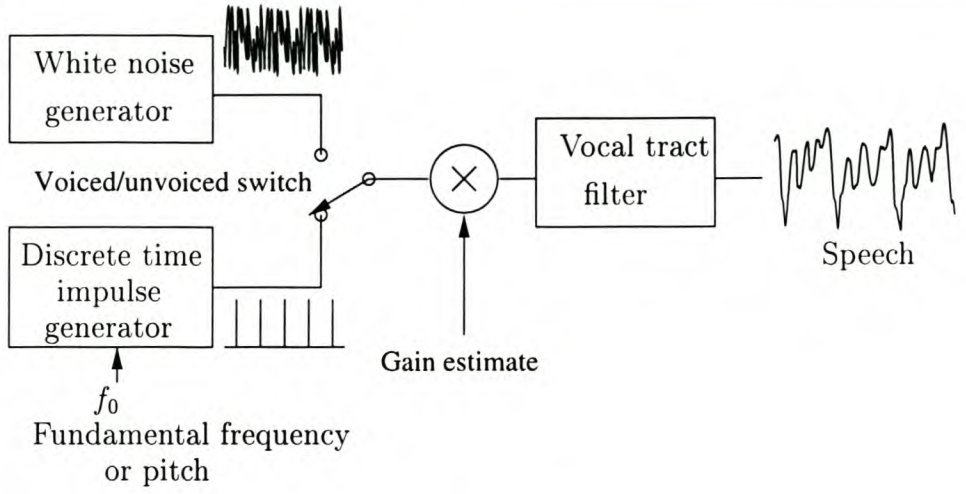

Figure 2.2: The source-filter model for speech modelling.

modelled by a pole-zero filter. Unvoiced sounds are modelled using white noise to represent the excitation signal, and a pole-zero filter function to represent the vocal tract impulse response. These models form what is known as the source-filter model of speech production.

2.2.2 Cepstral Analysis

The spectrum of the vocal tract is of great importance in speech recognition, and so we would like to reconstruct it by removing the effect of the excitation source from the speech signal via filtering. Filtering in the frequency domain, however, can only separate signals that consist of linear combinations of input signals, where the component frequencies of the input signals are separated in the frequency domain. Unfortunately, speech is the result of the convolution of two signals in the time domain, so the speech spectrum is the result of the multiplication of the vocal tract spectrum and excitation spectrum. Thus, filtering in the frequency domain will not separate the vocal tract and excitation spectra, as these signals are not additive in the frequency domain.

In order to "filter" the excitation spectrum from the speech spectrum, we need to transform the spectra into a domain where multiplication becomes addition. Such a domain is the cepstral domain (where the unit is time), where the power cepstrum $c_s(n)$ for a speech signal $s(n)$ is given by the IDTFT of the log of magnitude squared of the

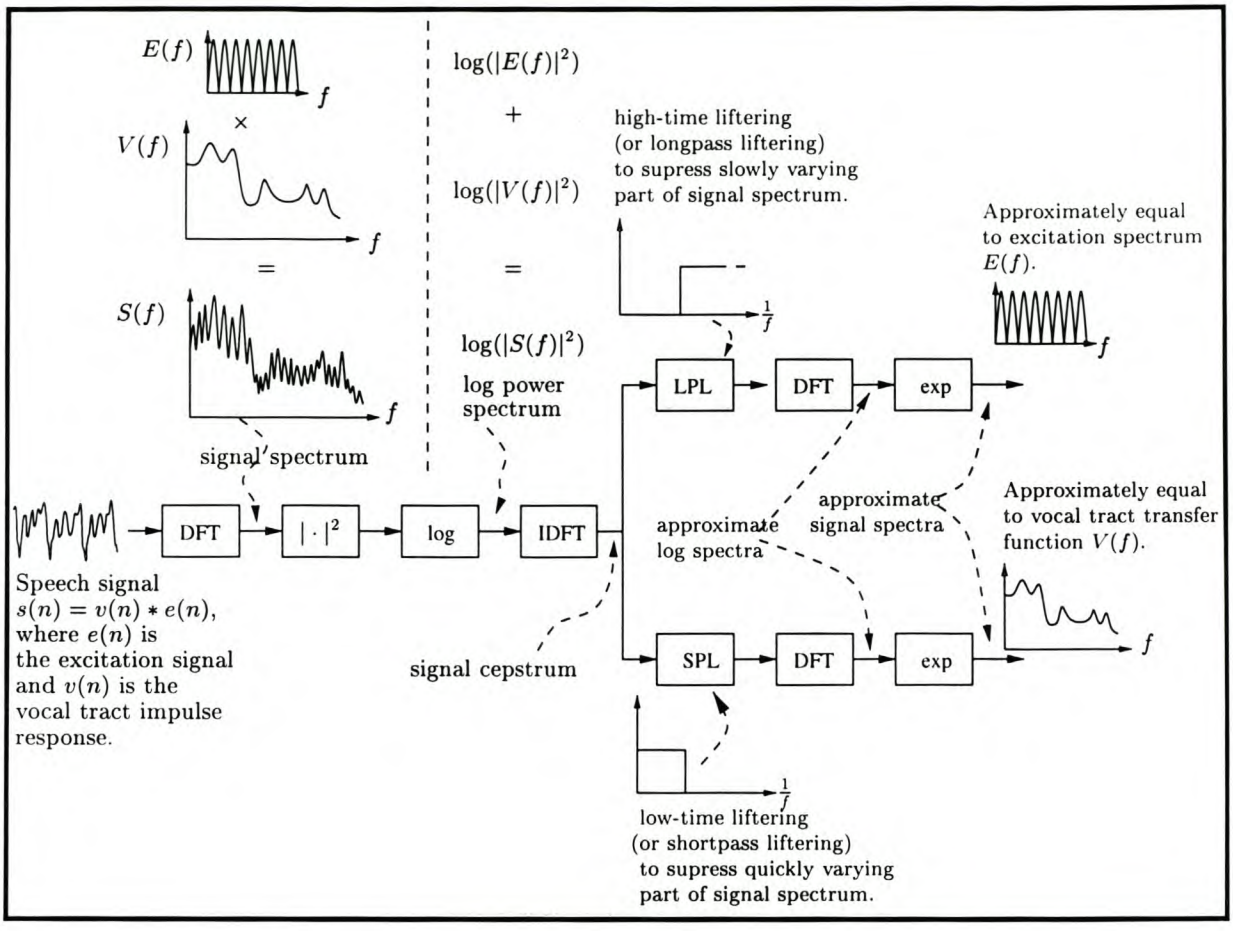

Figure 2.3: A diagram of liftering in the cepstral domain in order to separate the vocal tract power spectrum from the excitation power spectrum (adapted from Niesler [41]).

DTFT for $s(n)$. Thus

$$c_s(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\left\{\log\left|\sum_{l=-\infty}^{\infty} s(l)e^{-j\omega l}\right|^2\right\}e^{j\omega n}d\omega. \tag{2.1}$$

High-time (or longpass) and low-time (or shortpass) liftering — the equivalent to high and low pass filtering in the frequency domain — may now be done in the cepstral domain to separate the excitation cepstrum from the vocal tract cepstrum. The inverse cepstral transform is then applied to obtain the separated vocal tract impulse response — or excitation signal, if it is desired — in the time domain. In the cepstral domain, the vocal tract response is typically less than 5 ms, whereas the first peak in the cepstrum due to the excitation pulse train is in the vicinity of 8 ms. Liftering is thus easily accomplished.

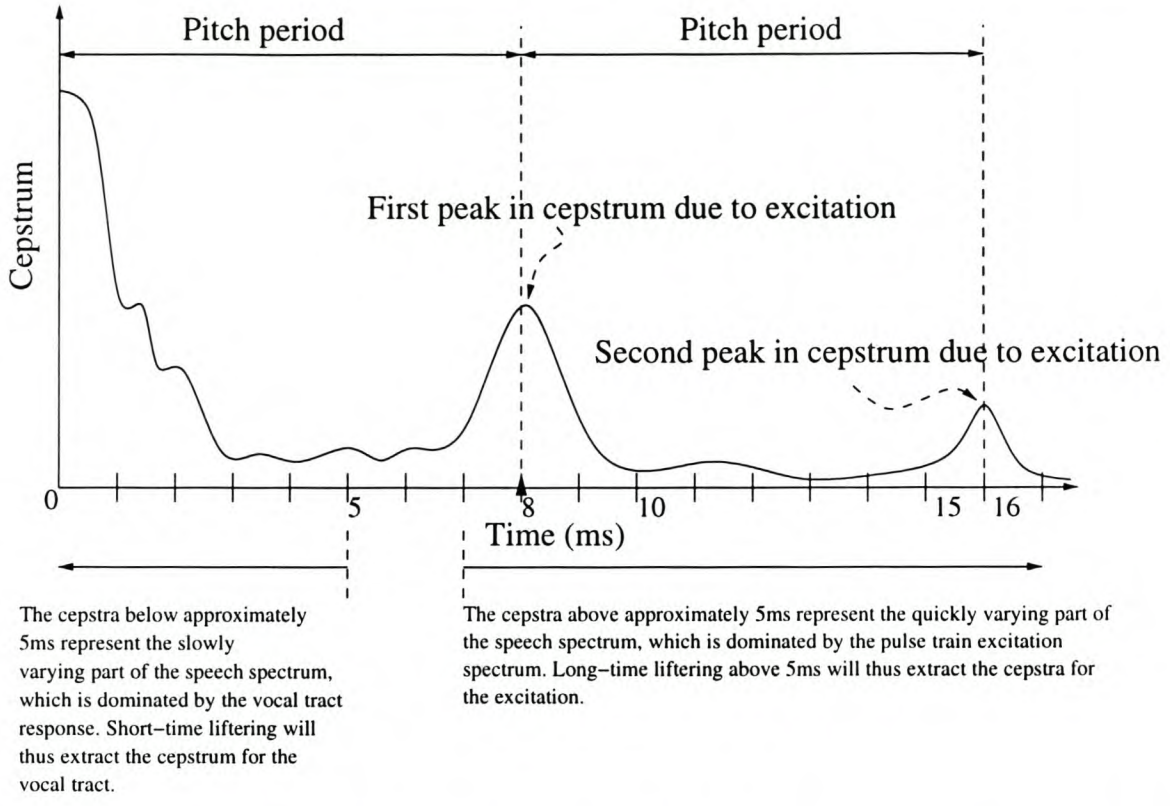

Figure 2.4: A representation of a typical cepstra vs. time plot for voiced speech, indicating the regions of cepstral dominance of the vocal tract and excitation spectra (adapted from Niesler [41]).

The cepstral coefficients $c_s(n)$ describing the vocal tract spectrum may then be combined to form a feature vector for the speech frame $s(n)$. Computing the continuous DTFT and IDTFT is not possible on a digital computer. However, the speech frames are of finite duration, thus the DFT and IDFT instead of the DTFT and IDTFT are used to compute the cepstrum. If the frame is length $N-1$, and the DFT $S(k)$ of signal $s(n)$ is

$$S(k) = \sum_{n=0}^{N-1} s(l) e^{-j(2\pi/N)kn} \quad , \quad k = 0, \ldots, N-1, \tag{2.2}$$

then the computed cepstra $\tilde{c}_s(n)$ are given by

$$\tilde{c}_s(n) = \frac{1}{N} \sum_{k=0}^{N-1} \log |S(k)|^2 \, e^{j(2\pi/N)kn} \quad , \quad n = 0, \ldots, N-1. \tag{2.3}$$

The relationship between the true cepstra and practically computed cepstra is

$$\tilde{c}_s(n) = \begin{cases} \sum_{q=-\infty}^{\infty} c_s(n+qN), & n = 0, \ldots, N-1 \\ 0, & \text{other } n \end{cases} . \tag{2.4}$$

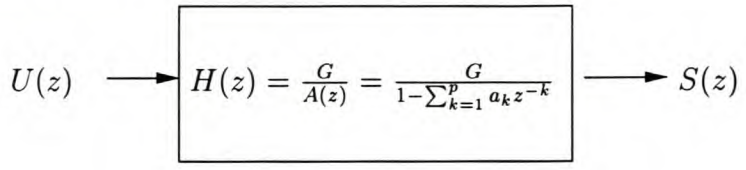

Figure 2.5: A depiction of the transfer function $H(z)$ (the vocal tract transfer function) to be estimated using linear prediction. $U(z)$ is the Z-transformed input signal (the excitation signal for the vocal tract) and $S(z)$ is the Z-transformed output signal (the speech signal). G represents the gain component of $H(z)$, and $A(z)$ represents the minimum-phase all-pole component of $H(z)$.

$\tilde{c}_s(n)$ is thus a periodic, aliased version of $c_s(n)$. Even though this aliasing is not particularly detrimental in speech recognition, it may be reduced by appending zeros (normally factor 10 or less) to the speech frame to increase the effective record length. In practice the speech is either a Hamming or Hanning windowed.

It should be noted that the transform used to obtain the power cepstra is not truly reversible, as the phase information of the original signal is lost when the absolute value of the signal spectrum is taken. Fortunately phase information is generally of little use for speech recognition purposes. However, for purposes such as vocoding where a speech signal is to be generated, the phase-preserving complex cepstra should be used.

2.2.3 Linear Prediction

Linear prediction is a method that can be implemented to obtain an estimate of vocal tract transfer function — much like the implementation of cepstral liftering to obtain an approximate vocal tract transfer function. Linear prediction attempts to model an element in a discrete time sequence $s(n)$ as a weighted sum of prior sequence elements and prior input signal elements. The resulting model may be represented by an *autoregressive moving average*(ARMA) or pole-zero system.

For linear prediction, the form of the vocal tract transfer function (introduced during the discussion of the source-filter representation of speech production in Section 2.2.1) is generally simplified to be an all-pole instead of a pole-zero transfer function. The motivation for this is twofold:

- It is analytically difficult to form pole-zero transfer functions in linear prediction, whereas simple solutions comprising linear equations exist for all-pole models.

- Any causal rational pole-zero system $\Theta(z)$ may be decomposed as

$$\Theta(z) = G\Theta_{\text{min}}(z)\Theta_{\text{ap}}(z), \tag{2.5}$$

where G is a constant gain term, $\Theta_{\text{min}}(z)$ is minimum phase and $\Theta_{\text{ap}}(z)$ is all-pass, i.e. $|\Theta_{\text{ap}}(e^{j\omega})| = 1$, $\forall\omega$. Also, the minimum-phase component may be expressed as an all-pole system. Phase information in the speech signal is relatively unimportant for speech recognition (Deller [9], p. 269-270). As an all-pole model can exactly model a magnitude spectrum, and the information needed for speech recognition lies mostly in the speech magnitude spectrum, linear prediction using an all-pole model is considered sufficient for most speech modelling applications. Thus the system we wish to model has transfer function $H(z) = G/A(z)$ (refer to Figure 2.5), where linear prediction is used to estimate the all-pole component $A(z) = 1 - \sum_{k=1}^{p} a_k z^{-k}$.

When only the prior sequence elements are used to form a signal estimate, it is known as *autoregressive* linear prediction. The resultant transfer function will be all-pole. A p-th order autoregressive linear predictive estimate $\hat{s}(n)$ for signal $s(n)$ at time n is given by

$$\hat{s}(n) = \sum_{k=1}^{p} a_k s(n-k), \tag{2.6}$$

where a_k are the predictor coefficients. The error $e(n)$ between the estimate and the true signal is given by

$$e(n) = s(n) - \hat{s}(n). \tag{2.7}$$

Thus, from Equation 2.6, the relationship between the true signal and the estimate is

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + e(n). \tag{2.8}$$

2.2.3.1 LP Parameter Estimation for a Deterministic Signal Using the Method of Least Squares

In order to obtain the linear prediction parameters for stochastic signals, the likelihood of the squared error, $\mathcal{L}\{e^2(n)\}$, is minimised, whereas for deterministic signals the total

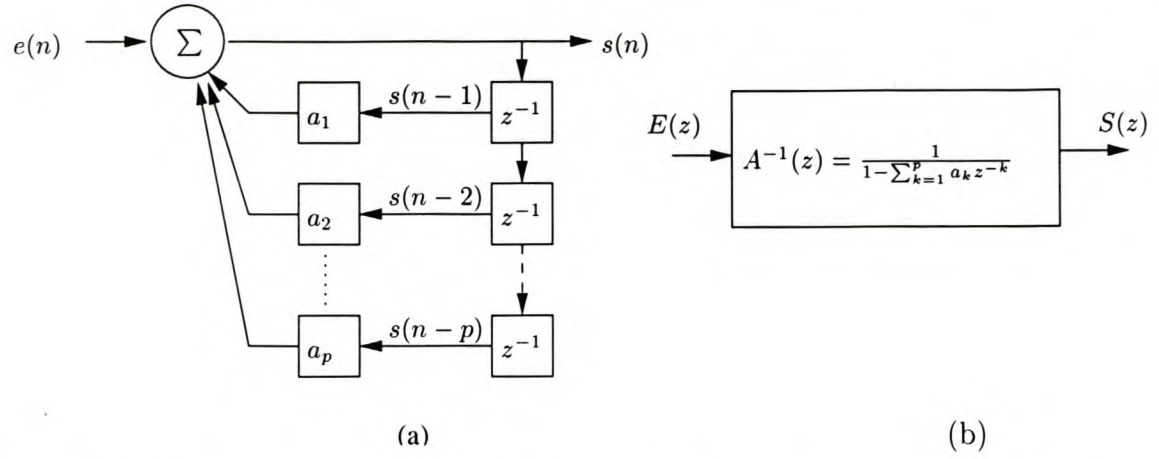

Figure 2.6: (a) Representation of all-pole linear prediction modelling in the time domain. (b) Representation of all-pole linear prediction in the frequency domain.

squared error $\sum_{n=-\infty}^{\infty} e^2(n)$ is minimised. If we denote the total squared error by E, then

$$E = \sum_n e^2(n) = \sum_n \left[s(n) - \sum_{k=1}^{p} a_k s(n-k) \right]^2 . \tag{2.9}$$

E is now minimised with respect to the LP parameters by setting

$$\frac{\partial E}{\partial a_i} = 0, \quad 1 \leq i \leq p. \tag{2.10}$$

From Equations 2.9 and 2.10 the following set of equations is obtained

$$\sum_{k=1}^{p} a_k \sum_n s(n-k)s(n-i) = \sum_n s(n)s(n-i), \quad 1 \leq i \leq p. \tag{2.11}$$

The LP parameters $\{a_i\}_i = 1 \ldots p$ may be obtained by solving the set of equations in Equation 2.11. The range of summation over n is of import, and two methods — the *covariance*and the *autocorrelation*methods — with differing summation ranges are commonly employed. In the covariance method the range of summation is chosen so that the minimisation is over the finite interval $0 \leq n \leq N$, whereas for the autocorrelation method the error is minimised over the interval $-\infty \leq n \leq \infty$. Our choice is to use the autocorrelation method, as it is computationally friendly and it is guaranteed to yield stable LP models.

2.2.3.2 The Autocorrelation Method

Autocorrelation vector and matrix are defined next. The autocorrelation vector $\mathbf{r}$ is defined so that each of its elements $\mathbf{r}_i$ are given by the short-term autocorrelation estimate $r_s(i)$ for an N-point frame of signal $s(n)$, so that

$$\mathbf{r}_i = r_s(i) = \frac{1}{N} \sum_{n=-\infty}^{\infty} s(n)s(n-i). \tag{2.12}$$

Autocorrelation matrix R is defined so that each of its elements $R_{ik} = r_s(i-k)$ are given by

$$R_{ik} = r_s(i-k) = \sum_{n=-\infty}^{\infty} s(n-k)s(n-i). \tag{2.13}$$

For the autocorrelation method, the total squared error E is minimised over the range $-\infty \leq n \leq \infty$, so that Equation 2.11 becomes

$$\sum_{k=1}^{p} a_k \sum_{n=-\infty}^{\infty} s(n-k)s(n-i) = \sum_{n=-\infty}^{\infty} s(n)s(n-i), \quad 1 \leq i \leq p. \tag{2.14}$$

Multiplying both sides by $\frac{1}{N}$, it is found that

$$\sum_{k=1}^{p} a_k \frac{1}{N} \sum_{n=-\infty}^{\infty} s(n-k)s(n-i) = \frac{1}{N} \sum_{n=-\infty}^{\infty} s(n)s(n-i) \tag{2.15}$$

$$\sum_{k=1}^{p} a_k r_s(i-k) = r_s(i), \quad 1 \leq i \leq p, \tag{2.16}$$

or, in the corresponding matrix form,

$$R\mathbf{a} = \mathbf{r}, \tag{2.17}$$

where $\mathbf{a}$ is a vector such that each of its p elements is one of the LP parameters ($\mathbf{a}_i = a_i,\ 1 \leq i \leq p$).

Matrix R is symmetric and Toeplitz. Due to these properties, the very efficient Durbin's algorithm (also known as the Levinson-Durbin or L-D algorithm) can be used to solve for $\mathbf{a}$ in Equation 2.17.

Generally, speech is framed using a windowing function so that the quasi-stationary assumption is valid. A windowed speech frame $s'(n)$ is zero outside some interval $0 \leq n \leq N-1$, so that

$$s'(n) = \begin{cases} s(n)w(n), & 0 \leq n \leq N-1 \\ 0, & \text{elsewhere} \end{cases}, \tag{2.18}$$

where $w(n)$ is the windowing function.

Equations for the elements of the autocorrelation vector (Equation 2.13) and autocorrelation matrix (Equation 2.14) then become

$$\mathbf{r}_i = r_s(i) = \frac{1}{N} \sum_{n=0}^{N-1+i} s(n)s(n-i), \quad i \geq 0, \tag{2.19}$$

and

$$R_{ik} = r_s(i-k) = \sum_{n=0}^{N-1+i-k} s(n-k)s(n-i), \quad i \geq 0,\ k \geq 0, \tag{2.20}$$

respectively.

For the analysis of long time frames, a block windowing function would be appropriate. Speech, however, is highly transient. The time analysis must thus be limited, making a Hamming or Hanning window appropriate.

2.2.3.3 A Few Remarks on Linear Prediction

A few points on linear prediction are stated briefly without any proofs:

- When a vocal tract transfer is estimated from a speech signal using linear prediction, the estimated spectra are generally better fits for the original spectra than the estimates obtained using shortpass liftering in the cepstral domain.

- Linear prediction tends to match the peaks in the signal spectrum better than the rest of the spectrum due to the least-squares criterion. This is of importance for speech processing, as it helps to accurately determine the characteristic resonant peaks or formants in the speech spectrum.

- From an inverse filtering point of view, linear prediction can be described as the process of finding the inverse filter that whitens the signal spectrum, i.e. the post-filtering spectrum is flat. Not surprisingly, the best LP match (in the sense that the total error squared is a minimum) for a signal spectrum is possible for those cases where the true input signal has a flat spectrum. Such input signals include the impulse (Kronecker delta) — a deterministic signal — and stationary white noise — a stochastic signal. From a speech modelling point of view this is advantageous, as the primary excitation (input) signals in speech can be modelled by white noise

(unvoiced speech) and impulse train (voiced speech) functions. LP modelling is thus likely to provide a good estimate of the vocal tract transfer function. It should be noted that the pulse train becomes closer to the ideal input signal as the period of the pulse train goes to infinity. This explains why LP modelling for voiced sounds is generally better for lower pitched (relatively long pulse train period) male-produced speech than higher pitched (relatively short pulse train period) female- or child-produced speech.

- The choice of the linear predictor model order p is not a trivial matter. It is known that as $p \to \infty$ it becomes possible to match any minimum-phase system using linear prediction. This, however, is not practical. We thus seek to find p large enough to give a good estimate for the problem at hand, but not so large that the computational complexity becomes prohibitive.

2.2.4 Linear Prediction Coefficient Cepstra

Research has shown [8] that cepstral parameters formed from the spectrum of an LP model yield better performance than LP parameters. These cepstral coefficients can be computed from the LP coefficients by recursively applying the following equation:

$$c_n = a_n \sum_{m=1}^{n-1} \frac{m}{n} c_m a_{n-m}, \quad 1 \leq n \leq p, \tag{2.21}$$

where a_0 is typically chosen to be the model gain G that can be computed if the input signal is known. It can be shown that [9]

$$G \approx \begin{cases} \sqrt{r_s(0) - \sum_{i=1}^{p} a_k r_s(k)}, & \text{unvoiced case} \\ \sqrt{P\left(r_s(0) - \sum_{i=1}^{p} a_k r_s(k)\right)}, & \text{voiced case} \end{cases}, \tag{2.22}$$

where P is the pitch period.

An LP model spectrum represents a "smoothed" version of the original speech frame spectrum, as it tends to remove the effect of the excitation spectrum. Computing cepstrum for a speech frame has similar smoothing properties. Part of the reason for the better performance of LPCCs is that the LPCC features are essentially a doubly smoothed version of the speech frame spectrum. It is possible to fine tune this smoothing by manipulating the cepstral coefficients. Another reason for the LPCCs' superiority over

LPCs is that the resulting cepstra bear a useful relation to the log spectrum of the LP filter. Suppose $A_1(\omega)$ and $A_2(\omega)$ represent the inverse filters for two different speech frames, and $c_n^{(1)}$ and $c_n^{(2)}$ represent the cepstral sequences computed using the parameters of $A_1(z)$ and $A_2(z)$ respectively. If we ignore the gain for both systems, then

$$\sum_{n=1}^{\infty} \left[c_n^{(1)} - c_n^{(2)}\right]^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left[\log |A_1^{-1}(\omega) - A_2^{-1}(\omega)|\right]^2 d\omega. \tag{2.23}$$

The euclidean distance between two cepstral sequences is thus a good measure of the spectral difference between two LP models.

2.2.5 Perceptual Linear Prediction

Perceptual linear prediction (PLP) extends linear prediction by including three concepts from psychoacoustics of human hearing in the LP framework. These three concepts are:

1. **Spectral resolution of human hearing:** When two tones are close to each another in frequency, the human auditory system can often only perceive the dominant tone, i.e. the tone with the highest power.

2. **Equal-loudness preemphasis:** The human auditory system is more sensitive to frequencies in the midrange of the auditory spectrum.

3. **Intensity-loudness power law:** There is a non-linear relationship between the intensity of sound and the perceived loudness.

Including these three psychoacoustic concepts makes PLP-feature extraction more consistent with what is known of human hearing than LP-feature extraction. A stepwise procedure for calculating PLP features is now given:

1. **Sampling, windowing and spectral analysis:** The procedure of sampling, windowing and spectral analysis is identical to the corresponding steps of LP analysis. The speech is sampled (8kHz for low quality speech, and 16kHz for high quality speech), and windowed (typically a Hamming or Hanning window with a window length of ten to thirty milliseconds). The power spectrum for each windowed sampled speech frame is then calculated.

2. **Spectral Resolution of Human Hearing:** A "critical band" is a frequency range in psychoacoustic studies that relates to the spectral resolution capabilities of human hearing. When two competing tones fall in the same critical band, the only tone that will be perceived is the most powerful tone. However, when two tones are separated in the frequency domain by more than the critical bandwidth, they are both perceived. This perceptual change is abrupt. Furthermore, the critical band is approximately constant for frequencies under 800 Hz, after which it increases logarithmically.

   To compensate for the critical band frequency resolution of human hearing, the frequency scale is first warped to the Bark scale. The Bark scale is a perceptual frequency scale such that one Bark covers one critical band, 24 Bark covers the audible spectrum, and the mapping between the frequency and Bark scales is given by

   $$\Omega\omega = 6 \log \left[\frac{\omega}{1200\pi} + \sqrt{\left(\frac{\omega}{1200\pi}\right)^2 + 1} \right], \tag{2.24}$$

   where ω is frequency and Ω is Bark.

   Making the resolution in the Bark scale equivalent to the resolution of human hearing, the spectrum in the Bark scale is convolved with a critical band filter function. This filter function has a bandwidth of one Bark, i.e. the bandwidth is the same as relevant critical band. Convolving the Bark spectrum with the critical band filter smoothes the power spectrum, removing frequency resolution to the same extent as the human hearing system. As the frequency resolution has been degraded, it is permissible to downsample the power spectrum. Typically 18 samples are taken, where a sample is taken approximately every one Bark.

3. **Equal Loudness Preemphasis:** Human hearing is not equally sensitive to all frequencies, and it is most sensitive to frequencies around 3.5 kHz. Compensation for the non-uniform sensitivity, the spectrum is warped by the following function:

   $$E(\omega) = \frac{\left(\omega^2 + 56.8 \cdot 10^6\right) \omega^4}{\left(\omega^2 + 6.3 \cdot 10^6\right) \left(\omega^2 + 0.38 \cdot 10^9\right) \left(\omega^6 + 9.58 \cdot 10^{26}\right)}. \tag{2.25}$$

   Note that Equation 2.24 is used so that Equation 2.25 may be applied to the downsampled smoother power spectrum (in Bark).

4. **Intensity-Loudness Power Law:** There is a non-linear relationship between the intensity of a sound and the perceived loudness of the sound. To compensate, the following function is applied to the spectrum:

$$L(\omega) = \sqrt[3]{I(\omega)}. \tag{2.26}$$

Once more, Equation 2.24 is used so that Equation 2.26 may be applied in the Bark-warped frequency scale.

5. **LP Analysis:** A loudness-compensated preemphasised downsampled smoothed power spectrum (in Bark) is now available. The IFFT of this power spectrum then yields an autocorrelation, from which a set of LP coefficients are calculated. This set of LP coefficients form the PLP features. Additionally, PLP cepstral coefficients can be calculated from the PLP coefficients in the same way that LPCCs are determined from LPCs.

The computational requirements for determining PLP features is similar to that required for LP analysis. PLP features generally outperform LPC and LPCC features, and has been shown to be more noise-robust than MFCC features (MFCCs, like PLPs, compensate for psychoacoustic effects in human hearing).

2.2.6 Delta Parameters

Delta (or differenced) parameters are often used to extend feature vectors. If $c_n(t)$ is a n-th feature parameter (i.e. the n-th element of feature vector $\mathbf{c}(t)$) for the t-th speech frame, then the delta feature parameter for frame n is given by

$$\Delta c_n(t) = c_n(t+\delta) - c_n(t-\delta), \tag{2.27}$$

where δ is a parameter chosen to smooth the estimate. δ is typically set to one or two, so that the delta parameter is computed by differencing future and past parameters of the feature parameter.

Delta parameters can now be computed for all the feature parameters, and a new feature vector may be formed by appending some or all of the delta parameters to the original feature vector.

2.3 Parameter Optimisation for Statistical Models

This section reviews some of the key estimation methods used for finding optimal parameters for statistical models. The first, *maximum likelihood estimation*(ML estimation), yields an optimal set of parameters that maximises the likelihood of the observed data given the model parameters. The second, the *expectation maximisation algorithm*(EM algorithm), is an iterative maximisation procedure used when not all the data needed to estimate the model parameters are observable. These methods are employed in many ways for parameter estimation of HMMs and the derivation of HMM-based speaker adaptation methods.

2.3.1 Definition of Likelihood

This is based on the definition of likelihood in [7], p. 11.

Let $Y = (Y_1, \ldots, Y_n)$ represent a random process where the pdf of Y belongs to some family $\mathcal{F}$, and let the observations $y = (y_1, \ldots, y_n)$ be realised values of this process. It is not known which pdf in the family is the true pdf of Y. It is often useful to consider how the density changes at the particular observed value y for different functions in $\mathcal{F}$. The ability of a certain pdf to "explain" the particular y is thus analysed, and to emphasise this the *likelihood* of $f(\cdot)$ at y is defined by

$$L\{f(\cdot)|y\} = f(y). \tag{2.28}$$

Equivalently, the natural logarithm of Equation 2.28 can be used. This is called the *log-likelihood*

$$l\{f(\cdot)|y\} = \ln(f(y)). \tag{2.29}$$

2.3.2 Maximum Likelihood Estimation

Many problems in engineering consist of choosing a model to describe a real-world problem, and then finding the optimal (most likely) model parameters that would give the best fit of the proposed model given data from the real-world process. Suppose λ represents a complete set of model parameters in Λ, where Λ is the space spanning all possible sets

of model parameters, and $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ represents a set of observations for the real-world process. In order to find an optimal parameter set, we would like to maximise the probability $P(\lambda|X)$ of the parameter set given the observed data. An optimal estimate $\tilde{\lambda}$ for the set of parameters may thus be found by solving

$$\tilde{\lambda} = \underset{\lambda}{\text{argmax}}\, P(\lambda|X). \tag{2.30}$$

$P(\lambda|X)$ is not generally a readily obtainable function, whereas $P(X|\lambda)$ is. Fortuitously — under certain conditions — maximisation of $P(X|\lambda)$ is equivalent to maximisation of $P(\lambda|X)$. The proof is as follows

$$P(\lambda|X) = \frac{P(\lambda, X)}{P(X)}, \tag{2.31}$$

and

$$P(X|\lambda) = \frac{P(\lambda, X)}{P(\lambda)}. \tag{2.32}$$

Therefore

$$P(\lambda|X) = \frac{P(X|\lambda)P(\lambda)}{P(X)}. \tag{2.33}$$

Clearly, the λ that maximises the left side will also maximise the right side, so that

$$\tilde{\lambda} = \underset{\lambda}{\text{argmax}}\, P(\lambda|X) = \underset{\lambda}{\text{argmax}}\, P(X|\lambda)P(\lambda). \tag{2.34}$$

For the special case where the prior pdf of the parameter sets, $P(\lambda)$, is flat, i.e.

$$P(\lambda) = \text{constant}, \tag{2.35}$$

the following holds:

$$\tilde{\lambda} = \underset{\lambda}{\text{argmax}}\, P(\lambda|X) = \underset{\lambda}{\text{argmax}}\, P(X|\lambda). \tag{2.36}$$

The maximisation of the likelihood of the data given the model parameters under the assumption of equal *a priori* parameter set probabilities is termed *maximum likelihood* estimation. Instead of maximising the likelihood, the log-likelihood may be maximised, as the logarithmic function is monotonically increasing. Thus

$$\tilde{\lambda} = \underset{\lambda}{\text{argmax}}\, P(X|\lambda) = \underset{\lambda}{\text{argmax}}\, \ln\left[P(X|\lambda)\right]. \tag{2.37}$$

Maximising the log-likelihood is often the simpler way of determining the optimal λ, as the family of exponential pdfs is often employed in engineering solutions. For the sake of simpler notation, the likelihood and log-likelihood functions are often used to denote the relevant probabilities, where

$$L_x(\lambda) = P(X|\lambda) \tag{2.38}$$

is the likelihood function, and

$$l_x(\lambda) = \ln P(X|\lambda) \tag{2.39}$$

is the log-likelihood function.

As an example, the ML estimate of a mean vector for a Gaussian pdf given a set of observations will now be derived.

The Gaussian pdf $b(\mathbf{x})$ is given by

$$b(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}\,|C|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu)'C^{-1}(\mathbf{x}-\mu)}, \tag{2.40}$$

where n is the dimension of each observation vector, and μ and C are respectively the mean vector and covariance matrix for the Gaussian pdf. If the observations $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ are assumed to be independent and identically distributed (iid), that is $P(X) = P(\mathbf{x}_1, \ldots, \mathbf{x}_T) = P(\mathbf{x}_1)P(\mathbf{x}_2)\ldots P(\mathbf{x}_T)$, then the ML estimate $\tilde{\mu}$ of the mean vector is obtained by solving

$$\begin{aligned}
\tilde{\mu} &= \underset{\mu}{\text{argmax}}\; P(X|\mu) = \underset{\mu}{\text{argmax}}\; \ln\left[P(X|\mu)\right] \\
&= \underset{\mu}{\text{argmax}}\; \ln\left[\prod_{t=1}^{T} P(\mathbf{x}_t|\mu)\right] \\
&= \underset{\mu}{\text{argmax}} \sum_{t=1}^{T} \ln\left[P(\mathbf{x}_t|\mu)\right] \\
&= \underset{\mu}{\text{argmax}} \sum_{t=1}^{T} \ln\left[\frac{1}{(2\pi)^{n/2}\,|C|^{1/2}}\right] - \frac{1}{2}(\mathbf{x}_t - \mu)'\,C^{-1}\,(\mathbf{x}_t - \mu)\,.
\end{aligned} \tag{2.41}$$

The optimal estimate for μ is the one that maximises $\ln\left[P(X|\mu)\right]$, and it is found by taking the derivative of $\ln\left[P(X|\mu)\right]$ with respect to μ, setting the result equal to zero, and solving for μ. We thus proceed by taking the derivative of Equation 2.41 with respect to μ

$$\frac{\partial}{\partial\mu}\left\{\sum_{t=1}^{T} \ln\left[\frac{1}{(2\pi)^{n/2}\,|C|^{1/2}}\right] - \frac{1}{2}(\mathbf{x}_t - \mu)'\,C^{-1}\,(\mathbf{x}_t - \mu)\right\},$$

moving the derivative inside the summation over t and noting that the first term does not depend on μ (and therefore has a zero derivative)

$$\begin{aligned}
&= \sum_{t=1}^{T} \left\{ 0 - \frac{1}{2} \frac{\partial}{\partial \mu} (\mathbf{x}_t - \mu)' C^{-1} (\mathbf{x}_t - \mu) \right\} \\
&= \sum_{t=1}^{T} \left\{ 0 - \frac{1}{2} \frac{\partial}{\partial \mu} \left(\mathbf{x}_t' C^{-1} \mathbf{x}_t - 2\mu' C^{-1} \mathbf{x}_t + \mu' C^{-1} \mu \right) \right\} \\
&= \sum_{t=1}^{T} \left\{ -\frac{1}{2} \left(-2\mathbf{x}_t' C^{-1} + 2\mu' C^{-1} \right) \right\} \\
&= \sum_{t=1}^{T} \left\{ \mathbf{x}_t' C^{-1} - \mu' C^{-1} \right\}. \qquad (2.42)
\end{aligned}$$

Setting the result of the differentiation in Equation 2.42 equal to zero and solving for μ, the familiar ML estimate of the mean vector of a Gaussian pdf (assuming the observations are iid) is found:

$$\tilde{\mu} = \frac{\sum_{t=1}^{T} \mathbf{x}_t}{T}. \qquad (2.43)$$

2.3.2.1 Sufficient Statistics

Any real or vector valued function $\mathbf{t}(X)$ of the data X is known as a *statistic*. Statistics are *sufficient* if they supply all the information necessary to estimate the model parameters. A sufficient statistic thus "summarises" the data, such that only the statistic is necessary for the reestimation of parameters, and not the entire original data set. The concepts of sufficient statistics may be expressed as follows: Statistic $\mathbf{t}(X)$ is sufficient if there are functions (note that these are not pdfs) $g(\mathbf{t}(X), \lambda)$ and $h(X)$ such that

$$P(X|\lambda) = g(\mathbf{t}(X), \lambda)\, h(X). \qquad (2.44)$$

Maximising $g(\mathbf{t}(X), \lambda)$ with respect to λ will yield the same parameters as maximising $P(X|\lambda)$ with respect to λ. This is readily seen if the log-likelihood $\ln[P(X|\lambda)]$ is maximised instead of the likelihood $P(X|\lambda)$, as

$$\ln P(X|\lambda) = \ln[g(\mathbf{t}(X), \lambda)] + \ln[h(X)]. \qquad (2.45)$$

Thus

$$\begin{aligned}
\frac{\partial}{\partial \lambda} P(X|\lambda) &= \frac{\partial}{\partial \lambda} \{\ln[g(\mathbf{t}(X), \lambda)]\} + \frac{\partial}{\partial \lambda} \{\ln[h(X)]\} \qquad (2.46) \\
&= \frac{\partial}{\partial \lambda} \{\ln[g(\mathbf{t}(X), \lambda)]\} + 0. \qquad (2.47)
\end{aligned}$$

2.3.2.2 The Exponential Family of Distributions

The exponential family of distributions include all pdfs of the form

$$f(X|\lambda) = b(X)e^{[\mathbf{c}'(\lambda)\mathbf{t}(X)]}/a(\lambda), \tag{2.48}$$

where $\mathbf{t}(X)$ is the sufficient statistic of the family. This family of distributions is often employed as it offers a wide variety of distributions of engineering interest, such as the Gaussian, Poisson, Rayleigh, uniform and binomial distributions. ML estimation for all of the distributions belonging to the exponential family may be simplified, as only the sufficient statistic of each is necessary for the estimation of model parameters.

2.3.3 The Expectation-Maximisation Algorithm

In order to perform maximum likelihood estimation, all the data needed to estimate a parameter is required. In many pattern recognition problems, however, all the necessary data is not always available. Often only part of the data is observed, while another part remains hidden. The EM (*expectation-maximisation*) algorithm is well suited to tackling parameter estimation problems where a part of the complete data needed to compute an ML estimate is not directly observable.

Let $\mathcal{Y}$ be the sample space of observable data, and $\mathbf{y} \in R^m$ is an observation from $\mathcal{Y}$. Similarly, let $\mathcal{X}$ denote the full underlying space, and let $\mathbf{x} \in R^n$ be an outcome of $\mathcal{X}$, with $n \geq m$. $\mathbf{x}$ is referred to as the *complete data*. There is a many-to-one mapping $\mathbf{y} = y(\mathbf{x})$ between the complete data and the observable data. Thus, any observation $\mathbf{y}$ represents a subset of $\mathcal{X}$, denoted by $\mathcal{X}(\mathbf{y})$.

Let λ represent a set of parameters in Λ, the space containing all possible parameter sets. If the pdf of the complete data given the model parameters is $f(\mathbf{x}|\lambda)$, then the pdf of the incomplete data is

$$g(\mathbf{y}|\lambda) = \int_{\mathcal{X}(\mathbf{y})} f(\mathbf{x}|\lambda)d\mathbf{x}. \tag{2.49}$$

Equation 2.49 may be worded as follows: the pdf of the observable data $\mathbf{y}$ is equal to the integral of the pdf of the complete data over that region of $\mathcal{X}$ represented by $\mathbf{y}$.

To form an ML estimate of λ given the data, we need to maximise the log-likelihood $\ln[g(\mathbf{y}|\lambda)]$ (also referred to as the log-likelihood of the observed data, $l_y(\lambda)$) with respect

to λ. An ML estimate for λ using $l_y(\lambda)$ will not have an analytic solution in general; $l_x(\lambda)$ (i.e. $\ln[f(\mathbf{x}|\lambda)]$), however, normally has a well-defined analytically solvable maximum. Finding the maximum of the log-likelihood function $l_x(\lambda)$ for the complete data is not possible, as the complete data $\mathbf{x}$ is not known (the complete data is not observed directly, but only by means of $\mathbf{y}$). Direct maximisation of $\ln[f(\mathbf{x}|\lambda)]$ is thus impossible. The EM algorithm surmounts the problem by iteratively maximising the expectation of $\ln[f(\mathbf{x}|\lambda)]$ given the observable data $\mathbf{y}$ and the current estimate of λ. By taking the estimation of the likelihood given the observable data and current set of model parameters, we are "completing" the observable data and estimating the complete data.

The EM algorithm owes its name to the fact that each iteration can be split into two steps: an *expectation* step, and a *maximisation* step. If $\lambda^{[k]}$ is the estimate of the parameter set at the k-th iteration, then the two steps can be formulated as:

- **The expectation step:** Here we compute the desired expectation, denoted by $Q\left(\lambda|\lambda^{[k]}\right)$, where

$$\begin{aligned} Q\left(\lambda|\lambda^{[k]}\right) &= E\left[\ln f(\mathbf{x}|\lambda)|\mathbf{y}, \lambda^{[k]}\right] && (2.50) \\ &= \int \left[\ln f(\mathbf{x}|\lambda)\right] f(\mathbf{x}|\mathbf{y}, \lambda^{[k]}) d\mathbf{x}. && (2.51) \end{aligned}$$

  The second argument $\lambda^{[k]}$ of the Q function conditions the expectation, and is considered fixed. The first argument is the conditioning argument for the complete data log-likelihood $f(\mathbf{x}|\lambda)$, and it is the parameter with respect to which the log-likelihood will be maximised.

- **The maximisation step:** Let $\lambda^{[k+1]}$ be the value of λ that maximises the Q function. Thus

$$\lambda^{[k+1]} = \underset{\lambda}{\operatorname{argmax}}\, Q\left(\lambda|\lambda^{[k]}\right). \qquad (2.52)$$

  The estimate $\lambda^{[k+1]}$ will be used as the current parameter set for the expectation step of the next iteration, i.e. iteration $k+2$, of the algorithm.

The EM algorithm thus consists of choosing an initial parameter set $\lambda^{[k]}$, and then performing expectation and maximisation steps iteratively until convergence. Convergence

may be determined by examining the distance between successive parameter estimates and stopping when the parameter sets become closer than ϵ. Thus we stop iterating when

$$\left\|\lambda^{[k]} - \lambda^{[k+1]}\right\| < \epsilon, \tag{2.53}$$

where $\|\cdot\|$ is a appropriate distance measure and ϵ is an suitable error distance between two successive parameter sets.

The EM algorithm has the following properties:

- The maximisation of $Q\left(\lambda|\lambda^{[k]}\right)$ at each iteration yields an increase in $l_y(\lambda)$, the log-likelihood of the observed data given the model parameters.
- $l_y(\lambda)$ is non-decreasing at every iteration. That is

  $$l_y\left(\lambda^{[k+1]}\right) \geq l_y\left(\lambda^{[k]}\right), \tag{2.54}$$

  with equality only when $\lambda^{[k]}$ is a stationary point (eg. a global maximum, local maximum, saddle point or other point) of $l_y(\lambda)$.

The EM algorithm thus guarantees that the sequence of parameter estimates $\lambda^{[0]}, \lambda^{[1]}, \ldots, \lambda^{[k]}$ results in a bounded non-decreasing sequence of log-likelihoods $l_y\left(\lambda^{[0]}\right) \leq l_y\left(\lambda^{[1]}\right) \leq \ldots \leq l_y\left(\lambda^{[k]}\right)$ which must converge as $k \to \infty$. [1] A complete proof of the above can be found in [10, 52].

2.3.3.1 Specialisation of the EM Algorithm for Distributions of the Exponential Family

The EM algorithm can be simplified for the exponential family, as each of the distributions in this family has a well defined sufficient statistic. The expectation and maximisation steps of the EM algorithm reduce to the following:

- **The expectation step:**

  $$Q\left(\lambda|\lambda^{[k]}\right) = E\left[\ln b(\mathbf{x})|\mathbf{y}, \lambda^{[k]}\right] + \mathbf{c}(\lambda)' E\left[\mathbf{t}(\mathbf{x})|\mathbf{y}, \lambda^{[k]}\right] - \ln a(\lambda). \tag{2.55}$$

  In deriving the above it should be noted that the expectation $E\left[\ln a(\lambda)|\mathbf{y}, \lambda^{[k]}\right]$ is simply $\ln[a(\lambda)]$, so that no expectation computation is necessary. The term

[1] The convergence for the EM algorithm is approximately that of a steepest decent algorithm [52]. Convergence of the EM algorithm may be accelerated by employing conjugate gradient methods [21].

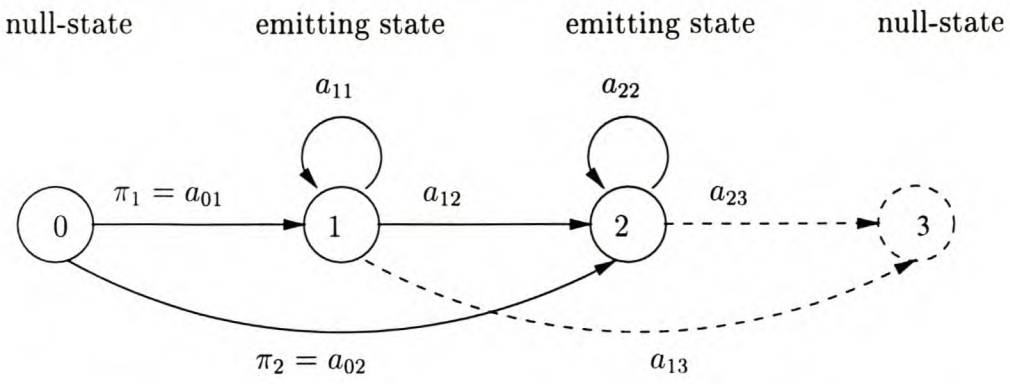

Figure 2.7: A representation of a typical HMM. States 1 and 2 are the emitting states. State 0 is the beginning null-state, and state 3 is the optional ending null-state. The state transition probabilities are labelled by a_{ij}.

$E\left[\ln b(\mathbf{x})|\mathbf{y}, \lambda^{[k]}\right]$ will not play a role when maximising with respect to λ in the maximisation step, and is thus not computed. Only the expectation in the second term of Equation 2.55 needs to be computed. Denoting this expectation by $\mathbf{t}^{[k+1]}$, the entire expectation step reduces to determining

$$\mathbf{t}^{[k+1]} = E\left[\mathbf{t}(\mathbf{x})|\mathbf{y}, \lambda^{[k]}\right]. \tag{2.56}$$

- **The maximisation step:** The new parameter estimate $\lambda^{[k+1]}$ is now given by

$$\lambda^{[k+1]} = \underset{\lambda}{\operatorname{argmax}} \left\{\mathbf{c}(\lambda)'\mathbf{t}^{[k+1]} - \ln\left[a(\lambda)\right]\right\}. \tag{2.57}$$

2.4 Modelling Speech with Hidden Markov Models

A hidden Markov model (HMM) is a stochastic model of a process that changes with time. The ability to model a quasi-stationary random process makes HMMs ideal to model features extracted from a speech signal, and has resulted in HMMs becoming the most popular model for speech recognition tasks. An HMM[2] consists of a fixed number of states (represented by the encircled 0, 1, and 2 in Figure 2.7), where each emitting state (states 1 and 2 for Figure 2.7) has a pdf associated with it. The states are connected via state transition probabilities (the arrows labeled a_{ij} in Figure 2.7), i.e. the probability of making

[2]Continuous density first-order Moore forms of HMMs are treated in this text, as speaker adaptation techniques to date have focused on this form.

a transition from one state to another. An HMM is thus a set of pdfs, interconnected by links, where each link represents the probability associated with making a transition from one pdf to another. An HMM thus models a time-changing stochastic process as a time-changing sequence of states. At any time, the properties of the process is modelled by the pdf of one of the states. When time moves on, and the properties of the signal change, the most likely state to represent the process for the new time is selected. This new state is selected according to the cost of making a transition to it from the previous state, and the match of the state pdf with the attributes of the process. Note that the most likely state may be the same as the previous state.

The symbols denoting HMM structures and parameters, as well as the symbols for the observation sequence from the process will now be introduced:

- T is the length of the observation sequence.
- N is the number of states.
- $Q = \{q_1, q_2, \ldots, q_N\}$ are the states.
- $A = \{a_{ij}\}$, $a_{ij} = P(q_j \text{ at } t+1 | q_i \text{ at } t)$ is the state transition probability, i.e. the probability being in state j at time $t+1$ given that the previous state at time t was state i. The transition probabilities must always satisfy $\sum_{j=1}^{N} a_{ij} = 1$.
- $B = \{b_j(\mathbf{o})\}$, $b_j(\mathbf{o}) = P(\mathbf{o}|q_j)$ is the state pdf, i.e. the probability of state j generating observation $\mathbf{o}$.
- $\pi = \{\pi_i\}$, $\pi_i = P(q_i \text{ at } t = 1)$ is the probability of beginning in state q_i. The set of initial probabilities $\{\pi_1, \ldots, \pi_N\}$ is known as the initial state distribution.

$\lambda = (\pi, A, B)$ will be used as a compact representation of an HMM and its parameter set. As previously stated, HMMs are used to model observation sequences. HMMs may also be used to generate observation sequences. To elucidate the working of an HMM, the steps necessary to generate an observation sequence $O = \mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T$ will now be stated:

1. An initial state i_1 is chosen according to the initial state distribution, π. The "time" t is set to $t = 1$.

2. An observation $\mathbf{o}_t$ is chosen according to the state pdf $b_{i_t}(\mathbf{o})$ for state i_t.

3. The next state in the sequence, state i_{t+1} at time $t+1$, is chosen according to the state transition probabilities from state i_t, $a_{i_t i_{t+1}}$.

4. The time t is incremented by one, and if $t < T$ we return to step 2, if not, then the procedure is terminated, as the observation sequence of length T has been generated.

In the above, all the states associated have state pdfs, and are referred to as emitting states. Instead of using a separate set of π_i parameters for the initial state distribution, a null-state (non-emitting state) with no pdf may be prepended to the HMM (state 0 in Figure 2.7). The transition probabilities from state 0 to state i are then equivalent to the initial state distribution, so that $\{a_{01} = \pi_1,\ a_{02} = \pi_2,\ \ldots,\ a_{0N} = \pi_N\}$. The addition of a beginning null-state will not alter any of the derivations, and is simply an alternate representation. It should be noted that an ending null-state (such as state 3 in Figure 2.7) is sometimes appended to an HMM, so that set of states is $\{q_0, \ldots, q_{N+1}\}$, and states q_0 and q_{N+1} are null-states. The transition probability from a penultimate state to this final state explicitly models the probability of ending in the penultimate state. Adding an ending null-state will slightly change the equations derived for the likelihood computation, segmentation and reestimation of the HMMs, as the final transition probability must be included.

HMMs are generally used to solve three kinds of problems:

- Likelihood computation: Compute the probability of an observation sequence O given a set of model parameters, λ, i.e. compute the likelihood $L_o(\lambda) = P(O|\lambda)$. This likelihood is important for recognition and scoring purposes. For instance: A set of HMMs are used to model a set of phonemes, with each HMM representing one phoneme. An observation sequence is then recorded, and we must decide which phoneme was uttered. The likelihood of the observation sequence for each HMM is then computed, and the decision of which phoneme was uttered is made according to which HMM had the highest likelihood.

- Segmentation: Given an observation sequence $O = \mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T$, we must find the state sequence $\theta = i_1, i_2, \ldots, i_T$ most likely to have generated it must be found. Such a state sequence is useful for studying model behaviour.

- Reestimation: Find a set of parameters λ to maximise $P(O|\lambda)$, so that good models for speech signals may be trained. Due to the nature of HMMs, the state sequence forms the hidden data that must be estimated in order to form EM estimates of the HMM parameters. The dependency of reestimation on the hidden state information means that segmentation and reestimation go hand in hand for HMMs.

Two key algorithms — the Viterbi and Baum-Welch algorithms — useful for computations of all three problems will be introduced before a discussion of the problems.

2.4.1 The Baum-Welch Algorithm

This algorithm has several uses, namely determining the likelihood of the observation data given the model parameters, determining state-occupation probabilities and obtaining values necessary for the estimation of HMM parameters. The Baum-Welch procedure may be split into a forward iteration and a backward iteration. The forward iteration will be treated first. The forward variable $\alpha_t(i)$ is introduced, where

$$\alpha_t(i) = P(\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_t, i_t = q_i|\lambda). \tag{2.58}$$

The forward variable is thus the likelihood of the partial observation sequence up unto time t and being in state q_i at time t given the model parameters. The forward variable may be solved inductively as follows:

1. Initialise the first set of forward variables at the first time step, $t = 1$.

$$\alpha_1(i) = \pi_i b_i(\mathbf{o}_1), \quad 1 \leq i \leq N. \tag{2.59}$$

2. Compute $\alpha_{t+1}(j)$ iteratively for the remaining time steps.

$$\begin{aligned} \text{For } t &= 1, 2, \ldots, T-1, \\ \alpha_{t+1}(j) &= \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(\mathbf{o}_{t+1}), \quad 1 \leq j \leq N, \end{aligned} \tag{2.60}$$

3. Determine the likelihood of the observation sequence.

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{2.61}$$

Only N^2T calculations are required to determine $P(O|\lambda)$ (as opposed to $2TN^T$ for straightforward computation — see Section 2.4.3, Equation 2.75). The efficiency of the Baum-Welch algorithm is due to the fact that there are only N states, and therefore only N possible places for path to emerge at time t. Each forward variable $\alpha_t(i)$ represents the summed likelihood of all possible path sections up to time t going through state i at time t - in effect every forward parameter is a "summary" of the likelihood of all possible path sections up to time t. As we work with the "summary" of path section likelihoods, it is not necessary to calculate all possible paths individually, which would be computationally expensive. Calculating the next set of forward variables (representing the next set of summed likelihoods of all possible paths going through a state at time $t+2$) is accomplished using the simple update in Equation 2.60. The final likelihood is the sum of the forward variables for time T, i.e. the sum of all the sums of likelihoods of all possible paths ending in states $1..N$ at time T. In a similar fashion to the forward iteration, a backward variable $\beta_t(i)$ is defined for the backward iteration, so that

$$\beta_t(i) = P(\mathbf{o}_{t+1}, \mathbf{o}_{t+2}, \ldots, \mathbf{o}_T | i_t = q_i, \lambda). \tag{2.62}$$

As was the case with the forward variable, the backward variable may now be solved inductively:

1. Initialise the backward variables at the final time step, $t = T$.

$$\beta_T(i) = 1, \quad 1 \leq i \leq N. \tag{2.63}$$

2. Iteratively determine the backward variables for the remaining time steps.

$$\begin{aligned} \text{For } t &= T-1, T-2, \ldots, 1, \\ \beta_t(i) &= \sum_{j=1}^{N} a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j), \quad 1 \leq i \leq N, \end{aligned} \tag{2.64}$$

The backward iteration, like the forward iteration, only requires in the order of N^2T calculations.

2.4.2 The Viterbi Algorithm

The Viterbi algorithm is used to find the optimal state sequence θ for an HMM, as well as obtaining values necessary for HMM parameter reestimation. The algorithm bears great

resemblance to the forward iteration of the Baum-Welch procedure (see Section 2.4.1). Viterbi segmentation relies on the fact that at each point in time t, there are only N (the amount of emitting states) possible outcomes for path sequences, i.e. all possible path sequences up until time t must terminate in one of the N states. The algorithm will now be given in stepwise fashion, followed by an explanation of the idea:

1. Initialise at the first time step, $t = 1$.

$$\delta_1(i) = \pi_i b_i(\mathbf{o}_1), \quad 1 \leq i \leq N \tag{2.65}$$

$$\psi_1(i) = 0. \tag{2.66}$$

2. Iteratively compute of $\delta_t(i)$ and $\psi_t(i)$ for $2 \leq t \leq T$:

$$\text{For } t = 2, 3, \ldots, T$$

$$\delta_t(i) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(\mathbf{o}_t), \quad 1 \leq j \leq N \tag{2.67}$$

$$\psi_t(i) = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}], \quad 1 \leq j \leq N. \tag{2.68}$$

   $\psi_t(j)$ represents the state at time $t-1$ most likely to have preceded state j at time t. $\delta_t(i)$ is the likelihood of the most probable state sequence of all state sequences ending in state i at time t.

3. Terminate the iteration. Determe the most likely final state and the probability associated with it.

$$\tilde{P} = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{2.69}$$

$$\tilde{i}_T = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_T(i)]. \tag{2.70}$$

4. Determine the optimal path (state sequence) by backtracing from the optimal state at time T, $\tilde{i}_T$.

$$\text{For } t = T-1, T-2, \ldots, 1$$

$$\tilde{i}_t = \psi_{t+1}(\tilde{i}_{t+1}). \tag{2.71}$$

   The optimal state sequence is then $\tilde{\theta} = \tilde{i}_1, \tilde{i}_2, \ldots, \tilde{i}_T$.

In the above algorithm the most likely state sequence is determined. This is done by initialising all possible beginnings of state sequences. Note that the best state sequence ending in state j at time t can only come from one of the optimal state sequences ending at time $t-1$, and not one of the sub-optimal state sequences ending at time $t-1$. At every time step t and every state i it is only necessary to compute the best state sequence ending in state i at time t, while the remaining sub-optimal state sequences ending in state i at time t may be discarded. Using the iterative procedure the optimal final state for the sequences ending at time T is then determined, and by backtracking through the previously most likely states the optimal state sequence is obtained.

The Viterbi and Baum-Welch Algorithms have now been treated, and so we move on to the treatment of the three HMM problems.

2.4.3 Likelihood Computation

As previously stated, determining which HMM is more likely to have generated an observation sequence is useful for phoneme, word and other speech recognition tasks.

We wish to determine the likelihood of the data (observation sequence) given the current set of model parameters, $P(O|\lambda)$. To determine this likelihood, $P(O|\theta,\lambda)$ is first determined, where $\theta = i_1, i_2, \ldots, i_T$ is a fixed state sequence.

$$P(O|\theta,\lambda) = b_{i_1}(\mathbf{o}_1)b_{i_2}(\mathbf{o}_2)\ldots b_{i_T}(\mathbf{o}_T). \tag{2.72}$$

The probability for a fixed state sequence I given parameter set λ is

$$P(\theta|\lambda) = \pi_{i_1}a_{i_1i_2}a_{i_2i_3}\ldots a_{i_{T-1}i_T}. \tag{2.73}$$

Thus the likelihood of the model generating the observation sequence O and state sequence θ simultaneously is $P(O,\theta|\lambda) = P(O|\theta,\lambda)P(\theta|\lambda)$. Determining $P(O|\lambda)$ is now a simple matter of summing $P(O,\theta|\lambda)$ over all possible state sequences. Therefore

$$P(O|\lambda) = \sum_{\theta\in\Theta} P(O|\theta,\lambda)P(\theta|\lambda) \tag{2.74}$$

$$= \sum_{i_1,i_2,\ldots,i_T} \pi_{i_1}b_{i_1}(\mathbf{o}_1)a_{i_1i_2}b_{i_2}(\mathbf{o}_2)\ldots a_{i_{T-1}i_T}b_{i_T}(\mathbf{o}_T), \tag{2.75}$$

where Θ is the set of all possible state sequences. Computation of the likelihood using Equation 2.75 is computationally expensive — about $2TN^T$ calculations are required.

Therefore, $P(O|\lambda)$ is normally determined using the computationally efficient Baum-Welch procedure.

2.4.4 Segmentation

Several segmentations of an HMM for a given observation sequence are possible. We might want to know which state is the most likely to have occurred at a certain time, or we might need the state-occupation probabilities at a certain time. We might also wish to find the single best state sequence, and not simply the sequence of those states that were most likely at each time step. For each of these, application of the Viterbi or Baum-Welch algorithms may be used to obtain the desired segmentation. The algorithms to use in order to obtain a few of the segmentations will now be stated.

- **Obtaining an optimal state sequence:** If Θ is the set of all possible state sequences, and θ is a state sequence in Θ, then we wish to find the optimal θ that maximises $P(O, \theta|\lambda)$. The Viterbi algorithm may be employed to find this optimal state sequence.

- **Obtaining the state occupation probabilities:** The state occupation probability $\gamma^{(i)}(t)$ is the probability of being in state i at time t, and it may be expressed as

$$\gamma^{(i)}(t) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)}, \tag{2.76}$$

  where the forward variable $\alpha_t(i)$, backward variable $\beta_t(i)$ and the likelihood of the observation sequence given the model parameters $P(O|\lambda)$ are all obtained using the Baum-Welch algorithm.

- **Obtaining "hard" state occupation probabilities:** The segmentation of an HMM by state occupation probabilities using the Baum-Welch algorithm, is referred to as "soft" segmentation. Segmentation using the Viterbi algorithm, where a decision is made as to which state was most likely to have occurred at a specific time, is referred to as "hard" segmentation. Using the optimal state sequence $\tilde{\theta}$ from the Viterbi algorithm, a set of hard state occupation probabilities may be defined

such that

$$\gamma^{(i)}(t) = \begin{cases} 1, & \text{if } \tilde{\theta}_t = i \\ 0, & \text{if } \tilde{\theta}_t \neq i. \end{cases} \tag{2.77}$$

The most commonly used state pdfs for HMMs in speech processing are the single Gaussian and mixture Gaussian pdfs. As such, it is often desirable to obtain mixture occupation probabilities for reestimation purposes. Possible segmentations to obtain mixture occupation probabilities will be discussed after stating the form of a mixture Gaussian pdf (or GMM — Gaussian mixture model).

The mixture Gaussian $f^{(i)}(\mathbf{o})$ used to represent the state pdf for state i is given by

$$f^{(i)}(\mathbf{o}) = \sum_{m \in i} c_m^{(i)} b_m^{(i)}(\mathbf{o}), \tag{2.78}$$

where i is the state, m a mixture component in the GMM (mixture Gaussian or Gaussian mixture model) of state i, $c_m^{(i)}$ is the weight of mixture component m and $b_m^{(i)}(\mathbf{o})$ is the Gaussian pdf for mixture component m in state i. Each mixture component is a Gaussian pdf, $b_m^{(i)}(\mathbf{o})$, given by

$$b_m^{(i)}(\mathbf{o}) = \frac{1}{(2\pi)^{n/2} \left| C_m^{(i)} \right|^{1/2}} e^{-\frac{1}{2}\left(\mathbf{o}-\mu_m^{(i)}\right)' C_m^{(i)-1} \left(\mathbf{o}-\mu_m^{(i)}\right)}. \tag{2.79}$$

where n is the dimension of observation vector $\mathbf{o}$, and $\mu_m^{(i)}$ and $C_m^{(i)}$ are the mean vector and covariance matrix of the Gaussian pdf.

The following definitions of mixture occupation probability may now be made:

- **Mixture occupation probabilities for Baum-Welch segmentation:** The mixture occupation probability $\gamma_m^{(i)}(t)$ for mixture component m in state s at time t is given by

$$\gamma_m^{(i)}(t) = \gamma^{(i)}(t) \frac{c_m^{(i)} b_m^{(i)}(\mathbf{o})}{f^{(i)}(\mathbf{o})}. \tag{2.80}$$

- **Mixture occupation probabilities for Viterbi segmentation:** The mixture occupation probability $\gamma_m^{(i)}(t)$ for mixture component m in state s at time t is again given by

$$\gamma_m^{(i)}(t) = \gamma^{(i)}(t) \frac{c_m^{(i)} b_m^{(i)}(\mathbf{o})}{f^{(i)}(\mathbf{o})}, \tag{2.81}$$

where $\gamma^{(i)}(t)$ is the state occupation probability obtained via the Viterbi algorithm as defined in Equation 2.77. Nguyen [40] refers to this as the *semi-Viterbi mode* of obtaining mixture occupation probabilities.

2.4.5 Reestimation of HMM parameters

There is no direct ML estimation for HMM parameters given an observation sequence. As such, we must resort to using gradient techniques, or the EM algorithm treated in Section 2.3.3. As stated in Section 2.3.3, the EM algorithm is used to maximise $P(X|\lambda)$ in cases where the complete data needed to estimate parameters are not available. The EM algorithm first "completes" the data by estimating the complete data given the observed data and current set of model parameters. For HMMs, the observed data X is the observation sequence O, and the hidden (missing) data is the state sequence θ that generated the observation sequence. For HMM reestimation using the EM algorithm, the state sequence is estimated first, and then using the estimated sequence maximum likelihood estimates for a new set of HMM parameters are formed. Multiple iterations of the EM algorithm will converge to a stationary point on $P(O|\lambda)$.

The Baum-Welch algorithm was created specifically for the EM reestimation of HMM parameters, and the reestimation formulas obtained using Baum-Welch segmentation and reestimation are stated in the next section.

Alternatively, Viterbi reestimation of parameters may be used, where Viterbi segmentation is used to obtain the optimal state sequence, which is used to complete the data and estimate the new parameters. Viterbi reestimation represents the EM algorithm for maximising $P(O, \tilde{\theta}|\lambda)$, where $\tilde{\theta}$ is the optimal state sequence. Though this is not the "true" EM maximisation of $P(O|\lambda)$, Viterbi reestimation is often employed in practice as it requires much less computation than Baum-Welch reestimation, and the recognition results are similar for the hidden Markov model structures commonly used for speech modeling. The Viterbi reestimation formulas will be stated in the section following the treatment of the Baum-Welch reestimation formulas.

2.4.5.1 Baum-Welch Reestimation of Parameters

First we introduce $\xi_t(i,j)$, where

$$\xi_t(i,j) = P(q_i = \theta_t, q_j = \theta_{t+1}|O,\lambda). \tag{2.82}$$

$\xi_t(i,j)$ is thus the probability of being in state i at time t, and of making the transition to state j at time $t+1$ given the observation sequence O and the current set of model parameters λ. Another expression for $\gamma^{(i)}(t)$ in terms of $\xi_t(i,j)$ is now possible:

$$\gamma^{(i)}(t) = \sum_{j=1}^{N} \xi_t(i,j). \tag{2.83}$$

The Baum-Welch reestimation formulas for π and A are then

$$\tilde{\pi}_i = \gamma^{(i)}(1), \quad 1 \leq i \leq N, \tag{2.84}$$

and

$$\tilde{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma^{(i)}(t)}. \tag{2.85}$$

For the case of mixture Gaussian state pdfs, the reestimation equations for the mixture component weights $c_m^{(i)}$, mean vectors $\mu_m^{(i)}$ and covariance matrices $C_m^{(i)}$ are

$$\tilde{c}_m^{(i)} = \frac{\sum_{t=1}^{T} \gamma_m^{(i)}(t)}{\sum_{l=1}^{M} \sum_{t=1}^{T} \gamma_l^{(i)}(t)} = \frac{\sum_{t=1}^{T} \gamma_m^{(i)}(t)}{\sum_{t=1}^{T} \gamma^{(i)}(t)}, \tag{2.86}$$

and

$$\tilde{\mu}_m^{(i)} = \frac{\sum_{t=1}^{T} \gamma_m^{(i)}(t)\mathbf{o}_t}{\sum_{t=1}^{T} \gamma_m^{(i)}(t)}, \tag{2.87}$$

and

$$\tilde{C}_m^{(i)} = \frac{\sum_{t=1}^{T} \gamma_m^{(i)}(t)[\mathbf{o}_t - \mu_m^{(i)}][\mathbf{o}_t - \mu_m^{(i)}]'}{\sum_{t=1}^{T} \gamma_m^{(i)}(t)}, \tag{2.88}$$

where $\gamma^{(i)}(t)$ is the state occupation probability obtained via the Baum-Welch algorithm, given by Equation 2.76, and $\gamma_l^{(i)}(t)$ is the mixture occupation probability, also obtained via the Baum-Welch algorithm, given by Equation 2.80.

2.4.5.2 Viterbi Reestimation of Parameters

First, the hidden data must be estimated, and so the optimal state sequence $\tilde{\theta}$ is computed using Viterbi segmentation. While computing $\tilde{\theta}$ during the backtracking procedure of the segmentation process, it is necessary to keep track of the following tallies:

1. n_{ij}, the number of transitions between states i and j in the optimal state sequence.
2. n_i, the number of transitions from state i to any state in the optimal state sequence.

The reestimation formula for the state transition probabilities a_{ij} is then

$$\tilde{a}_{ij} = \frac{n_{ij}}{n_i}. \tag{2.89}$$

Since the optimal state sequence has been determined, we also automatically have Viterbi state occupation probabilities $\gamma^{(i)}(t)$ as defined by Equation 2.77, and the mixture occupation probabilities $\gamma_m^{(i)}(t)$ defined by Equation 2.81. Using these occupation probabilities, the reestimation formulas for mixture Gaussian state pdfs are

$$\tilde{c}_m^{(i)} = \frac{\sum_{t=1}^{T} \gamma_m^{(i)}(t)}{\sum_{i=1}^{M} \sum_{t=1}^{T} \gamma_l^{(i)}(t)} = \frac{\sum_{t=1}^{T} \gamma_m^{(i)}(t)}{\sum_{t=1}^{T} \gamma^{(i)}(t)}, \tag{2.90}$$

for the mixture component weights, and

$$\tilde{\mu}_m^{(i)} = \frac{\sum_{t=1}^{T} \gamma_m^{(i)}(t) \mathbf{o}_t}{\sum_{t=1}^{T} \gamma_m^{(i)}(t)}, \tag{2.91}$$

for the mean vector of Gaussian mixture component m, and

$$\tilde{C}_m^{(i)} = \frac{\sum_{t=1}^{T} \gamma_m^{(i)}(t)[\mathbf{o}_t - \mu_m^{(i)}][\mathbf{o}_t - \mu_m^{(i)}]'}{\sum_{t=1}^{T} \gamma_m^{(i)}(t)}, \tag{2.92}$$

for the covariance matrix of Gaussian mixture component m. It is noted that the Viterbi reestimation formulas for mixture Gaussian state pdf parameters are identical to the Baum-Welch reestimation formulas, but for the fact that the Viterbi or "hard" state occupation probabilities are used here.

2.5 The Likelihood Function and Auxiliary Function

One of the advantages common to all the HMM reestimating speaker adaptation techniques treated in this thesis, is that they all fit into the same likelihood frame as normal

HMM reestimation. In order to reestimate the HMM parameters, these adaptation techniques all use the EM algorithm. As such, they rely on Baum-Welch or Viterbi segmentation as the expectation step of the EM algorithm, to estimate the hidden state sequence information. The maximisation step, however, is specific to the adaptation method. An auxiliary function can be defined such that an increase of the auxiliary function is equivalent to an increase of the likelihood function. This section is dedicated to introducing the likelihood function and subsequent auxiliary function common to the derivation of all treated speaker adaptation techniques.

2.5.1 Defining the Auxiliary Function

The available adaptation data, O, is a set of T observation vectors:

$$O = \mathbf{o}_1, \mathbf{o}_2 \dots \mathbf{o}_T. \tag{2.93}$$

The current set of HMM parameters is λ and the set of adapted model parameters is $\hat{\lambda}$. Θ represents the set of all possible state sequences (length T) given the data O. The likelihood that the current model generates the observed data O, is then given by

$$L(O|\lambda) = \sum_{\theta \in \Theta} L(O, \theta|\lambda). \tag{2.94}$$

In the above, $L(O, \theta|\lambda)$ is the likelihood of generating the sequence of observation vectors, O, using state sequence θ and given the current set of model parameters λ. During adaptation, the objective function to maximise is $L(O|\lambda)$.

If we view the mixture components as a parallel branch in the state sequence, Equation 2.94 can be written as

$$L(O|\lambda) = \sum_{\theta \in \Theta} \sum_{\zeta \in \Psi_\theta} L(O, \theta, \zeta|\lambda), \tag{2.95}$$

where Ψ_θ is the set of all possible mixture component sequences given the state sequence θ, and ζ is a mixture component sequence in Ψ_θ. An auxiliary function $Q(\lambda, \hat{\lambda})$, given by

$$Q(\lambda, \hat{\lambda}) = \sum_{\theta \in \Theta} \sum_{\zeta \in \Psi_\theta} L(O, \theta, \zeta|\lambda) \log \left[L(O, \theta, \zeta|\hat{\lambda}) \right] \tag{2.96}$$

is introduced at this point. Model parameters that maximise the auxiliary function also increase the value of the objective function $L(O|\lambda)$, and by repeating the procedure of

maximising the auxiliary function and updating the model parameters with the reestimated parameters $\hat{\lambda}$, the objective function is iteratively maximised[3] [4, 35, 24].

2.5.2 Further Expansion and Manipulation of the Auxiliary Function $Q(\lambda, \hat{\lambda})$

The pdf for every state of the HMM is a mixture Gaussian $f(\mathbf{o})$ given by

$$f^{(s)}(\mathbf{o}) = \sum_{m \in s} c_m^{(s)} b_m^{(s)}(\mathbf{o}) \tag{2.97}$$

where s is the state, m a mixture component in the GMM of state s, and $c_m^{(s)}$ is the weight of mixture component m. The likelihood of the observation sequence using the state and mixture sequences and given the current model parameters can now be expanded to

$$L(O, \theta, \zeta|\lambda) = \pi_{\theta_0} \prod_{t=1}^{T} a_{\theta_{t-1}\theta_t} c_{\zeta_t}^{(\theta_t)} b_{\zeta_t}^{(\theta_t)}(\mathbf{o}_t) \tag{2.98}$$

where π_i is the probability that the HMM begins in state i, and a_{ij} is the state transition probability from state i to state j for the HMM.

Now 2.98 is substituted for $L(O, \theta, \zeta|\hat{\lambda})$ in the auxiliary equation.

$$Q(\lambda, \hat{\lambda}) = \sum_{\theta \in \Theta} \sum_{\zeta \in \Psi_\theta} L(O, \theta, \zeta|\lambda) \log \left\{ \pi_{\theta_0} \prod_{t=1}^{T} a_{\theta_{t-1}\theta_t} c_{\zeta_t}^{(\theta_t)} b_{\zeta_t}^{(\theta_t)}(\mathbf{o}_t) \right\} \tag{2.99}$$

$$= \sum_{\theta \in \Theta} \sum_{\zeta \in \Psi_\theta} L(O, \theta, \zeta|\lambda) \tag{2.100}$$

$$\times \left\{ \log \pi_{\theta_0} + \sum_{t=1}^{T} \log a_{\theta_{t-1}\theta_t} + \sum_{t=1}^{T} \log c_{\zeta_t}^{(\theta_t)} + \sum_{t=1}^{T} \log b_{\zeta_t}^{(\theta_t)}(\mathbf{o}_t) \right\}$$

The speaker adaptation methods treated focus on the reestimation of the mean vectors exclusively,[4] while all the other HMM parameters remain unchanged. Thus, only the last term in the { } braces containing the Gaussian pdf which is dependent on the mean vectors

[3] The convergence of this algorithm for improving $L(O|\lambda)$ using the maximisation of $Q(\lambda, \hat{\lambda})$ was first proved by Baum (in 1972) and later extended to mixture pdfs and vector observations.

[4] MAP adaptation need not be on mean vectors only, but can be applied on all HMM parameters [18]. MLLR-like maximum likelihood linear transformations that have the ability to adapt covariance matrices as well as mean vectors were derived by Gales [15]. Though eigenvoice techniques have thus far focused only on the adaptation of mean vectors [40, 28, 51], R. Kuhn et al. [28] state that eigenvoices may be used to model all parameters in an HMM.

will be of importance for the purposes of maximisation. The first term containing the initial state probabilities, the second term containing the state transition probabilities and the third term containing the mixture component weights may be treated as a constant term so that

$$Q(\lambda, \hat{\lambda}) = \text{constant} + \sum_{\theta \in \Theta} \sum_{\zeta \in \Psi_\theta} L(O, \theta, \zeta | \lambda) \sum_{t=1}^{T} \log b_{\zeta_t}^{(\theta_t)}(\mathbf{o}_t) \tag{2.101}$$

The constant term can henceforth be removed from the equation of the auxiliary function, as it will have no influence on the maximisation procedure. The state occupation probability $\gamma^{(s)}(t)$ and mixture occupation probability $\gamma_m^{(s)}(t)$ will now be defined. If S is the set of all state distributions in the system, and $\gamma^{(s)}(t)$ is the *a posteriori* probability of being in state s at time t given that observation sequence O was generated, then

$$\gamma^{(s)}(t) = \frac{1}{L(O|\lambda)} \sum_{\theta \in \Theta} L(O, \theta_t = s | \lambda) \tag{2.102}$$

If M_s is the set of all mixture components in state s, then $\gamma_m^{(s)}(t)$ is the *a posteriori* probability of being in mixture component m of state s at time t given that observation sequence O was generated.

$$\gamma_m^{(s)}(t) = \frac{1}{L(O|\lambda)} \sum_{\theta \in \Theta} \sum_{\zeta \in \Psi_\theta} L(O, \theta_t = s, \zeta_t = m | \lambda) \tag{2.103}$$

$$= \gamma^{(s)}(t) \frac{c_m^{(s)} b_m^{(s)}(\mathbf{o}_t)}{\sum_{r \in s} c_r^{(s)} b_r^{(s)}(\mathbf{o}_t)} \tag{2.104}$$

Equation 2.101 can now be written as (recall that the constant term has been omitted)

$$Q(\lambda, \hat{\lambda}) = L(O|\lambda) \sum_{s=1}^{\mathrm{S}} \sum_{m=1}^{M_s} \sum_{t=1}^{T} \gamma_m^{(s)}(t) \log \left[b_m^{(s)}(\mathbf{o}_t) \right]$$

$$= -\frac{1}{2} L(O|\lambda) \sum_{s=1}^{\mathrm{S}} \sum_{m=1}^{M_s} \sum_{t=1}^{T} \gamma_m^{(s)}(t) \left[n \log(2\pi) + \log |C_m^{(s)}| + h_m^{(s)}(\mathbf{o}_t) \right]$$

where

$$h_m^{(s)}(\mathbf{o}_t) = \left(\mathbf{o}_t - \hat{\mu}_m^{(s)} \right)' C_m^{(s)-1} \left(\mathbf{o}_t - \hat{\mu}_m^{(s)} \right) \tag{2.105}$$

Equation 2.105 represents the furthest specialisation of the auxiliary function (defined by Equation 2.96) common to the derivations of the treated speaker adaptation methods. Any further refinement depends on the adaptation method in question, therefore further

discussion on the auxiliary function and maximisation thereof is suspended until the relevant MLLR and MLED[5] sections. Consult Sections 4.3.2 and 5.3.2 for the MLLR and MLED auxiliary functions respectively.

2.6 HMM Design for Phoneme Recognition

Completing EM iterations to train an HMM guarantees that a local maximum of the likelihood function of the HMM given the data will be achieved, but not that that this maximum will be the global maximum. For this reason, the effectivity of the EM algorithm is determined in large by the initial values of the HMM as well as constraints placed on the HMM model.

Ergodic HMMs have a large number of parameters, and the consequent freedom of the model increases the complexity of the likelihood function — complexity that can greatly increase the number of ill-suited local maxima. Simply employing an ergodic HMM (refer to Figure 2.9) to model phonemes will almost certainly result in a poorly estimated model. A way to avoid poor estimation due to entrapment in local minima is to employ any prior knowledge of what the model structure should be. This places a relevant constraint on the EM algorithm and will help to attain higher local maxima.

The sound of a phoneme changes with time, and so the feature vectors form a highly coherent progression through time. Thus, a desirable characteristic in the HMM will be that the state progression be forced to be from left to right in the model, making skipping back to previous states impossible.

Restrictions of the left-to-right HMM may be relaxed somewhat by allowing forward skips between states. This makes allowances for phonemes that were pronounced faster than usual, or where the speaker does not pronounce some of the states in the progression.

This thesis's model of choice was a left-to-right HMM with a skip width of one or more. Skip width was one for five-state (including beginning and ending null-states) HMMs, and was increased if the number of states was higher. An example of an eight-state, double

[5]MAP is not derived in full, though the same auxiliary function is employed when adaptation is restricted to mean vectors. WP depends on the maximisation of the above in order to obtain ML estimates of mean vectors, but not for the derivation of the WP reestimation equations.

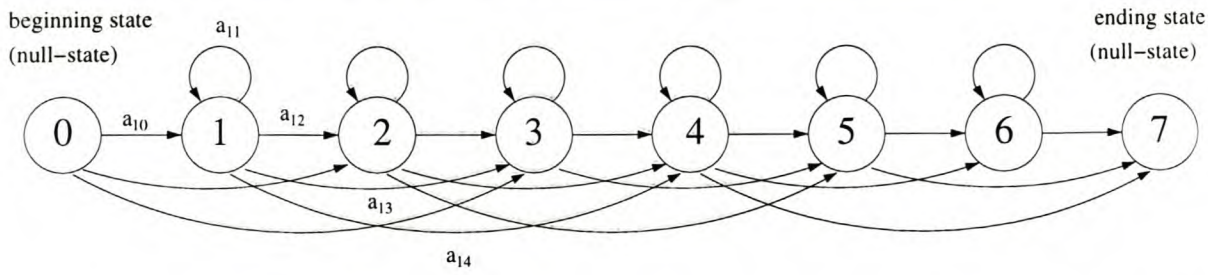

Figure 2.8: An example of an eight-state, left-to-right, double skip width HMM used to model a phoneme.

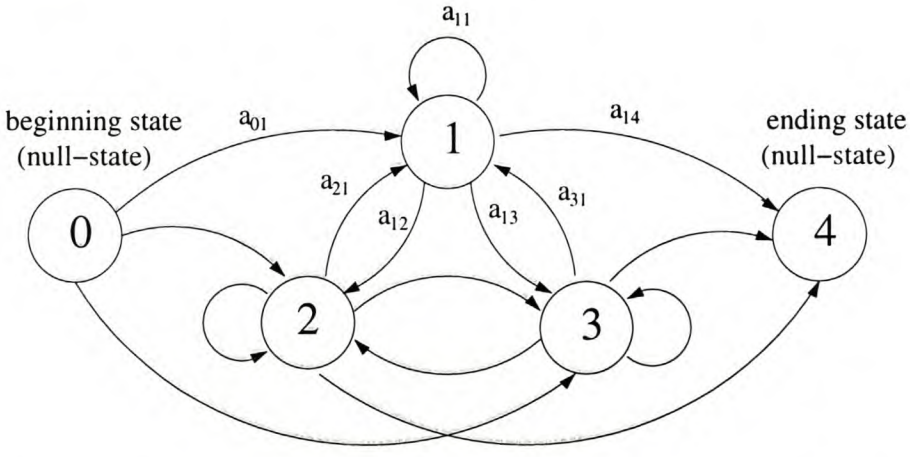

Figure 2.9: An example of the HMMs used to model silence.

skip width HMM is depicted in Figure 2.8.

In order to ensure that features are not passed along and bunched up at the penultimate state of the HMM during training, the initial loop-back probability (a_{11} for state 1) is set to 0.7. Transitions to all other states were then set to equal each other.

Phonemes are well modelled by a left-to-right scheme; silence, however, is not. Silence is for the most part noise, and as such it does not display the same forward progression of phonic phases as phonemes do. Little is known about the temporal dynamics of noise, and in the absence of relevant prior knowledge the model of choice reverts to an ergodic HMM. An example of a typical ergodic model used in this thesis is depicted in Figure 2.9. The only constraints placed are that the loop-back probability for each state be 0.5 and all other transitions departing from a state were made equal. Both speech corpora used for experimental purposes (ISOLET and TIMIT) were high-quality recordings, and most

of the silence encountered in them would consist of low-power white noise.

Sometimes a great deal of data might well be extracted from the silence sections, for instance milieu noise (e.g. traffic) recorded when a cellular phone user is not speaking. Some of the distinctive qualities of this noise could be used to design a more appropriate HMM structure, and this should improve the recognition of silent sections. Knowing what parts of a recording consist of silence is important, as observation data for these parts can be employed for noise cancellation in the spoken sections, resulting in an increase in the recognition of the spoken sections.

2.7 Reducing the Impact of Local Maxima During HMM Training

Cranking a randomly initialised HMM through Viterbi or Baum-Welch training algorithms will probably achieve a highly sub-optimal local maximum for the likelihood function. It is often easier think in terms of the negative log-likelihood function ($-\log L(O|\lambda)$), where the local maxima of the likelihood function become local minima situated in bowls in the negative log-likelihood function. Once stuck in a bowl, retraining using the same data will only cause deeper entrenchment. Using more data (if available) is also unlikely to make much difference, as it will probably only shift the parameters slightly and remain in the same bowl, or — if the previous data set was lopsided — the new estimate might shift the parameters enough to land in a local minimum near the first minimum. Local minima can gobble up endless amounts of data with little or no improvement in the model estimate. All that will happen is that a more robust estimate of parameters for a poor local minima will be achieved. Calculating the likelihood function (or better yet, the recognition performance) of the HMM for all possible values of all its parameters using tiny increments for these parameter values is probably the only true protection against local minima. Such a calculation is not feasible, as even a very small HMM will have a wealth of parameters, and each of these will have to be stepped through a great number of fine increments. How big the number and how fine the increments should be is yet another problem.

One method of defense against local maxima lies in appropriate incorporation of prior

knowledge into the training procedure, as the prior knowledge can restrain the training to an area of HMM parameters likely to yield high local maxima for the likelihood function. Selecting an appropriate model structure (see Section 2.6) is one method of restraining training to useful regions of the parameter space. This, however, only initialises the structure of the HMM, and not the parameter values of the underlying state pdfs. How can those and other parameters effectively be initialised? What should be done if the model desired is an HMM where the number of Gaussians in each mixture pdf is very large? Such complex pdfs are not easily initialised by making an educated guess at the beginning of the training procedure.

The answer lies in breaking the training procedure up into small steps, each step relying on prior knowledge supplied by the results of the previous step. A simple step-wise algorithm was designed to address the problem. The algorithm is given below in pseudocode, after which the design philosophy will be stated (note that the algorithm is only valid for HMMs with GMM state pdfs).

◇ *for each of the HMM models to be trained*

- ◇ *choose the initial model structure for the HMM, as dictated by prior knowledge of the problem to model*
- ◇ *choose the type of initial pdf for each of the states in the HMM.*
- ◇ *choose the number of component pdfs in the mixture pdf that will eventually be used in each state as the state pdf The more the data, the more components may be robustly estimated.*
- ◇ *choose Viterbi or Baum-Welch as a reestimator for the HMM, and the maximum iterations for it to use during an EM procedure.*
- ◇ *gather all the training utterances for the HMM.*
- ◇ *for each utterance*
  - ◇ *make* **s** *the number of states that are not null-states.*
  - ◇ *segment the feature vectors into* **s** *equal-sized chronological data blocks.*
  - ◇ *assign the first segment to the first state, and the following*

segments to their corresponding states.

◇ *for each non-nullstate*

◇ *determine the pdf parameters using the ML estimate of all the data assigned to the state.*

(the state pdf parameters have now been robustly initialised, and the state transition probabilities are taken from the initial model structure.)

◇ *use all utterances, and employ the reestimator (Viterbi or Baum-Welch) in EM iterations, and train until convergence or maximum iteration count is reached.*

◇ *while the number of mixture components is less than the desired number of components*

◇ *for each utterance*

◇ *given the utterance, determine the state occupation probability for the new model, and assign feature vectors to states according to the state occupation probability for the feature vector sequence.*

◇ *increase the number of component pdfs for the mixture pdf in each state by 1, and let* $\mathbf{m}$ *denote the new number of pdf components.*

◇ *for each state*

◇ *segment the feature vectors assigned to the state into* $\mathbf{m}$ *parts by using the k-means algorithm.*

◇ *using the segmentation of the data, determine the ML estimate of each of the component pdfs and their weights in the mixture.*

◇ *use all utterances, and employ the reestimator (Viterbi or Baum-Welch) in EM iterations, and train until convergence or maximum interaction count is reached.*

◇ *save the completed model.*

As stated previously, when using the EM algorithm, achieving a maximum for the likelihood function is possible, but this maximum is not guaranteed to be the global

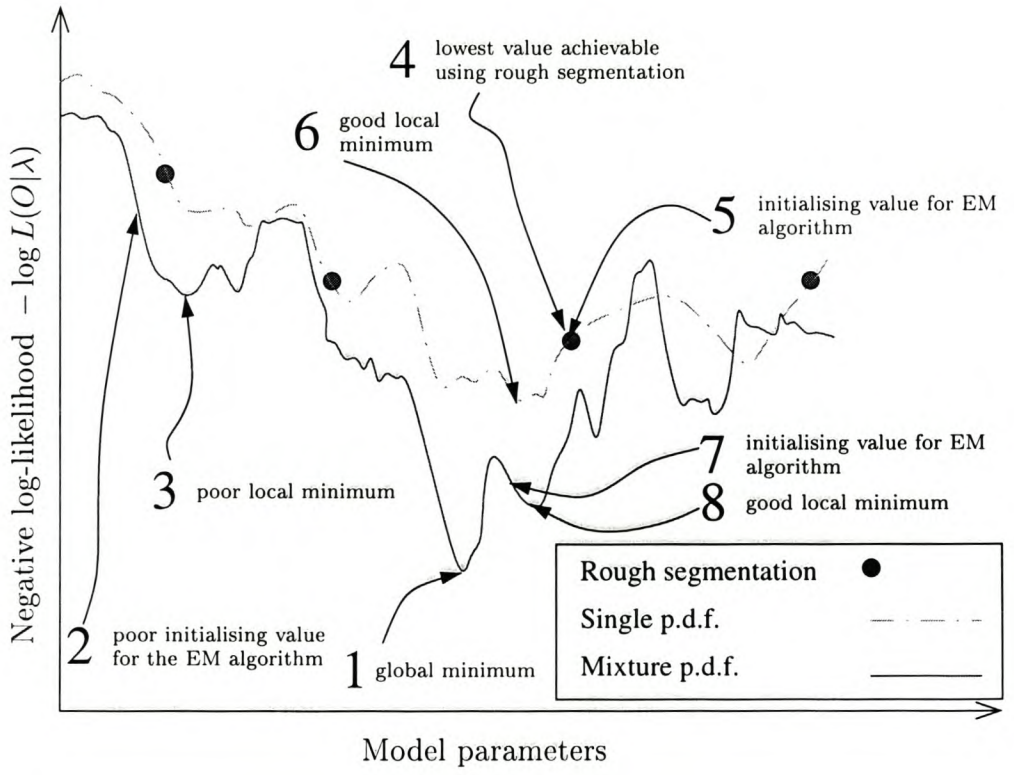

Figure 2.10: The effect of local minima on training. The dashed curve represents the negative log-likelihood of the HMM with single Gaussian state pdfs. The red dots on the curve represent possible parameter values from the rough segmentation of data. The solid curve represents the negative log-likelihood of the HMM with GMM state pdfs. The points labelled 1 to 8 represent values of the negative log-likelihood of the two HMMs at various stages in the algorithmic training process.

maximum. Performance of the EM algorithm is strongly linked to how well the model parameters are initialised. The step-wise algorithm used is essentially a set of progressively finer EM algorithms, each acting as an initialiser for its successor. It is hoped that in this fashion it is more likely to achieve the global maximum, or at least a higher local maximum than simply running a Viterbi (or Baum-Welch) EM reestimation of the HMM blindly.

Figure 2.10 is intended to give a rough idea of what happens when the HMM model is trained using progressively finer training iterations. The negative log-likelihood of the data given the model is plotted, so that the maxima of the likelihood function are now the minima of the negative log-likelihood function. Such a plot should have been multidimensional, but a one-dimensional representation was chosen for simplicity's sake.

Point 4 (Figure 2.10) represents the negative log-likelihood for a HMM with single Gaussian state pdfs when using the simple rough segmentation of the input data. Note that such a segmentation allows a very restricted range of parameter estimates, so that we cannot achieve any value on the dashed-dot log-likelihood curve. The dashed-dot curve represents the entire $-\log L(O|\lambda)$ for the HMM where every non-nullstate has a single pdf, and no restraints are placed on either the estimation of the pdfs or the transition probabilities. A complete Viterbi (or Baum-Welch) reestimation of all the model parameters may be done at this stage. The final continuous curve represents the $-\log L(O|\lambda)$ function for the final HMM where every non-nullstate has a mixture pdf, no restraints are placed on the model parameters, and a complete Viterbi (or Baum-Welch) reestimation of all the model parameters is possible.

In Figure 2.10 the global minimum for the full-complexity model is indicated by 1. Had the HMM simply been randomly initialised (indicated by 2), training is likely to yield one of the other local minima (indicated by 3) with parameter values close to the initialising values of the EM algorithm.

During the first segmentation of the data into chronological blocks and determining initial ML estimates for the state pdf parameters, the range of values for the parameters for the HMM model is effectively reduced, and fewer points on the likelihood function are attainable. As fewer parameter values are available, the likelihood function is effectively smoothed, resulting in fewer local minima, thereby making it easier to get close to the

global minimum on the smoothed function. Since the segmentation is very rough, the ML estimates for the pdfs are relatively robust. The minimum achieved is indicated by 4.

Using the pdfs estimated during the rough segmentation as the initial state pdfs, and the transition probabilities of the designed desired model structure as the state initial transition probabilities for the HMM, a few iterations of complete Viterbi reestimations of the model parameters can now be done. The model initialisation point is indicated by 5, and the eventual minimum achieved is indicated by 6. The entire curve is lower than the previous curve, as the model has been allowed more degrees of freedom for adaptation, and can thus model the observed data more accurately. Unfortunately this will also increase the complexity of the curve, making the occurrence of local minima more likely.

More degrees of freedom are now given by increasing the number of mixture components for each state from one component to two. More degrees of freedom means that the $-\log L(O|\lambda)$ will be more complex than before, but also lower than the other curves. The initial values for the state mixture pdfs are obtained by an ML estimate on the data segmentation of the previous HMM model, and the initial state transition probabilities remain the same as for the previous model. In Figure 2.10 the initialisation point is indicated by 7, and the subsequent minima obtained after training is indicated by 8.

In this fashion a more robust and accurate estimation of the HMM parameters is possible than when the HMM is simply trained using random initialisation of model parameters. To test the performance of the algorithm, two SI systems were trained. The first system was trained by using random initialisation and Viterbi reestimation for the HMMs and the second was trained using the algorithm described in this section. Each SI system had an HMM for each phoneme in the reduced phoneme set [32] for the TIMIT [39] corpus. The HMMs had three emitting states and an mixture (diagonal) Gaussian pdfs with eight mixture components for each state. For the system that was initialised with random values, the phoneme recognition rate was 66.51%, and for the system that was trained with our algorithm, the phoneme recognition rate was 67.55%.

Though they were not employed in this thesis, other methods for the avoidance of ill-suited local minima exist, such as the use of genetic algorithms and simulated annealing during the training process. These methods are a little more data-driven and rely less on

prior knowledge. However, good initialisation as supplied by the step-wise algorithm will narrow the range of parameters through which these algorithms need to search, reducing training time and locating better minima.

Chapter 3

MAP Adaptation

3.1 Introduction to MAP Adaptation

Maximum *a posteriori* (MAP) adaptation of HMMs was introduced by Gauvain and Lee [18]. ML adaptation of HMMs yields poor results for sparse training data. MAP adaptation makes it possible to include prior information into the estimation process, thereby making it more suitable for robust estimation of HMM parameters using relatively sparse data.

MAP adaptation does not adapt parameters for which no data was observed, and can therefore not compete against the more powerful rapid speaker adaptation techniques such as CAT, MLLR, MLED and weighted eigen-decomposition. RMP, CAT, MLLR and especially MLED can still make robust estimates in conditions of extreme data sparseness due to their data spreading capabilities. Rapid adaptation techniques such as MLLR and MLED can operate under conditions where data is so sparse that a MAP estimate for a "seen" parameter (i.e. a parameter for which data was observed) is not robust.

Though it cannot compete with other methods under conditions of extreme data sparseness, MAP adaptation has many valuable qualities. The computational requirements for MAP adaptation are much lower than those required for other adaptation methods. It also has the advantage that it converges to the ML estimate as the number of observation vectors tends to infinity, something neither MLED nor MLLR[1] can claim.

[1] MLLR is convergent for the special case where a separate transformation matrix is estimated for each mean vector. See Appendix B.2

MAP adaptation is thus useful under conditions where most of the model parameters are seen and have moderate amounts of data assigned to them.

3.2 Basic MAP Equations

When we train a statistical model, we wish to find the optimal set of model parameters given the observed data. In other words, we wish to maximise $f(\theta|Y)$ with respect to θ, where θ represents the set of model parameters, Y is the observed data, and $f(\cdot|\theta)$ is the pdf associated with Y. $f(\theta|Y)$ is not always a readily obtainable function, and in Section 2.3.2 it is shown that maximising $f(Y|\theta)g(\theta)$ is equivalent to maximising $f(\theta|Y)$, where $g(\theta)$ is the prior pdf of θ. Optimising $f(Y|\theta)g(\theta)$ with respect to θ is known as maximum *a posteriori* estimation (MAP estimation), and the optimal parameter set θ_{MAP} is found by solving

$$\theta_{\text{MAP}} = \underset{\theta}{\text{argmax}}\, f(Y|\theta)g(\theta). \tag{3.1}$$

Comparing this with the ML estimation equation

$$\theta_{\text{ML}} = \underset{\theta}{\text{argmax}}\, f(Y|\theta). \tag{3.2}$$

ML estimation may be seen as a special case of MAP estimation where a non-informative prior pdf for θ has been assumed, i.e. $g(\theta)$ is constant. When no prior knowledge regarding the distribution of the parameter set is available, the ML estimate is the right one to choose; however, when such knowledge is available, it should be incorporated via MAP estimation in order to make a better estimate.

In order for Equation 3.1 to be employed, three issues must be addressed: the choice of prior distribution family, the parameter values for the prior densities, and the evaluation of the MAP estimate.

1. **The choice of prior distribution family:** As is the case for ML estimation, MAP estimation is relatively simple if the family of pdfs $\{f(\cdot|\theta), \theta \in \Theta\}$ has a sufficient statistic, $t(x)$ (see Section 2.3.2.1) for θ. A sufficient statistic is a real or vector valued function of the data and contains all the information necessary to estimate model parameters. If $f(x|\theta)$ has a sufficient statistic $t(x)$ for θ, it may be factored

as follows:

$$f(x|\theta) = k\left(\theta|t(x)\right) h(x), \tag{3.3}$$

where $h(x)$ is independent of θ, and $k\left(\theta|t(x)\right)$ — known as the kernel density — is a function of θ that depends on x only through the sufficient statistic $t(x)$.

When $f(x|\theta)$ has a sufficient statistic, the natural solution for the prior pdf is to choose it from the same conjugate family $\{k(\cdot|\phi), \phi \in \Phi\}$ to which the kernel density $k\left(\theta|t(x)\right)$ belongs (see Appendix A.1). As $h(x)$ is independent of θ, MAP estimation may then be performed by maximising $k\left(\theta|\phi\right) k\left(\theta|t(x)\right)$ instead of $f(x|\theta)g(\theta)$.

2. **The specification of parameters for the prior density:** According to Gauvain and Lee [18], the parameters for the prior density may be based on subjective knowledge of the stochastic process. Alternately, they may be determined by an empirical Bayes approach whereby the prior parameters are determined directly from the adaptation data. One problem incurred when using the empirical Bayes approach, is that the parameters of the prior density add more parameters that need to be estimated using the limited amount of adaptation data. This problem is alleviated by constraining the size of the prior family by adding constraints on the prior parameters, and also by tying of the prior parameters.

3. **Evaluation of the MAP estimate:** If the complete data is available and a sufficient statistic exists for the problem, then the solution is usually straightforward. For HMMs, however, the complete data is not known. Fortunately Dempster *et al.* [10] proved that the EM algorithm may be applied to MAP estimation, where the function that is iteratively maximised is $R(\theta, \tilde{\theta})$, given by

$$R(\theta, \tilde{\theta}) = Q(\theta, \tilde{\theta}) + \ln g(\theta), \tag{3.4}$$

where $Q(\theta, \tilde{\theta})$ is the auxiliary function defined for ML EM maximisation defined by Equation 2.50 in Section 2.3.3.

3.3 MAP Adaptation of HMMs

Gauvain and Lee [18] show that the prior density for an HMM may be modelled as the product of Dirichlet and normal-Wishart densities (see Appendix A.2). The parameters

of an HMM cannot be simply determined using MAP estimation, as the state sequence is unknown and the complete data is therefore not available. Thus the EM algorithm is applied to MAP estimation in order to maximise the parameters. Either the Baum-Welch or the Viterbi algorithm will be used to obtain the state and mixture occupation probabilities. All the HMM parameters may be maximised using MAP adaptation, but as the other adaptation methods treated adapt only the mean vectors, this thesis will constrain MAP adaptation to adapt only the mean vectors of the Gaussian pdfs in the HMM.

The reestimation for a mean vector is

$$\tilde{\mu}_m^{(s)} = \frac{\tau_m^{(s)}\mu_m^{(s)} + \sum_{t=1}^{T}\gamma_m^{(s)}\mathbf{o}_t}{\tau_m^{(s)} + \sum_{t=1}^{T}\gamma_m^{(s)}}, \tag{3.5}$$

where s is the state, m is the component number, $\tau_m^{(s)}$ is a parameter originating from the normal-Wishart prior density for the mixture component m, $\mathbf{o}_t$ is the observation vector for time t, and $\gamma_m^{(s)}$ is the mixture-occupation probability.

Suppose the number of observation vectors goes to infinity. Then

$$\lim_{T\to\infty} \frac{\tau_m^{(s)}\mu_m^{(s)} + \sum_{t=1}^{T}\gamma_m^{(s)}\mathbf{o}_t}{\tau_m^{(s)} + \sum_{t=1}^{T}\gamma_m^{(s)}} = \frac{\sum_{t=1}^{T}\gamma_m^{(s)}\mathbf{o}_t}{\sum_{t=1}^{T}\gamma_m^{(s)}}. \tag{3.6}$$

Therefore, as the number of observation vectors goes to infinity, the MAP estimate of the mean vector converges to the ML estimate for the mean vector.

From Equation 3.5, we see that $\tau_m^{(s)}$ may be interpreted as a confidence measure in the new estimate. The smaller $\tau_m^{(s)}$, the more confidence is placed on the portion of the mean vector determined by the observed data. The larger $\tau_m^{(s)}$, the more the new estimate relies on the value of the previous mean vector, $\mu_m^{(s)}$. Though the empirical Bayes method can be used to obtain the value of $\tau_m^{(s)}$, $\tau_m^{(s)}$ was determined heuristically. We simply tested the recognition performance for various $\tau_m^{(s)}$ values and selected the optimal $\tau_m^{(s)}$ value. Also, all the $\tau_m^{(s)}$ were set equal, so that a single τ was used. This is equivalent to using the same prior pdf for all the Gaussian pdfs in the HMM.

To summarise: The MAP adaptation of a single mean vector is given by Equation 3.5, where the mixture occupation probabilities were obtained using either the Baum-Welch or Viterbi algorithms. As the number of observation vectors approaches infinity, the MAP estimate converges to the value of the ML estimate for the mean vector. For the case

of adapting mean vectors, the τ parameter from the prior pdf operates as a confidence measure between the old and new mean vector estimates.

3.4 MAP Algorithm Summary

This is a succinct treatment of implementing the MAP algorithm for the adaptation of mean vectors in GMMs in HMMs of a speaker speech model. The treatment will proceed in three steps:

- The algorithm and its prerequisites, divided into online and offline steps.
- Points to bear in mind for general MAP implementations.
- Points to bear in mind specific to the implementation of the algorithm in this thesis.

3.4.1 Algorithm Implementation

The algorithm is split into online and offline steps:

- **Offline Steps:**
  - Train an SI model, using a large amount of representative speech data from a representative set of training speakers. This SI model will form the initial model for each new target speaker for which MAP adaptation is to be performed. The mean vectors ($\mu_m^{(s)}$) of the SI model also form part of the prior for MAP adaptation.
  - Using a test subset of speakers, determine $\tau_m^{(s)}$ values using the empirical Bayes approach, or through trail and error.
- **Online Steps:**
  - Use the Viterbi or Baum-Welch algorithms and the SI model to segment the adaptation data obtained from the target speaker. From the segmentation, the following accumulators are obtained:

$$\sum_{t=1}^{T} \gamma_m^{(s)}(t)\mathbf{o}_t,$$

and

$$\sum_{t=1}^{T} \gamma_m^{(s)}(t).$$

– Using the prior mean vectors from the SI model, $\mu_m^{(s)}$, the values arising from the prior pdfs $\tau_m^{(s)}$, and the accumulators computed during the segmentation of the adaptation data, the new, adapted mean vectors for the target speaker are computed using

$$\tilde{\mu}_m^{(s)} = \frac{\tau_m^{(s)} \mu_m^{(s)} + \sum_{t=1}^{T} \gamma_m^{(s)} \mathbf{o}_t}{\tau_m^{(s)} + \sum_{t=1}^{T} \gamma_m^{(s)}}.$$

3.4.2 General Implementation Issues

The following points must be noted regarding the implementation of the MAP algorithm in general:

- The SI model and the speaker models obtained by MAP adaptation must have the same structure, i.e. the same number of HMMs, states, Gaussian pdfs in GMMs, and state transition probability links. This does not mean that the HMMs in a single speaker model must all be identical, or that HMMs for different speakers must have identical parameter values, only that the HMMs must have similar structure.

- The prior mean vectors $\mu_m^{(s)}$ used during the calculation of the mean vectors are those of the SI model. If multiple adaptation iterations are used, the same prior SI mean vectors ($\mu_m^{(s)}$) should be used, and not the mean vectors obtained during the previous adaptation run. The only aspect that will thus change during multiple MAP adaptation iterations is the segmentation of the adaptation data, for which the newest available model is used.

- The treatment in this thesis is only for the MAP adaptation of mean vectors. It should be noted, however, that MAP adaptation can be used to reestimate all HMM parameters, and therefore all parameters of a speaker speech model.

3.4.3 Implementation Issues Specific to this Thesis

The following points must be noted regarding the implementation of the MAP algorithm in this thesis:

- The adaptation is static, i.e. all of the available adaptation data is gathered before the model is reestimated. A separate and independent data set is used for testing, except for some of the experiments on the TIMIT corpus.
- Adaptation is supervised, i.e. the transcription of the adaptation data is known.
- Only the mean vectors are reestimated. Parameters that are not reestimated remain the same as that of the SI model.
- In order to test the performance of MAP adaptation against MLLR adaptation, the covariance matrices in the Gaussian pdfs in the speaker speech models are limited to being diagonal matrices.
- When multiple adaptation iterations were used, the prior mean vectors $\mu_m^{(s)}$ were taken from the mean vectors of the previous model. Strictly speaking, the prior mean vectors for MAP adaptation should always remain those of the SI model.

3.5 Strengths and Weaknesses of MAP

Two strengths of MAP adaptation are: convergence to the ML estimate as the number of observation vectors goes to infinity, and low computational requirements.

A detailed deduction of the computational requirements of several adaptation methods may be found in [51]. Assume that a speaker model consists of H HMMs (each representing a phoneme), S states per HMM and M mixture components per state and n-dimensional observation vectors. Further, assume that each observation is T frames long, and that P is the number of phonemes observed. The computational requirements for segmentation and determining the accumulators are $O\left((SM)^2T\right)$ in time and $O(HSMn)$ in memory. The computational requirements of MAP — excluding the Baum-Welch or Viterbi segmentation and determination of accumulators — are then approximately [51]:

- **Memory:** $O(HSMn)$
- **Computations:** $O\left(SM(SMT + Pn)\right)$

The cost of segmentation and determining the accumulators is not included, as it is a cost shared by all the treated adaptation methods.

Most weaknesses of MAP adaptation result from the requirement that each parameter must be estimated using only the seen observation data for that parameter. Two weaknesses resulting from this requirement are:

- MAP is unable to adapt parameters for which no data was observed. Other adaptation methods such as MLED and MLLR have the ability to adapt all the HMM mean vectors using a small number of observations for a few mean vectors.

- Dimension reduction of the speaker model and the pooling of observed data go hand in hand. MAP does not reduce the dimension of the model, and therefore does not pool all the observed data. MLLR pools the observed data for a group of mean vectors, and reduces the dimension of the parameters to be adapted by estimating a single transform for all mean vectors in the group. MLED pools the observed data for the entire speaker model, and adapts all model parameters using the pooled data in a lower-dimensional space. Due to the dimension reduction capabilities of MLLR and MLED, these methods require much less adaptation data to generate robust estimates than MAP adaptation would.

The weaknesses of MAP make it an inappropriate choice when the amount of available adaptation data is very little. The strengths, however, make it an excellent post-adaptation method of increasing recognition performance of those parameters for which adequate amounts of data were observed. For instance, MLLR or MLED may be applied as a rapid speaker adaptation method, making a rough but robust estimate for the new speaker model. The computationally light MAP is then used on the MLLR of MLED estimate to form finer estimates of those mean vectors for which enough adaptation data were observed.

Chapter 4

MLLR Adaptation

4.1 Introduction to the MLLR Adaptation Approach

Maximum likelihood linear regression (MLLR) adaptation is a popular and widely used rapid speaker adaptation technique. So much so that MLLR using global clustering is often used as a benchmark for other speaker adaptation methods.

Much research has been done since its introduction in 1995 by C.J. Leggetter and P.C. Woodland [34, 33], and now several more flexible MLLR-derived techniques are available [15] as well as more extensive clustering methods [17, 23] for the original MLLR method.

MLLR focuses on the adaptation of CDHMMs with GMM (including single Gaussian) state pdfs, where each Gaussian has a diagonal covariance matrix. Only the mean vectors of the Gaussian distributions are reestimated in MLLR adaptation, and all other model parameters remain fixed. Adaptation is realised by grouping (or clustering) a set of mean vectors together, estimating an optimal linear transformation for the group from adaptation data, and then applying the same linear transform to each of these mean vectors to form the new estimate for each of them. Several clusters may be used, and each mean vector will be adapted using only the linear transformation estimated for the cluster it was assigned to. This thesis will refer to these clusters as regression classes.

The parameter reduction inherent in the tying of the mean vectors has the effect of pooling together the data observed for a set of Gaussian distributions, allowing for robust estimates when observation data is scarce. Another benefit of this form of parameter reduction is that adaptation of distributions for which no data were observed is made

possible, as the mean vectors of these dataless distributions may be adapted using the linear transformation estimated from the data observed for the other distributions in the regression class. This last advantage distinguishes MLLR as a rapid speaker adaptation technique, because the ability to adapt distributions for which there were no observed data is not a trait that simple adaptation techniques such as ML and MAP exhibit.

4.2 Basic MLLR Adaptation Equations

As previously stated, the speaker model to be adapted consists of an HMM with mixture Gaussian state probability density functions, and the only parameters MLLR adapts are the mean vectors of the Gaussian models in the speaker model.

For the first MLLR adaptation iteration, the pre-adaptation model is the SI model. Each of the pre-adaptation Gaussian pdfs is given by

$$b_m^{(s)}(\mathbf{o}) = \frac{1}{(2\pi)^{n/2}\left|C_m^{(s)}\right|^{1/2}} e^{-\frac{1}{2}\left(\mathbf{o}-\mu_m^{(s)}\right)' C_m^{(s)-1}\left(\mathbf{o}-\mu_m^{(s)}\right)} \tag{4.1}$$

where n is the feature vector dimension, s is the state in the HMM, m is the mixture component of the GMM pdf for s, $C_m^{(s)}$ is the covariance matrix, $\mu_m^{(s)}$ is the mean vector and $\mathbf{o}$ is the feature vector.

MLLR assumes that mean vectors may be grouped into regression classes, and that a new estimate $\hat{\mu}_m^{(s)}$ for a mean vector may be formed by applying a linear transformation estimated for its regression class to the initial mean vector, $\mu_m^{(s)}$. Thus

$$\hat{\mu}_m^{(s)} = T_\alpha \mu_m^{(s)} + \mathbf{b}_\alpha, \tag{4.2}$$

where T_α is a matrix and $\mathbf{b}_\alpha$ is an offset vector calculated for the regression class to which $\mu_m^{(s)}$ belongs. T_α performs rotation and scaling of the original mean vector, and $\mathbf{b}_\alpha$ adds an offset to map the mean vector of the initial model to a mean vector for the new target speaker model.

An alternate representation of Equation 4.2 is possible if extended mean vectors are used.[1] An extended mean vector $\xi_m^{(s)}$ is created by prepending an offset term, ω, to the

[1] The alternative representation makes for greater ease during the derivation of the optimal linear transformations.

current mean vector. Therefore $\xi_m^{(s)}$ is given by

$$\xi_m^{(s)} = \begin{bmatrix} \omega \\ \mu_m^{(s)} \end{bmatrix}. \tag{4.3}$$

To complete the linear transformation of the original mean vector, the extended mean vector is multiplied by a transformation matrix W_α.

$$\hat{\mu}_m^{(s)} = W_\alpha \xi_m^{(s)} \tag{4.4}$$

where $\hat{\mu}_m^{(s)}$ is the $(n) \times 1$ adapted mean vector, W_α is the $n \times (n+1)$ transformation matrix and $\xi_m^{(s)}$ is the $(n+1) \times 1$ extended mean vector. It should be noted that the first column of W_α represents the offset vector $\mathbf{b}_\alpha$ that will be added to the original mean vector, and the remaining columns represent the rotation and scaling transform A_α of the original mean vector. If we wish an offset to be included in the regression, then $\omega = 1$; if not, then $\omega = 0$.

Thus, after adaptation the Gaussian pdf will be:

$$\hat{b}_m^{(s)}(\mathbf{o}) = \frac{1}{(2\pi)^{n/2} \left| C_m^{(s)} \right|^{1/2}} e^{-\frac{1}{2}\left(\mathbf{o} - W_\alpha \xi_m^{(s)}\right)' C_m^{(s)-1} \left(\mathbf{o} - W_\alpha \xi_m^{(s)}\right)} \tag{4.5}$$

A total of A regression classes are used to adapt the mean vectors of the speaker model, and for each of these regression classes a single transformation matrix is estimated. There are thus A transformation matrices, W_α, where α represents a regression class the transformation matrix belongs to and $\alpha \in \{1 \ldots A\}$. Each mean vector is assigned to one regression class exclusively, and will be transformed using the transformation matrix estimated for this regression class. The following section will treat the maximum likelihood estimation of these transformation matrices given an observed data sequence and the current model parameters.

4.3 Determining the Transformation Matrices for MLLR Adaptation

The following deduction is for the determination of a transformation matrix where the speaker model to adapt is a single HMM with mixture Gaussian state pdfs. At the end of the discussion it will be shown how the results are easily extended for the case where a speaker model consists of several HMMs.

4.3.1 Defining the Auxiliary Function

This section will briefly review the topics treated in Sections 2.5.1 and 2.5.2: reestimation of model parameters in a maximum likelihood framework, the likelihood function to be maximised, and the auxiliary function that may be maximised instead of the likelihood function. To estimate the best set of transformation matrices for MLLR adaptation, it is necessary to maximise the likelihood of the observed data given the model parameters, $L(O|\lambda)$. $L(O|\lambda)$ is given by Equation 2.94, which is restated here for convenience:

$$L(O|\lambda) = \sum_{\theta \in \Theta} \sum_{\zeta \in \Psi_\theta} L(O, \theta, \zeta|\lambda), \tag{4.6}$$

where Ψ_θ is the set of all possible mixture component sequences given the state sequence θ, and ζ is a mixture component sequence in Ψ_θ.

An auxiliary function may be defined such that parameters that maximise it will also increase the value of the likelihood function $L(O|\lambda)$. This auxiliary function $Q(\lambda|\hat{\lambda})$ is given by

$$Q(\lambda, \hat{\lambda}) = \sum_{\theta \in \Theta} \sum_{\zeta \in \Psi_\theta} L(O, \theta, \zeta|\lambda) \log \left[L(O, \theta, \zeta|\hat{\lambda}) \right]. \tag{4.7}$$

$L(O|\lambda)$ may now be maximised by iteratively maximising $Q(\lambda|\hat{\lambda})$ and updating the model parameters with the reestimated model parameters, $\hat{\lambda}$.

When the auxiliary function was expanded, it was found that some of the terms were constant, and thus would play no part in the maximisation of the auxiliary function. The expanded auxiliary function with constant terms removed is given by Equation 2.105. It is this function that will be maximised with respect to the transformation matrices (W_α, where $\alpha \in \{1 \ldots A\}$), and it is restated here for convenience

$$Q(\lambda, \hat{\lambda}) = -\frac{1}{2} L(O|\lambda) \sum_{s=1}^{S} \sum_{m=1}^{M_s} \sum_{t=1}^{T} \gamma_m^{(s)}(t) \left[n \log(2\pi) + \log |C_m^{(s)}| + h_m^{(s)}(\mathbf{o}_t) \right], \tag{4.8}$$

where

$$\begin{aligned} h_m^{(s)}(\mathbf{o}_t) &= \left(\mathbf{o}_t - \hat{\mu}_m^{(s)}\right)' C_m^{(s)-1} \left(\mathbf{o}_t - \hat{\mu}_m^{(s)}\right) \\ &= \left(\mathbf{o}_t - W_\alpha \xi_m^{(s)}\right)' C_m^{(s)-1} \left(\mathbf{o}_t - W_\alpha \xi_m^{(s)}\right). \end{aligned} \tag{4.9}$$

4.3.2 Maximising the Auxiliary Function $Q(\lambda, \hat{\lambda})$ with Respect to the Transformation Matrix W_α

The auxiliary equation as it appears in Equation 4.8 must be maximised with respect to the transformation matrices in order to increase the value of the objective function $L(O|\lambda)$. Note that there are no terms in the auxiliary function that contain more than one regression matrix, and that the regression matrices are independent of one another $\left(\frac{\partial W_i}{\partial W_j} = 0, i \neq j\right)$, so that the auxiliary function can be maximised separately for each of the transformation matrices.

$$\begin{aligned}
\frac{\partial Q(\lambda, \hat{\lambda})}{\partial W_\alpha} &= -\frac{1}{2} L(O|\lambda) \frac{\partial}{\partial W_\alpha} \sum_{s=1}^{\mathrm{S}} \sum_{m=1}^{M_s} \sum_{t=1}^{T} \gamma_m^{(s)}(t) \left[n \log(2\pi) + \log |C_m^{(s)}| + h_m^{(s)}(\mathbf{o}_t) \right] \\
&= -\frac{1}{2} L(O|\lambda) \sum_{s=1}^{\mathrm{S}} \sum_{m=1}^{M_s} \sum_{t=1}^{T} \gamma_m^{(s)}(t) \left[\frac{\partial}{\partial W_\alpha} h_m^{(s)}(\mathbf{o}_t) \right] \\
&= -\frac{1}{2} L(O|\lambda) \sum_{s=1}^{\mathrm{S}} \sum_{m=1}^{M_s} \sum_{t=1}^{T} \gamma_m^{(s)}(t) \left[-2 C_m^{(s)-1} \left[\mathbf{o}_t - W_\alpha \xi_m^{(s)} \right] \xi_m^{(s)\prime} \right] \\
&= L(O|\lambda) \sum_{s=1}^{\mathrm{S}} \sum_{m=1}^{M_s} \sum_{t=1}^{T} \gamma_m^{(s)}(t) C_m^{(s)-1} \left[\mathbf{o}_t - W_\alpha \xi_m^{(s)} \right] \xi_m^{(s)\prime}. \qquad (4.10)
\end{aligned}$$

For a more detailed discussion of determining the derivative of $h_m^{(s)}(\mathbf{o}_t)$ with respect to W_α, see Section B.1. To maximise the auxiliary function with respect to W_α, the result of the differentiation is now set to zero. If the mean vector of $h_m^{(s)}(\mathbf{o}_t)$ is tied to a regression matrix other than W_α, it will become zero after differentiation. We can thus change the summation over S and M_s to a summation over R_α, where R_α is the set of all pdfs with mean vectors $\mu_m^{(s)}$ tied to regression matrix W_α.

$$\begin{aligned}
\frac{\partial Q(\lambda, \hat{\lambda})}{\partial W_\alpha} &= 0 \\
L(O|\lambda) \sum_{(s,m) \in R_\alpha} \sum_{t=1}^{T} \gamma_m^{(s)}(t) C_m^{(s)-1} \left[\mathbf{o}_t - W_\alpha \xi_m^{(s)} \right] \xi_m^{(s)\prime} &= 0. \qquad (4.11)
\end{aligned}$$

Therefore

$$\sum_{(s,m) \in R_\alpha} \sum_{t=1}^{T} \gamma_m^{(s)}(t) C_m^{(s)-1} \mathbf{o}_t \xi_m^{(s)\prime} = \sum_{(s,m) \in R_\alpha} \sum_{t=1}^{T} \gamma_m^{(s)}(t) C_m^{(s)-1} W_\alpha \xi_m^{(s)} \xi_m^{(s)\prime}. \qquad (4.12)$$

$$Z = Y. \qquad (4.13)$$

It should be noted that there is no closed form solution for W_α unless all the covariance matrices are diagonal.[2] The right side of Equation 4.12 will now be represented by matrix Y, and the left side by Z. Furthermore, two other matrices, V and D, will be introduced:

$$V_m^{(s)} = \sum_{t=1}^{T} = \gamma_m^{(s)}(t) C_m^{(s)-1} \tag{4.14}$$

(an $n \times n$ diagonal matrix) and

$$D_m^{(s)} = \xi_m^{(s)} \xi_m^{(s)\prime} \tag{4.15}$$

an $(n+1) \times (n+1)$ matrix. Now the $n \times (n+1)$ matrix Y is given by

$$Y = \sum_{(s,m) \in R_\alpha} V_m^{(s)} W_\alpha D_m^{(s)} \tag{4.16}$$

If we denote the individual elements of Y, $V_m^{(s)}$, W and $D_m^{(s)}$ by y_{ij}, $v_{ij}^{(s,m)}$, w_{ij} and $d_{ij}^{(s,m)}$, respectively, then

$$y_{ij} = \sum_{p=1}^{n} \sum_{q=1}^{n+1} w_{pq} \left[\sum_{(s,m) \in R_\alpha} v_{ip}^{(s,m)} d_{qj}^{(s,m)} \right] \tag{4.17}$$

Since $D_m^{(s)}$ is symmetric and the covariance matrices are diagonal,

$$\sum_{(s,m) \in R_\alpha} v_{ip}^{(s,m)} d_{qj}^{(s,m)} = \begin{cases} \sum_{(s,m) \in R_\alpha} v_{ii}^{(s,m)} d_{jq}^{(s,m)} & \text{when } i = p \\ 0 & \text{when } i \neq p \end{cases} \tag{4.18}$$

If the elements of an $(n+1) \times (n+1)$ matrix $G^{(i)}$ are given by

$$g_{jq}^{(i)} = \sum_{(s,m) \in R_\alpha} v_{ii}^{(s,m)} d_{jq}^{(s,m)} \tag{4.19}$$

the elements of Y are may be expressed as

$$y_{ij} = \sum_{q=1}^{n+1} w_{iq} g_{jq}^{(i)} = z_{ij} \tag{4.20}$$

where z_{ij} are the individual elements of Z.

[2] Although MLLR has a closed form solution for the optimal transformation matrices W only when the models have diagonal covariance matrices, Gales [15] shows how EM can be used to iteratively determine transformation matrices for models with non-diagonal covariance matrices.

Since the elements of both Z and $G^{(i)}$ can be computed using only the model parameters and observations, they are not dependent on W_α. Thus W_α can be computed by the set of equations

$$w_i' = G^{(i)-1} z_i' \tag{4.21}$$

where w_i and z_i are the i-th rows of W_α and Z respectively. The equations can be solved row by row using Gaussian elimination. One interesting point seen from the derivations is that MLLR reduces to simple ML reestimation of the mean vectors for the case when the occupancy of each regression class consists of one mean vector (see Appendix B.2).

Extending the results to the case where a speaker model consists of various HMMs is straightforward. All of these separate HMMs can be seen as segments of an all-encompassing HMM model for the speaker. During supervised adaptation the HMM modelling the observation sequence is known, and in unsupervised adaptation the observation is assigned to the HMM most likely to have generated the observation sequence. In the all-encompassing HMM this means that the state occupation probabilities for individual HMMs to which the observation sequence is not assigned are zero. From Equation 4.11, R_α now includes mean vectors from different HMMs. Any summation over R_α,

$$\sum_{(s,m)\in R_\alpha} \tag{4.22}$$

from Equation 4.11 to the final solution in Equation 4.21 is replaced by the summation

$$\sum_{(h,s,m)\in R_\alpha} \tag{4.23}$$

where h is an HMM model in the set of HMM models for the speaker.

Attention should be paid to the implications of Equations 4.15 and 4.19. Each $G^{(i)}$ matrix is formed by a weighted sum of vectors multiplied by their transpose. Therefore, if the number of vectors (the extended mean vectors) used in the creation of $G^{(i)}$ is less than the dimension of the extended mean vectors, $G^{(i)}$ will be rank deficient and therefore not invertable. Even if the number of vectors is greater than the dimension of the vectors, the matrix $G^{(i)}$ could have a high condition number, which will result in numerical instability during the inversion process. Leggetter [33] suggests using the Moore-Penrose pseudo-inverse, but the experiments conducted in this thesis showed that it was better not

to perform adaptation at all than to perform adaptation using transformation matrices calculated from a pseudo-inverse. With the above in mind, a regression class should always have more mean vectors assigned to it than the dimension of an extended mean vector. Also, the number of mean vectors to which adaptation data were assigned should be checked, so that no inversion of a rank deficient $G^{(i)}$ is attempted. Even though it is expensive, it is wise to check the condition numbers of the $G^{(i)}$ matrices during MLLR adaptation, especially when the regression classes have few mean vectors assigned to them (as is the case with systems employing HMMs with few states and few pdfs), or when the amount of adaptation data is very little.

4.4 The Choice of Regression Classes

A very important factor in successfully employing MLLR adaptation is selecting suitable regression classes. MLLR adaptation is limited by the assumptions on which it is based. Since it is implied that all the mean vectors in a regression class may be optimally adapted using the same linear transform, we must attempt to form regression classes of mean vectors that conform to this assumption.

If adaptation data is relatively scarce, using a few large regression classes (where each regression class contains many mean vectors) is preferable to using a large number of small regression classes (where each regression class contains a small number of mean vectors). This is because the adaptation data available for each regression class is pooled, and large regression classes will have a greater amount of pooled adaptation data with which to estimate the transformation matrices than small regression classes. The more data are available for a transformation matrix estimate, the more robust the estimate will be.

Choosing a greater number of finer regression classes containing only a few mean vectors each, will necessitate the estimation of more transformation matrices using the same amount of data. Using more regression matrices will increase the likelihood of the speaker model, up to the point where one regression class is used for every mean vector, when the MLLR estimate reduces to the ML estimate. Provided sufficient data is available for the robust estimation of these transformation matrices, a decrease in error rate is expected. However, if too little pooled data is available for robust estimates of

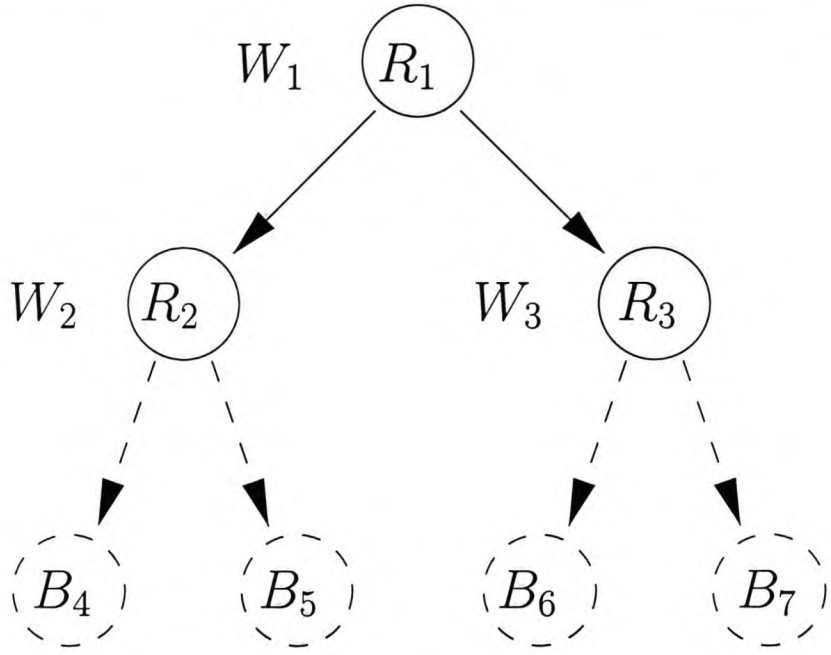

Figure 4.1: An example of a regression class tree.

transformations, then results will be poor.

Normally, the speech to be modelled consists of a finite set of phonemes or words. Phonetic relationships between the phonemes or words may be exploited in order to cluster mean vectors of similar phonemes or words together. It is hoped that similar-sounding phonemes will undergo similar transformations for a new speaker. The amount of adaptation data available will probably not be known, so the amount of regression classes that can be robustly estimated will not be known *a priori*. To remedy this, regression class trees are used.

4.4.1 Regression Class Trees

Any tree T may be implemented, where the tree is a hierarchy of regression classes. In the tree in Figure 4.1 are regression classes $\{R_1, R_2, R3\}$, each of which is made up of base classes $\{B_4, B_5, B_6, B_7\}$. These base classes could also be regression classes. As using finer regression classes will bring the estimate closer to that of the ML estimate, it is desirable to use regression classes as far down the tree as robust estimation will allow. A simple method to decide how far down the tree to go is by using an empirically determined minimum number of observation vectors for robust estimation. Such a procedure is described in the

following recursive algorithm (Gales [17]). Beginning at the top of the tree:

◇ *if parent node P has sufficient pooled data*
- ◇ *for all children C of P*
  - ◇ *if child of C has sufficient pooled data*
    - ◇ *if child of C is leaf node*
      - ◇ *estimate transformation matrix for C.*
      - ◇ *adapt mean vectors associated with C using transformation matrix estimated for C.*
    - ◇ *else*
      - ◇ *apply the recursive algorithm using child C as new parent node.*
  - ◇ *else*
    - ◇ *let P be regression class for mean vectors in C.*
- ◇ *if one or more child nodes of P had insufficient pooled data*
  - ◇ *estimate transformation matrix for P using pooled data of all child nodes in P.*
  - ◇ *apply transformation matrix associated with P on mean vectors of child nodes that were not adapted due to insufficient data.*

Using a regression class tree, it is possible to robustly adapt finer regression classes that have sufficient data, and also to adapt regression classes that have little or no data by using transformation matrices estimated for their parent class. Empirically determining a good threshold for sufficient adaptation data is not a pleasing method, and more elegant methods have been proposed by Gales [23, 17].

Assigning mean vectors to regression classes in the regression tree can be done by one of two methods: using phonetic knowledge of the speech corpus, or by clustering algorithms.

Tying using phonetic knowledge. This requires human expertise to form regression classes based on phonetic relationships in the speech. Drawbacks are that such

knowledge might not be available and that every new speech corpus will need a different expert phonetic clustering.

Tying by clustering algorithms. Generally this is done by clustering data according to proximity in acoustic space using some form of distance measure, or by iteratively determining the tying structure that will give the maximum for the sum of auxiliary functions (as defined in Equation 2.99) for various training speakers. Some more advanced methods even take into account the adaptation data available for the speaker model to be adapted. Clustering algorithms have the advantage that they are data-driven and require no phonetic knowledge input. However, more code must be generated, and a greater amount of off-line pre-adaptation computations must be done than would be the case for phonetic assignment.

It has been shown [33, 17] that very few clustering algorithms outperform the phonetic method for small problems. For the relatively small ISOLET database, the phonetic approach devised by P. Nguyen [40] was considered appropriate.

4.4.2 Choosing Regression Classes According to Phonetic Knowledge

There is a myriad of ways to form regression classes for a speaker model given phonetic knowledge of the system. Below are a few methods from rougher to finer granularity:

Global tying. All the mean vectors in the speaker model are grouped into a single regression class and adapted using the same transformation matrix.

Tying similar HMMs. The mean vectors of HMMs representing phonetically similar phonemes in the speaker model are grouped together.

Tying by HMM. All the mean vectors of an HMM representing a phoneme in the speaker model are grouped in the same regression class. This could be done if there are enough data to update the phoneme regardless of other phonemes, but not enough data to update the states of the phoneme separately. It is assumed that all the states in the phoneme will undergo approximately the same transformation.

Tying states in similar HMMs. This is the tying of similar phonemes in a state-by-state fashion. Mean vectors in the same state of various HMMs are tied. Thus, beginning and ending states of HMMs modelling similar phonemes tied. This offers differentiation between the transformations of different states for phonemes where states are likely to undergo dissimilar transformations.

Various other assignments might also be employed if the recognition problem warrants it. Care should be taken when creating a regression class tree from the regression classes to ensure that similar child nodes combine to form a parent node.

4.4.3 Assigning Regression Classes Using Clustering Algorithms

The following clustering schemes will be briefly discussed: acoustic proximity clustering and several optimal clustering methods as proposed by Gales [17] and Johnson and Woodland [23].

4.4.3.1 Acoustic Distance Clustering

The training data of many speakers are used. Data is segmented, and one of several distance metrics may be used (X and Y represent data segments). Johnson and Woodland [23] suggested the use of the arithmetic harmonic sphericity distance measure or the Gaussian divergence distance measure:

Arithmetic Harmonic Sphericity (AHS) [5]

$$d(X,Y) = \log[\mathrm{tr}(C_y C_x^{-1}) * \mathrm{tr}(C_x C_y^{-1})] - 2\log(n) \tag{4.24}$$

Gaussian Divergence

$$\begin{aligned} d(X,Y) &= 0.5\mathrm{tr}(C_x^{-1}C_y + C_y^{-1}C_x - 2I) && (4.25)\\ &+ 0.5(\mu_x - \mu_y)'(C_x^{-1} + C_y^{-1})(\mu_x - \mu_y) && (4.26) \end{aligned}$$

where C_x is a diagonal covariance matrix and μ_x is a mean vector of a data segment. Leggetter [33] made use of the Gaussian divergence measure, as well as the likelihood measure: a measure of the similarity between distributions.[3] When D distributions are

[3]The likelihood measure is also used in model decision tree building for clustering of distributions [33, 44]

merged to produce a single distribution e, the change in likelihood $\delta\mathcal{L}$ or likelihood measure is given by

$$\delta\mathcal{L} = \left(\sum_{d\in D}\frac{1}{2}\ln(|C_d|)\sum_{t=1}^{T}\gamma_d(t)\right) - \left(\frac{1}{2}\ln(|C_e|)\sum_{t=1}^{T}\gamma_e(t)\right) \tag{4.27}$$

where $\gamma_d(t)$ is the probability of being in state d at time t, and $\gamma_e(t)$ is the probability of being in the group of states e at time t. For more on these distance measures refer to Appendix B.3.

A binary split procedure can now be used to cluster the mean vectors using either of the above distance measures. Alternately a more flexible top-down split-and-merge clustering method may be used [23]. In this method an initial assignment of mean vectors is made to child nodes in the regression class tree. Segments are then moved between the child nodes to decrease the distance. Splitting continues until an empirical minimum occupancy for a regression class is reached. A merging procedure now follows during which similar child nodes are combined. Johnson and Woodland [23] also propose a method for re-assignment of segments to nodes using a direct maximisation of the log likelihood of the data, given the MLLR adaptation scheme. The recognition results for both the distance measure and the direct maximisation were of the same order.

4.4.3.2 Optimal Assignment to Regression Classes

Neither using phonetic knowledge to create a regression class tree, nor using acoustic distance methods are necessarily optimal. This paragraph focuses mostly on the work of Gales [17], who proposed several methods that are optimal in their assignment of regression classes. The assignment here is "optimal" in the sense that it maximises the likelihood of the data given the MLLR-estimated transformation matrices. An optimal assignment of a regression tree ($\mathbf{T}$) can be described by:

$$\hat{\mathbf{T}} = \underset{\mathbf{T}}{\operatorname{argmax}}\left\{\sum_{x=1}^{X} Q^{(x)}(\lambda, \hat{\lambda}|\mathbf{T})\right\} \tag{4.28}$$

where

$$\begin{aligned} Q^{(x)}(\lambda, \hat{\lambda}|\mathbf{T}) &= \text{constant} \\ &-\frac{1}{2}L(O|\lambda)\sum_{s=1}^{S}\sum_{m=1}^{M_s}\sum_{t=1}^{T}\gamma_m^{(s)}(t)\left[n\log(2\pi) + \log|C_m^{(s)}| + h_m^{(s)}(\mathbf{o}_t)\right] \end{aligned} \tag{4.29}$$

and $\{x = 1..X\}$ is a set of all the speakers available in the tree-building process. It is possible to make the splitting in the tree locally optimal, as this method only guarantees a local maximum auxiliary function. Using this strategy, a fixed regression tree can be obtained using tree-building data from several training speakers. The regression tree can also by built using only the using the adaptation data for the speaker or speakers to be adapted.

Optimal Assignment of Hard Clustering: Hard clustering is the normal use of MLLR, where a mean vector is adapted using one transformation matrix estimated for a regression class. A set of regression classes (typically obtained from acoustic clustering methods) is assumed and transformation matrices are obtained. Base classes are then assigned to the regression class that maximises its auxiliary function. If a base class has been reassigned to another regression class, the transformation matrices are reestimated using the new clustering, and the process is repeated. The process can be iterated a fixed number of times, or until no base class swaps to a different regression class.

Optimal Assignment of Fuzzy Clustering: Fuzzy clustering makes use of a weighted sum of transformation matrices to adapt a mean vector. This makes it possible to tie a mean vector to more than one regression class. The transformation for a mean vector is now given by:

$$\hat{\mu} = \left[\sum_{p=1}^{P} \omega_p W^{(p)}\right] \xi = \sum_{p=1}^{P} \omega_p \hat{\mu}^{(p)}. \tag{4.30}$$

For a given set of weights, transformation matrices can be estimated, which in turn can be used to reestimate the weight vectors [17]. The process of maximising the auxiliary function with respect to the transformation vectors, and then maximising the new auxiliary function with respect to the weights is an iterative method of maximising the likelihood of the tree-building data given the model. Fuzzy clustering is thus an optimal and soft clustering method of assigning base classes to regression classes. As fuzzy clustering has many more parameters to adapt than hard clustering, the degrees of freedom in the adaptation may be lessened by constraining the base classes to belong to certain regression classes only.

Stopping Criterion for Optimal Tree-building Methods: If the regression classes are not fixed when adaptation is done, a stopping criterion is necessary to ensure the generation of robustly estimated transformation matrices. As previously stated, this can be done by empirically determining a minimum number of observation vectors necessary to robustly estimate a transformation matrix for a regression class. Alternatively, non-empirical methods (cross-validation or iterative MLLR) that take into account the available adaptation data have been suggested by Gales [17].

Performance of Various Regression Tree Building Schemes: Previous research has shown that similar results are obtained through phonetic clustering and acoustic distance clustering. The only deciding factors here is whether or not phonetic knowledge is available and whether or not it is viable to generate code to perform an acoustic distance clustering.

Optimal methods for assigning base classes to regression classes would suggest a decrease in error rate. However, even though the optimal methods may increase the likelihood of the adaptation data considerably, it does not always coincide with a similar decrease in error rate. In some cases the letter error rate becomes even higher. This indicates that clustering schemes maximising the likelihood of the adaptation data using an MLLR scheme may increase the likelihood of the model, but that this does not bear a direct relationship to the error rate.

Gales showed that only the stopping criterion associated with the iterative MLLR method resulted in a lower error rate as compared to an empirically determined stopping criterion.

4.5 MLLR Algorithm Summary

This is a succinct treatment of implementing the MLLR algorithm for the adaptation of mean vectors in GMMs in HMMs of a speaker speech model. The treatment will proceed in three steps:

- The algorithm and its prerequisites, divided into online and offline steps.
- Points to bear in mind for general MLLR implementations.

- Points to bear in mind specific to the implementation of the algorithm in this thesis.

4.5.1 Algorithm Implementation

The algorithm is split into online and offline steps:

- **Offline Steps:**
  - Train an SI model, using a large amount of representative speech data from a representative set of training speakers. This SI model will form the initial model for each new target speaker for which MLLR adaptation is to be performed.
  - Assign mean vectors to base regression classes, and create a regression class tree. This is accomplished using any of the methods discussed in 4.4. For this thesis, phonetic knowledge was used to assign mean vectors to base regression classes, and to combine these base regression classes into a regression class tree.
- **Online Steps:**
  - Use the Viterbi or Baum-Welch algorithms and the SI model to segment the adaptation data obtained from the target speaker. From the segmentation, the following accumulators are obtained:

    $$\sum_{t=1}^{T} \gamma_m^{(s)}(t)\mathbf{o}_t,$$

    and

    $$\sum_{t=1}^{T} \gamma_m^{(s)}(t).$$

  - Given the data segmentation, the regression class tree can be altered according to the amount of seen data for each node in the regression class. More complex methods methods can also be used to optimise the regression classes and regression class tree for the observed adaptation data.
  - Determine the transformation matrix W_α for each regression class for which there was enough observed data. This is accomplished in three steps for each regression class: one, compute matrix Z using Equation 4.12, two, compute

the G-matrices using Equations 4.14, 4.15 and 4.19, and three, solve each row of W_α using Equation 4.21 :

$$w_i' = G^{(i)-1} z_i'.$$

– The extended mean vectors (i.e. a mean vector with a one or a zero prepended to it, as in Equation 4.3) from the previous model (this is the SI model for the first training iteration), $\xi_m^{(s)}$, are transformed by the transformation matrix to which they were assigned to form the new estimate:

$$\hat{\mu}_m^{(s)} = W_\alpha \xi_m^{(s)}.$$

4.5.2 General Implementation Issues

The following points must be noted regarding the implementation of the MLLR algorithm in general:

- The SI model and the speaker models obtained by MLLR adaptation must have the same structure, i.e. the same number of HMMs, states, Gaussian pdfs in GMMs, and state transition probability links. This does not mean that the HMMs in a single speaker model must all be identical, or that HMMs for different speakers must have identical parameter values, only that the HMMs must have similar structure.

- The prior mean vectors $\mu_m^{(s)}$ used during the calculation of the new mean vectors are those of the SI model for the first MLLR iteration. If multiple adaptation iterations are used, the prior mean vectors are those trained during the previous adaptation run. Both the prior speech model and the data segmentation is thus different for each MLLR iteration.

- MLLR in its simplest form can only be used for the adaptation of mean vectors. Parameters that are not part of mean vectors are not adapted, and remain the same as those in the initial SI model.

- For a closed form solution of the transformation matrices, the covariance matrices of the Gaussian pdfs in the speaker speech models must be diagonal. If full covariance matrices are desired, more complex methods involving the EM algorithm have to be used to determine the transformation matrices.

- Enough mean vectors must be assigned to regression classes to prevent numerical instability. Numerical instability could also result if too small an amount of adaptation data is used to determine a regression class transformation matrix. It is therefore a good idea to check condition numbers during the calculation of the transformation matrices to prevent adaptation with poorly estimated transformation matrices.

4.5.3 Implementation Issues Specific to this Thesis

The following points must be noted regarding the implementation of the MLLR algorithm in this thesis:

- The adaptation is static, i.e. all of the available adaptation data is gathered before the model is reestimated. A separate and independent data set is used for testing, except for some of the experiments on the TIMIT corpus.
- Adaptation is supervised, i.e. the transcription of the adaptation data is known.
- Only the mean vectors are reestimated. Parameters that are not reestimated remain the same as that of the SI model.
- The regression class tree is determined from phonetic knowledge.
- Transformation matrices are only estimated for nodes in the regression class tree for which there were enough observed data. Mean vectors are adapted by the transformation matrix of their node (or the node closest to their node) in the regression class tree, or, if no regression matrix was estimated for any of the parent nodes of the base regression class of the mean vector, the mean vector is not adapted and remains the same.

4.6 Strengths and Weaknesses of MLLR

Every adaptation method has its pros and cons. These arise from the base assumptions of the method, implications of the underlying equations, computational aspects or the interaction of the adaptation method with the specific speech modelling problem. In this section the good and bad points regarding MLLR adaptation will be examined.

One of the base assumptions is that the mean vectors tied to the same regression class undergo a similar transformation when adaptation is performed for a new speaker. This assumption is a dual-edged sword. Firstly it affords MLLR a method to adapt parameters for which no data was observed, making it a powerful adaptation method. On the other hand, correctly clustering mean vectors is not an easily addressed problem.

When assigning mean vectors to a cluster, how similar must their behaviour be for the cluster transformation to be valid? For a global regression class covering a wide scope of mean vectors, the single linear transformation will tend to capture an average translation, scaling and rotation for the mean vectors. When adapting for a new speaker, such a broad similarity would be displayed for most of the mean vectors, and adaptation is likely to yield good results. For a finer regression class covering a narrow scope of mean vectors, the chance is greater that there is no broad linear transformation similarity among them. Thus, when using smaller regression clusters, the similarities between mean vectors must be much greater for estimation of a valid transformation. To summarise: If regression classes do not bind mean vectors that are expected to undergo a similar transformation, then adaptation will be poor, and the degree to which their behaviour must be similar is dependent on the number and scope of mean vectors assigned to the regression class.

Robust estimation of parameters is another issue related to MLLR performance, and the amount of data needed for robust estimation depends as much on the specific MLLR clustering scheme as on the amount of adaptation data available. The issues of clustering and robust estimation were treated in Section 4.4.

Another problem area for MLLR adaptation that depends on the clustering scheme as well as on the expected data of the speech recognition task, is the "balance" of the regression tree. A regression class tree will be balanced when the data available to estimate the transformation matrix is evenly spread over the mean vectors in that regression class. An unbalance in a regression tree is either due to poor tree design, or due to a highly uneven spread of observed data for the model. These problems are closely linked, as both have to do with the expected amount of observation data for the nodes in the regression tree.

For instance, assume a regression class tree has been built on phonetic knowledge of an expected data set. A certain node P in this tree consists of two child nodes A and E,

where A is the set of all phonemes with an a-like sound, and E is a set of all phonemes with an e-like sound. If the number of a-like sounds are much larger than the amount of e-like sounds, the tree has a designed imbalance. Should an even amount of data be available for all sounds, the following situation may occur: the A node has enough adaptation data and the mean vectors assigned to it are transformed. The E node, however, does not have enough adaptation data for robust estimation, and the mean vectors assigned to it are transformed by the transformation matrix of parent P. P uses both the data for the A and E nodes to form the transformation matrix. Most of the adaptation data emanates from the a-like sounds, and so an an a-like transformation will be forced on e-like sounds, resulting in poor adaptation.

An imbalance like the above may have been tailored into the tree if the small E set is expected to have many observed instances for the e-like sounds as opposed to a larger A set with fewer observed instances of a-like phonemes. The structural "imbalance" in the tree is used here to compensate for the expected data imbalance resulting from the more commonly found e-like sounds in the adaptation data. There are many ways in which an imbalance can occur, all of them resulting in a transformation unduly favouring a set of mean vectors at the expense of another set of mean vectors. The adaptation data that will be expected will thus have a great effect on choosing the regression tree in order to avoid any major imbalances.

Furthermore, MLLR is prone to numerical instability when very small amounts of adaptation data are available. It is therefore wise to check condition numbers when MLLR is to be used under such conditions.

Finally, we state the computational requirements for MLLR.[4] Assume that a speaker model consists of H HMMs (each representing a phoneme), S states per HMM and M mixture components per state and n-dimensional observation vectors. Furthermore, assume that there are A regression classes, and that each applies to $\frac{HSM}{A}$ states. If each observation is T frames long, and P is the number of phonemes observed, the computational requirements for MLLR adaptation (excluding the Baum-Welch or Viterbi segmentation and determination of accumulators) are then approximately [51]:

[4] Note that the computational requirements for the initial Baum-Welch or Viterbi segmentation is not included.

- **Memory:** $O(HSMn^3)$
- **Computations:** $O\left(SM(SMT + HAn^2)\right) + O\left(An^4\right)$

These requirements are greater than those required for MAP adaptation. The more regression classes are used, the greater the computational requirements become.

Chapter 5

MLED Adaptation

5.1 Introduction to Eigenvoice Decomposition

Maximum likelihood eigenvoice decomposition (MLED) [28, 40] is a rapid speaker adaptation technique inspired by the use of eigenfaces in human face recognition [25].

In facial recognition, the eigenvectors obtained via PCA (principal component analyses, also known as the Karhunen-Loève transform) of a set of training faces are termed "eigenfaces". A weighted sum of these orthogonal eigenfaces can then be used to represent a human face. Generally, very few of these eigenfaces are necessary to represent a face adequately for recognition purposes, and the resulting dimensional reduction of the recognition problem is normally quite phenomenal.

Kuhn *et al.* [27] postulated that if this concept could be applied to the family of patterns composed of human faces, then the same could be done for human speech. The use of an eigenspace essentially assumes that the inter-speaker variability in a set of SD models may be modelled in a space with lower dimension than the space spanned by the parameters of the original SD models. The first step in creating an eigenspace for speech models entails the creation of well-trained SD models for many individuals. For each SD model in the set, a representative "supervector" is created. A supervector is created by stringing the relevant parameters — i.e. those parameters to be represented and adapted in an eigenspace — of an SD model together in a single vector. Eigenvectors (of the same dimension as the supervectors), referred to as eigenvoices, are then extracted from the set of SD supervectors via PCA. The dominant eigenvoices — those with the largest

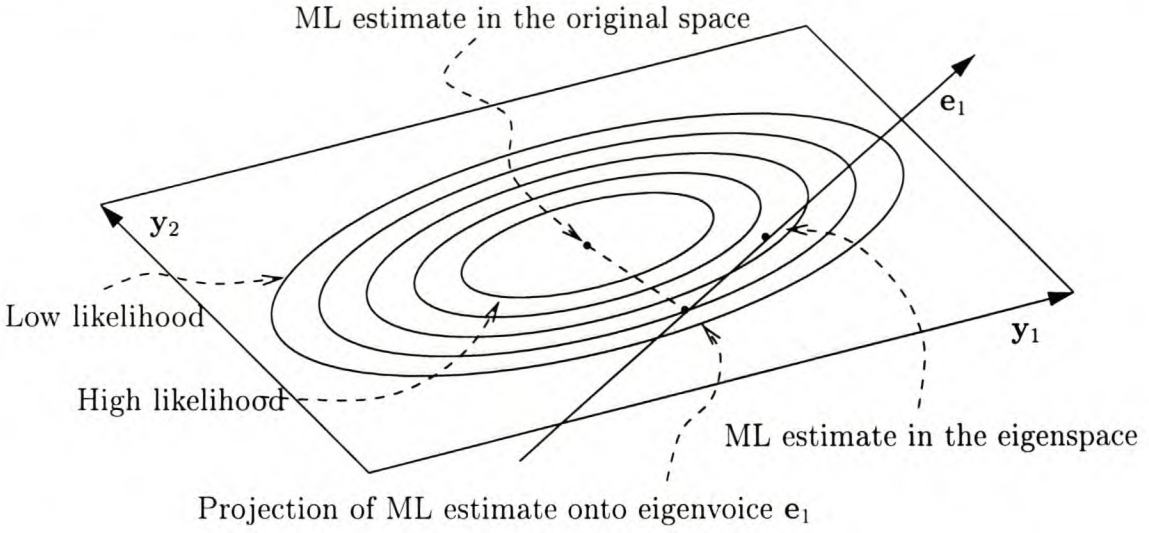

Figure 5.1: A simplified representation of the projection of the ML estimate in the original space onto the eigenspace vs. the ML estimate made in the eigenspace. The original 2-dimensional space is spanned by $\mathbf{y}_1$ and $\mathbf{y}_2$, and the 1-dimensional eigenspace is spanned by $\mathbf{e}_1$. The concentric ellipses on the original plane represent a Gaussian pdf for the 2 model parameters, with ellipses nearer to the centre representing a higher likelihood.

inter-speaker variances associated with them — are then used to span the eigenspace. Typically the inter-speaker variability can be accurately modelled using very few of these eigenvoices. A supervector for a new speaker may now be approximated by using a weighted sum of the dominant eigenvoices.

In itself, the parameter reduction afforded by PCA is already a great benefit when little adaptation data for a new speaker model is available. A new ML or MAP estimate for the speaker model given the adaptation data could simply be formed, and the supervector for the newly estimated model thus obtained can then be projected into the eigenspace and then subsequently projected back into the normal speaker space. This has the effect of doing a maximum likelihood adaptation and then constraining the adaptation to fit in the set of possible speaker models spanned by the eigenspace. Projection, however, is suboptimal for two reasons: One, parameters for which no new ML estimates are available (due to lack of data) are also projected into the eigenspace, and thus play a role in the adaptation. Two — as depicted in Figure 5.1 — the projection of the ML estimate onto the eigenspace does not necessarily represent the highest value of the likelihood

function representable in the eigenspace. The method of forming an ML estimate and then constraining the model to lie in the eigenspace is an optimal solution, as it does not incorporate the *a priori* information we have of the speaker models in the formation of the new estimate.

Kuhn *et al.* and Nguyen [28, 40] analysed the problems inherent in simple projection, and overcame them by developing a new eigenvoice-driven speaker adaptation method: MLED. Assuming that the supervector for a new speaker can be formed by a weighted sum of eigenvoices, a maximum likelihood estimation for the weights given the adaptation data and the current model is possible. This means that the *a priori* data we have for the speaker model — the fact that it must lie in the eigenspace — is a constraint directly incorporated in the adaptation procedure in a maximum likelihood fashion. MLED is thus the method of forming a maximum likelihood reestimation of the speaker model given the initial speaker model, the adaptation data and the eigenspace.

5.2 Basic MLED Adaptation Equations

In MLED, the only parameters that are adapted are mean vectors. The supervector μ created for each of the SD models thus has the following form:

$$\mu = \begin{bmatrix} \mu_1^{(1)} \\ \mu_2^{(1)} \\ \vdots \\ \mu_m^{(s)} \\ \vdots \\ \mu_{M_S}^{(S)} \end{bmatrix} \tag{5.1}$$

where S is the number of states in all the HMM models of a speaker model, and M_s is the number of mixture components in state s. There is thus a single supervector for each speaker, comprising of all the mean vectors in all the states in all the HMMs for the

speaker. An eigenvoice (or eigenvector) $\bar{\mu}$ will have the same form, and will be given by

$$\mathbf{e}_j = \begin{bmatrix} \mathbf{e}_1^{(1)}(j) \\ \mathbf{e}_2^{(1)}(j) \\ \vdots \\ \mathbf{e}_m^{(s)}(j) \\ \vdots \\ \mathbf{e}_{M_S}^{(S)}(j) \end{bmatrix} \quad \text{where} \quad j = 1 \ldots K, \tag{5.2}$$

where K is the number of eigenvoices chosen. In MLED, the new estimation of the supervector $\hat{\mu}$ for a new speaker model is given by a weighted summation of the eigenvoices.

$$\hat{\mu} = \sum_{j=1}^{K} w_j \mathbf{e}_j, \tag{5.3}$$

where w_j is the weight (or eigenweight) associated with eigenvoice j. Alternatively, each of the mean vectors can be calculated separately as a sum of "eigenmeans".

$$\hat{\mu}_m^{(s)} = \sum_{j=1}^{K} w_j \mathbf{e}_m^{(s)}(j). \tag{5.4}$$

Note that the eigenweights are not the eigenvalues. These eigenweights must be chosen so that they maximise the likelihood of the adaptation data.

5.3 Determining the Optimal Eigenweights for MLED Adaptation

The following deduction is for the determination of the set of optimal eigenweights where the speaker model to adapt is a single HMM with mixture Gaussian state pdfs. At the end of the discussion it will be shown how the results are easily extended for the case where a speaker model consists of several HMMs.

5.3.1 Defining the Auxiliary Function

This section briefly reviews the topics treated in Sections 2.5.1 and 2.5.2: reestimation of model parameters in a maximum likelihood framework, the likelihood function to be maximised, and the auxiliary function that can be maximised instead of the likelihood

function. To make a maximum likelihood estimate for the MLED eigenweights, it is necessary to maximise the likelihood of the observed data given the model parameters, $L(O|\lambda)$. $L(O|\lambda)$ is expressed by Equation 2.94, which is restated here for convenience:

$$L(O|\lambda) = \sum_{\theta \in \Theta} \sum_{\zeta \in \Psi_\theta} L(O, \theta, \zeta|\lambda), \tag{5.5}$$

where Ψ_θ is the set of all possible mixture component sequences given the state sequence θ, and ζ is a mixture component sequence in Ψ_θ.

An auxiliary function may be defined such that parameters that maximise it will also increase the value of the likelihood function $L(O|\lambda)$. This auxiliary function $Q(\lambda|\hat{\lambda})$ is given by

$$Q(\lambda, \hat{\lambda}) = \sum_{\theta \in \Theta} \sum_{\zeta \in \Psi_\theta} L(O, \theta, \zeta|\lambda) \log L(O, \theta, \zeta|\hat{\lambda}). \tag{5.6}$$

$L(O|\lambda)$ may now be maximised by iteratively maximising $Q(\lambda|\hat{\lambda})$ and updating the model parameters with the reestimated model parameters, $\hat{\lambda}$.

Expanding the auxiliary function, it is seen that some of the terms are constant, and thus would play no part in the maximisation of the auxiliary function. The expanded auxiliary function with constant terms removed is given by Equation 2.105. It is this function that will be maximised with respect to the eigenweights (w_j, $j = 1 \ldots K$), and it is also restated here for convenience

$$Q(\lambda, \hat{\lambda}) = -\frac{1}{2} L(O|\lambda) \sum_{s=1}^{S} \sum_{m=1}^{M_s} \sum_{t=1}^{T} \gamma_m^{(s)}(t) \left[n \log(2\pi) + \log|C_m^{(s)}| + h_m^{(s)}(\mathbf{o}_t) \right], \tag{5.7}$$

where

$$h_m^{(s)}(\mathbf{o}_t) = \left(\mathbf{o}_t - \hat{\mu}_m^{(s)}\right)' C_m^{(s)-1} \left(\mathbf{o}_t - \hat{\mu}_m^{(s)}\right) \tag{5.8}$$

$$= \left(\mathbf{o}_t - \sum_{j=1}^{K} w_j \mathbf{e}_m^{(s)}(j)\right)' C_m^{(s)-1} \left(\mathbf{o}_t - \sum_{j=1}^{K} w_j \mathbf{e}_m^{(s)}(j)\right). \tag{5.9}$$

5.3.2 Maximising the Auxiliary Function

To estimate the eigenweights, the set of equations found by differentiating the auxiliary function $Q(\lambda, \hat{\lambda})$ with respect to each of the eigenweights and equating the result to zero

must be solved simultaneously. The eigenvoices are orthogonal, and so it is reasonable to assume that the eigenweights will be independent of one another ($\frac{\partial w_i}{\partial w_j} = 0, i \neq j$).

$$\frac{\partial Q}{\partial w_i} = 0, \quad i = 1 \ldots K. \tag{5.10}$$

Differentiating with respect to each of the eigenweights and equating to zero we find

$$\frac{\partial Q(\lambda, \hat{\lambda})}{\partial w_i} = -\frac{1}{2} L(O|\lambda) \sum_{s=1}^{S} \sum_{m=1}^{M_s} \sum_{t=1}^{T} \gamma_m^{(s)}(t) \left[\frac{\partial}{\partial w_i} h_m^{(s)}(\mathbf{o}_t) \right] = 0, \quad i = 1 \ldots K, \tag{5.11}$$

thus

$$\sum_{s=1}^{S} \sum_{m=1}^{M_s} \sum_{t=1}^{T} \gamma_m^{(s)}(t) \left[\frac{\partial}{\partial w_i} h_m^{(s)}(\mathbf{o}_t) \right] = 0, \quad i = 1 \ldots K. \tag{5.12}$$

5.3.2.1 Determining the Derivative of $h_m^{(s)}(\mathbf{o}_t)$ with Respect to an Eigenweight

The new estimate for a mean vector is the weighted sum of eigenmeans. Because the eigenweights are assumed to be independent of one another the following holds:

$$\frac{\partial}{\partial w_i} \hat{\mu}_m^{(s)} = \frac{\partial}{\partial w_i} \sum_{j=1}^{K} w_j \mathbf{e}_m^{(s)}(j) = \mathbf{e}_m^{(s)}(i). \tag{5.13}$$

Now

$$\begin{aligned}
\frac{\partial}{\partial w_i} h_m^{(s)}(\mathbf{o}_t) &= \frac{\partial}{\partial w_i} \left(\mathbf{o}_t - \hat{\mu}_m^{(s)} \right)' C_m^{(s)-1} \left(\mathbf{o}_t - \hat{\mu}_m^{(s)} \right) && (5.14) \\
&= \frac{\partial}{\partial w_i} \left\{ \mathbf{o}_t' C_m^{(s)-1} \mathbf{o}_t - 2\mathbf{o}_t' C_m^{(s)-1} \hat{\mu}_m^{(s)'} + \hat{\mu}_m^{(s)'} C_m^{(s)-1} \hat{\mu}_m^{(s)} \right\} && (5.15) \\
&= -2\mathbf{o}_t' C_m^{(s)-1} \left(\frac{\partial}{\partial w_i} \hat{\mu}_m^{(s)} \right) + \frac{\partial}{\partial w_i} \left(\hat{\mu}_m^{(s)'} C_m^{(s)-1} \hat{\mu}_m^{(s)} \right). && (5.16)
\end{aligned}$$

Substituting Equation 5.13 for the adapted mean vectors gives

$$= -2\mathbf{o}_t' C_m^{(s)-1} \mathbf{e}_m^{(s)}(i) + \left(\frac{\partial}{\partial w_i} \hat{\mu}_m^{(s)'} \right) C_m^{(s)-1} \hat{\mu}_m^{(s)} + \hat{\mu}_m^{(s)'} C_m^{(s)-1} \left(\frac{\partial}{\partial w_i} \hat{\mu}_m^{(s)} \right). \tag{5.17}$$

Since scalars are equal to their transpose, so

$$= -2\mathbf{o}_t' C_m^{(s)-1} \mathbf{e}_m^{(s)}(i) + 2 \left(\frac{\partial}{\partial w_i} \hat{\mu}_m^{(s)'} \right) C_m^{(s)-1} \hat{\mu}_m^{(s)}. \tag{5.18}$$

Using Equation 5.13 again

$$= 2 \left[-\mathbf{o}_t' C_m^{(s)-1} \mathbf{e}_m^{(s)}(i) + \mathbf{e}_m^{(s)'}(i) C_m^{(s)-1} \sum_{j=1}^{K} w_j \mathbf{e}_m^{(s)}(j) \right]. \tag{5.19}$$

Substituting the results of Equation 5.19 in Equation 5.12 gives

$$\frac{\partial Q}{\partial w_i} = 0 = \sum_s \sum_m \sum_t \gamma_m^{(s)}(t) \left\{ -\mathbf{o}_t' C_m^{(s)-1} \mathbf{e}_m^{(s)}(i) + \mathbf{e}_m^{(s)'}(i) C_m^{(s)-1} \sum_{j=1}^{K} w_j \mathbf{e}_m^{(s)}(j) \right\} \quad (5.20)$$

or, equivalently

$$\sum_s \sum_m \sum_t \gamma_m^{(s)}(t) \mathbf{e}_m^{(s)'}(i) C_m^{(s)-1} \mathbf{o}_t$$
$$= \sum_s \sum_m \sum_t \gamma_m^{(s)}(t) \sum_{j=1}^{K} w_j \mathbf{e}_m^{(s)'}(j) C_m^{(s)-1} \mathbf{e}_m^{(s)}(i), \quad i = 1 \ldots K. \quad (5.21)$$

Equation 5.21 defines the set of linear equations that must be solved in order to obtain optimal set of eigenweights. This set of equations can also be expressed as

$$\mathbf{v} = Q\mathbf{w}, \quad (5.22)$$

where $\mathbf{w}$ is the K-dimensional vector of eigenweights

$$\mathbf{w} = [w_1, w_2, \ldots, w_K]', \quad (5.23)$$

and $\mathbf{v}$ is an K-dimensional vector given by

$$\mathbf{v} = \begin{bmatrix} \sum_s \sum_m \sum_t \gamma_m^{(s)}(t) \mathbf{e}_m^{(s)'}(1) C_m^{(s)-1} \mathbf{o}_t \\ \sum_s \sum_m \sum_t \gamma_m^{(s)}(t) \mathbf{e}_m^{(s)'}(2) C_m^{(s)-1} \mathbf{o}_t \\ \vdots \\ \sum_s \sum_m \sum_t \gamma_m^{(s)}(t) \mathbf{e}_m^{(s)'}(K) C_m^{(s)-1} \mathbf{o}_t \end{bmatrix}, \quad (5.24)$$

and every element q_{ij} of the $K \times K$ matrix Q is given by

$$q_{ij} = \sum_s \sum_m \sum_t \gamma_m^{(s)}(t) \mathbf{e}_m^{(s)'}(j) C_m^{(s)-1} \mathbf{e}_m^{(s)}(i). \quad (5.25)$$

Thus

$$\mathbf{w} = Q^{-1}\mathbf{v}. \quad (5.26)$$

The set of eigenweights can thus be found by inverting matrix Q and multiplying it by $\mathbf{v}$ or by Gaussian elimination. The newly estimated mean vectors or supervector can now be calculated using Equation 5.4 or 5.3 respectively.

5.4 Creation of the Eigenvoices from a Set of Speaker Dependent Models

We now apply PCA to the set supervectors representing the SD (speaker dependent) models available to create the eigenspace. Ideally, these SD models should have the same structure in terms of the number of states, the links between the states and the number of mixture components per state. The corresponding state transition probabilities and covariance matrices in different SD models should be the same, so that only the mean vectors model differences between different speakers. This can be done by first pooling all the data for all the speakers and training a speaker independent (SI) model. All model parameters except the mean vectors can now be held fixed, and an SD model for each speaker can be created using EM training algorithms. Should little training data be available for these SD models, the well-known MLLR method can be used for model estimation.

A supervector for each of these SD models (where each of these SD models comprises of several HMMs, and each HMM represents a phoneme) is now created, and PCA is then applied to the set of supervectors to obtain the eigenvectors (eigenvoices) of the set.

5.4.1 Principal Component Analysis

As this is a well-known method, only a summary of the relevant equations and their import will be given. PCA is also referred to as the Karhunen-Loève transform (KLT).

PCA is used for the purpose of parameter reduction, and a transform is required for a vector in the current space to a vector in a space with (possibly) fewer dimensions. If $\mathbf{y}$ is a vector of dimension m in the original space, and $\mathbf{x}$ is the projection of $\mathbf{y}$ in the new space, then

$$\mathbf{y} = U\mathbf{x}. \tag{5.27}$$

If the number of data vectors $\mathbf{y}$ is n, then they can be combined to form the $m \times n$ data

matrix Y. The eigen decomposition of the correlation matrix R_y is then given by

$$R_y U = \frac{1}{n} YY'U = U\Lambda = \begin{bmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_m \end{bmatrix} \begin{bmatrix} \lambda_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \lambda_m \end{bmatrix}, \tag{5.28}$$

where U is the $m \times m$ eigenvector matrix, and Λ is the $m \times m$ diagonal eigenvalue matrix of the correlation matrix.

The eigenvectors $\mathbf{u}$ of the correlation matrix are the columns of U and the eigenvalues λ are contained in the diagonal eigenvalue matrix Λ. The number of non-zero eigenvalues is $\min(m, n)$. Eigenvectors are orthogonal, and can be made orthonormal. For an orthonormal matrix the following holds:

$$U^{-1} = U' \quad \text{or} \quad U^{-1}U = U'U = I. \tag{5.29}$$

The correlation matrix of X (none of the dimensions have been discarded yet) now has the property that it is diagonal and equal to the eigenvalues of the correlation matrix of the y-vectors.

$$\frac{1}{n} XX' = \frac{1}{n}(U'Y)(U'Y)' = \frac{1}{n} U'(YY'U) = U'(U\Lambda) = I\Lambda = \Lambda. \tag{5.30}$$

Equation 5.27 can now be interpreted as the representation of $\mathbf{y}$ as the weighted sum of eigenvectors, where the weights are the elements of the $\mathbf{x}$ vector. Dimension reduction is possible if we represent $\mathbf{y}$ using a only subset of the eigenvectors and $\mathbf{x}$ element values. The error in this approximation of $\mathbf{y}$ is minimised in a mean-squared sense if the discarded eigenvectors in the approximation are those that are associated with the smallest eigenvalues.

A fixed number of eigenvalues can be retained for the approximation of a $\mathbf{y}$ vector. Alternatively a rough idea of the error in the approximation can be obtained by determining the sum of the eigenvalues that are retained, and expressing it as a percentage of the sum of all the eigenvalues. In practice, the first few eigenvectors represent the bulk of the sum of eigenvalues, so that the rest of the eigenvectors may be discarded. The result is a great reduction in the dimension of the training problem, as the dimension of $\mathbf{x}$ becomes the same as the number of eigenvectors used in the approximation.

5.4.2 Using SVD to Determine the Eigenvectors and Values for PCA

It is not necessary to compute the correlation matrix of the data in order to obtain the eigenvectors for PCA. Singular value decomposition (SVD) can be used directly on the data matrix Y to obtain the necessary matrices. The SVD of a data matrix Y is given by

$$\underset{m \times n}{Y} = \underset{m \times m}{U} \quad \underset{m \times n}{\Sigma} \quad \underset{n \times n,}{V'} \tag{5.31}$$

where U is the $m \times m$ left singular matrix, Σ an $m \times n$ diagonal matrix with the nonnegative singular values $\sigma_j, j = 1 \ldots \min(m, n)$ arranged in descending order of magnitude on the diagonal, and V is the the $n \times n$ right singular matrix.

Both U and V are orthonormal matrices. By rearranging Equation 5.31, it is readily shown that the columns of U are the eigenvectors of YY', and the columns of V are the eigenvectors of $Y'Y$. Thus, if Equation 5.31 is substituted for Y in Equation 5.28 then

$$\frac{1}{n}YY'U = \frac{1}{n}U\Sigma V'(U\Sigma V')'U = \frac{1}{n}U\Sigma V'V\Sigma U'U = \frac{1}{n}U\Sigma^2 = U\Lambda. \tag{5.32}$$

The non-zero eigenvalues $\lambda_j, j = 1 \ldots \min(m, n)$ of the correlation matrix R_y are related to the non-zero singular values of the SVD of Y by

$$\lambda_j = \frac{1}{n}\sigma_j^2 \quad , j = 1 \ldots \min(m, n). \tag{5.33}$$

The rank of matrix Y is the same as the number of nonzero singular values. Also, $r \leq \min(m, n)$. If the rank of Y is r, then Equation 5.31 can be reduced to

$$\underset{m \times n}{Y} = \underset{m \times r}{U_+} \quad \underset{r \times r}{\Sigma_+} \quad \underset{r \times n,}{V'_+} \tag{5.34}$$

where Σ_+ is the diagonal matrix containing the r nonzero singular values of Y, and U_+ and V_+ are the first r columns of U and V. U_+ and V_+ form the orthonormal basis for the columns and rows of Y respectively, whereas the remaining columns of U and V form the orthonormal basis for the left and right null-spaces of Y respectively. As we are not interested in the null-spaces, Equation 5.34 will be used in preference of Equation 5.31.

5.4.3 Utilising SVD to Reduce Memory Requirements

Normally the supervectors and eigenvectors used in MLED are very large. For instance a simple SD model used for the ISOLET comprised of the following: 26 HMMs (one for each letter) with 6 states (not including null states) for each of these HMMs. One HMM for the silence model with 3 states (not including null states). There is one Gaussian pdf per state, and the mean vector is 18-dimensional. The dimension for each of the supervectors (and eigenvectors) is then

$$[(26 \times 6) + (1 \times 3)] \times 1 \times 18 = 2862. \tag{5.35}$$

When a mixture Gaussian with 6 mixture components is used instead of a single Gaussian, the dimension of a supervector escalates to 17172. For the ISOLET corpus, there are 150 speakers. So even when all the speakers are used, $n = 150 \ll m = 2862$. For most speech databases, the number of speakers will be less than the dimension of the supervector for a SD model.

When computing the eigenvectors for a set of vectors with large dimension, computer memory problems may be encountered. The correlation matrix R_y has dimension $m \times m = (17172 \times 17172)$ when mixture Gaussian pdfs are employed. Using 32-bit floating point numbers, this is equivalent to a 1125 MB (31 MB for the single Gaussian case) memory requirement for the storage of the correlation matrix alone. Imagine the effect more complex SD models for larger speech corpora would have on the memory requirements for the computation of the eigenmeans.

Alleviating the memory burden was achieved by utilising a method used in face recognition [36], where the dimension of the data vectors (the number of points in an image) is often very large. Using this method of computation, the memory requirements may be drastically reduced, as the eigenvectors describing the null-spaces of Y are not of interest. The method is as follows:

Note that the SVD Equation 5.34 can be substituted for Y in the eigenvector decomposition of the "reverse" correlation matrix $Y'Y$.

$$Y'YV_+ = \left[(U_+\Sigma_+V_+')'(U_+\Sigma_+V_+')\right] V_+. \tag{5.36}$$

V_+ and U_+ are orthonormal, so

$$Y'YV_+ = V_+\Sigma_+^2. \tag{5.37}$$

The left singular matrix can now be determined by rearranging Equation 5.34

$$\underset{m \times r}{U_+} = \underset{m \times n}{Y} \underset{n \times r}{V_+} \underset{r \times r.}{\Sigma_+^{-1}} \tag{5.38}$$

When $n \ll m$, as in this thesis, the memory savings are considerable. This is due to the fact that the eigenvectors of the smaller matrix $Y'Y$ are determined instead of those of the larger matrix YY'. Only the necessary eigenvectors are computed, so it is unnecessary to bother with extra computations and memory for eigenvectors that span the null-spaces. When $n < m$ the above method will be used to obtain the eigenvectors, and when $m < n$ (unlikely in this application) the normal SVD or KLT may be employed. A small price is paid for the reduction in memory needs, as accuracy is impaired due to rounding errors when $Y'Y$ is computed. According to Muller *et al.* [36], the smallest singular values are affected most, which is most fortunate. Obviously, if we know we will only need $c \leq r$ eigenvectors, then we can simply reduce the number of columns of U_+ to the number that needs to be computed.

$$[\mathbf{u}_1 \dots \mathbf{u}_c] = Y[\mathbf{v}_1 \dots \mathbf{v}_c] \begin{bmatrix} \sigma_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \sigma_c \end{bmatrix}^{-1}. \tag{5.39}$$

5.5 The Effect of the Eigenspace on Recognition Performance

The following is a summary of the findings of Nguyen [40] and Westwood [51] on the dimension of the eigenspace and post-adaptation recognition performance.

Nguyen's experiments were on the relatively simple ISOLET [37] speech corpus, which consists of the letters of the alphabet spoken in isolation. A six (non-null) state HMM with a six mixture component GMM representing each state pdf was used to represent each of the 26 letters. The database has speech from 120 speakers. The data from 100 speakers were used to train 100 SD models from which an eigenspace was extracted via PCA. Nguyen used up to 30 eigenvoices, and found that post-adaptation recognition rate increased as the number of eigenvoices were increased. Also, an increase in the amount of adaptation data resulted in an increase in recognition rate. The maximum amount of

available adaptation data for a new speaker was a single utterance of each of the 26 letters of the alphabet.

Westwood used the Darpa corpus [38], which consists of 30 utterances for each of 109 speakers. Westwood used several modelling schemes to represent monophones of triphones, where the simplest speech models consisted of monophones modelled by HMMs with single Gaussian state pdfs, and the most complex models consisted of triphones represented by HMMs with 6 mixture component GMM state pdfs. All HMMs had three non-null states. 30 sentences were available for each speaker, 10 of which were used for adaptation and 20 for scoring. Westwood found that, in general, the relative amount of eigenvoices (relative to the number of model parameters) needed to account for half the speech variability in the set of SD models was less for the more complex modelling schemes. He attributed the decrease in variability to the lack of training data needed to form good SD model estimates for the more complex models, as the lack of training data makes less differentiation possible between SD models.

Westwood used three kinds of subspace bases: eigenvoices extracted via PCA from the correlation matrix of the SD supervectors, eigenvoices extracted via PCA from the covariance matrix, and bases comprising of the original SD supervectors. The last implementation indicates that the MLED and WP eigenvoice techniques bear great similarities to CAT and other speaker clustering methods. He found that recognition performance generally increased as the amount of PCA-extracted eigenvoices were increased, up to fifteen eigenvoices, after which the post-adaptation recognition rate decreases and increases erratically. Also, when few eigenvectors where used, PCA-extracted eigenvectors result in better performance than SD supervectors, indicating that eigenvoices were indeed capturing much of the inter-speaker variability between the speakers. However, the recognition rate using SD supervectors continues to increase as more supervectors are employed, and does not display the erratic behaviour of PCA-extracted eigenvoices. Westwood ruled out numerical instability as the cause of the erratic adaptation performance, and attributed it to a poorly estimated eigenspace. His findings indicated that only about 15 eigenvectors could be robustly estimated from the models of 109 speakers. He suggested that either a larger SD model set must be used for the robust determination of more than 15 eigenvoices, or that methods other than PCA must be used to extract a space that models the

inter-speaker variability better.

Nguyen found that the recognition rate rapidly increases for MLED as more adaptation data becomes available. However, he only used the ISOLET corpus, and had a maximum number of 26 spoken letters worth of adaptation data with which to adapt. Westwood, on the other hand, had 10 sentences worth of adaptation data, which is much more than an utterance of the alphabet. He experiments indicate that recognition rate for MLED increases until approximately three sentences of adaptation data, after which a plateau is reached and little or no improvement in recognition is achieved for more adaptation data. Though MLLR does not yield as much improvement as MLED for one or two sentences of adaptation data, it does not reach a plateau in recognition rate as MLED does. As the amount of adaptation data increases, MLLR performance increases. For the simple models, MLLR performance surpasses MLED performance after approximately 7 sentences of adaptation data. For the more complex models, MLLR performance is on par with MLED performance after approximately four adaptation sentences, and remains on par. The plateau in MLED recognition performance seems to indicate that the point in the eigenspace representing a speaker may be determined using very little adaptation data, and that using more data to estimate this point results in very little change in recognition performance as we have already reached a maximum likelihood in the eigenspace. The bound on the performance indicates a limit on the accuracy with which fifteen (or less) eigenvoices can represent the true model for a new speaker. Though Nguyen experimented using very little adaptation data, his experiments indicate that when few eigenvoices (one or five) are used, the tempo of improvement in recognition rate as more adaptation data is used becomes less and less. When more eigenvoices (ten) are used, the initial recognition rate is lower for one letter utterance of adaptation data. However, the tempo of improvement in recognition rate is much higher than it is for few eigenvoices, and that the tempo decreases more slowly than it does for fewer eigenvoices.

Based on the experiments and conclusions of Nguyen and Westwood, the following observations are made:

- The performance of MLED is highly dependent on how many eigenvoices are used, and how robustly these eigenvoices were estimated.
- When using PCA-extracted eigenvoices, the factor limiting the robust estimation of

eigenvoices is the number of available SD models. As previously stated, Westwood concluded that only about fifteen eigenvoices can be robustly estimated from a prior set of SD models for 109 speakers.

- When very little adaptation data is available (one to two phoneme utterances), very few (one or two) eigenweights for one or two eigenvoices can be robustly estimated. When more adaptation data is available (one or two sentences), eigenweights for more eigenvoices (ten to fifteen) may be robustly estimated.

- When the amount of adaptation data is more a few phoneme utterances, using more eigenvoices results in higher post adaptation recognition performance.

- Recognition improvement using MLED generally increases as more adaptation data are used, however, a plateau in performance is soon reached, and increases in adaptation data yield no further improvement. Using a limited set of eigenvoices thus result in a limited ability to accurately estimate a new speaker model. The lower the number of eigenvoices, the faster this limit is reached.

5.6 MLED Algorithm Summary

This is a succinct treatment of implementing the MLED algorithm for the adaptation of mean vectors in GMMs in HMMs of a speaker speech model. The treatment will proceed in three steps:

- The algorithm and its prerequisites, divided into online and offline steps.

- Points to bear in mind for general MLED implementations.

- Points to bear in mind specific to the implementation of the algorithm in this thesis.

5.6.1 Algorithm Implementation

The algorithm is split into online and offline steps:

- **Offline Steps:**

- Train an SI model, using a large amount of representative speech data from a representative set of training speakers. This SI model will form the initial model for each new target speaker for which MLED adaptation is to be performed.
- Create a set of robustly trained SD models, using speech data for each speaker in the set of SD models. The initial model for each of these models is the SI model. If enough data are available for robust model estimation, ML estimation of parameters may be used. If enough data for robust ML estimation are not available, speaker adaptation methods may be employed to estimate the SD models. Note that each SD model must have the same structure as the original SI model. Furthermore, only the mean vectors of these SD models may be reestimated. All other parameters must remain identical to that of the SI model.
- Create a supervector from each of the SD models in the prior set of SD models (i.e. those obtained in the previous step), and determine the dominant eigenvoices (Section 5.4).

- **Online Steps:**
  - Use the Viterbi or Baum-Welch algorithms and the SI model to segment the adaptation data obtained from the target speaker. From the segmentation, the following accumulators are obtained:

    $$\sum_{t=1}^{T} \gamma_m^{(s)}(t)\mathbf{o}_t,$$

    and

    $$\sum_{t=1}^{T} \gamma_m^{(s)}(t).$$

  - Determine the matrix Q and vector $\mathbf{v}$, where each of the elements of Q are given by Equation 5.25:

    $$q_{ij} = \sum_s \sum_m \sum_t \gamma_m^{(s)}(t)\mathbf{e}_m^{(s)'}(j)C_m^{(s)-1}\mathbf{e}_m^{(s)}(i).$$

    and each element of $\mathbf{v}$ is obtained from Equation 5.24:

    $$v_i = \sum_s \sum_m \sum_t \gamma_m^{(s)}(t)\mathbf{e}_m^{(s)'}(i)C_m^{(s)-1}\mathbf{o}_t.$$

– Obtain the set of optimal eigenweights by solving Equation 5.26:

$$\mathbf{w} = Q^{-1}\mathbf{v}. \tag{5.40}$$

– Using the eigenweights determined in the previous step, each new mean vector is determined from the (eigen-)weighted sum of eigenmeans (see Equation 5.4):

$$\hat{\mu}_m^{(s)} = \sum_{j=1}^{K} w_j \mathbf{e}_m^{(s)}(j).$$

5.6.2 General Implementation Issues

The following points must be noted regarding the implementation of the MLED algorithm in general:

- The SI model and the speaker models obtained by MLED adaptation must have the same structure, i.e. the same number of HMMs, states, Gaussian pdfs in GMMs, and state transition probability links. This does not mean that the HMMs in a single speaker model must all be identical, or that HMMs for different speakers must have identical parameter values, only that the HMMs must have similar structure.

- The prior mean vectors $\mu_m^{(s)}$ used during the calculation of the new mean vectors are those of the SI model for the first MLED iteration. If multiple adaptation iterations are used, the prior mean vectors are those trained during the previous adaptation run.

- MLED is only used for the adaptation of mean vectors. Parameters that are not part of mean vectors are not adapted, and remain the same is those in the initial SI model.

- Numerical instability can arise if too many eigenweights have to estimated from extremely little adaptation data. It is therefore prudent to check condition numbers when the Q matrices are inverted to help prevent poor adaptation.

5.6.3 Implementation Issues Specific to this Thesis

The following points must be noted regarding the implementation of the MLED algorithm in this thesis:

- The adaptation is static, i.e. all of the available adaptation data is gathered before the model is reestimated. A separate and independent data set is used for testing.
- Adaptation is supervised, i.e. the transcription of the adaptation data is known.
- Only the mean vectors are reestimated. Parameters that are not reestimated remain the same as that of the SI model.
- In order to test the performance of MLED adaptation against MLLR adaptation, the covariance matrices in the Gaussian pdfs in the speaker speech models are limited to being diagonal matrices.

5.7 Strengths and Weaknesses of MLED

All adaptation methods have strengths and weaknesses, which are important to consider when selecting an adaptation method for a specific task. This section discusses the positive and negative attributes of MLED, starting with the former:

- **Pooling all observed data:** MLED makes use of all the observed data for the entire speaker model when adapting a parameter. Of the methods treated, only the eigenvoice techniques (and a global MLLR implementation) have this ability. The observed data for all parameters are thus pooled before any attempt at parameter reestimation is made. Pooling all the observed data allows MLED to make very robust estimates with very little adaptation data.
- **Extreme degree of parameter reduction:** Neither of the Bayesian reestimation methods (ML and MAP) has any form of parameter reduction, so unseen parameters cannot be adapted.

  MLLR can be employed with varying degrees of parameter reduction, depending on the number of regression classes employed. MLLR is thus a rapid adaptation technique. Leggetter and Woodland [34] found that using full linear transformation matrices (the T_α matrix in Equation 4.2) is preferable to using diagonal transformation matrices. Thus, every regression matrix W_α requires the calculation of $n \times (n + 1)$ regression parameters. Also, the parameter reduction is not always

balanced (except when using a global regression class), as some regression classes will have more mean vectors than others.

MLED gives us the greatest degree of parameter reduction of all the treated methods. Both Nguyen [40] and Westwood [51] found that for a SD speaker set consisting of about 100 speakers, only about 15 eigenvoices may be robustly extracted. Thus, even the largest MLED implementations require only the calculation of 15 eigenweights, which is significantly fewer parameters than even global MLLR. The only other adaptation methods that can compete with such extreme parameter reduction capabilities are WP, an eigenvoice method, and speaker clustering methods, such as CAT.

MLED also has the following negative aspects:

- **Dependency on robustly estimated eigenvoices:** MLED is completely dependent on the extracted eigenvoices. MLED adaptation improves the recognition rate when more eigenvoices are used, but only if these eigenvoices are robustly estimated. For the eigenvoices to be robustly estimated, a large set of well-trained speaker dependent models are required. From the experiments in this thesis there seems to be a dependency of MLED performance on the quality (in terms of recognition rate) of the set of prior SD models. The poorer the performance of the SD models, i.e. the closer the recognition rates are to those achieved with the SI model, the less the differentiation between the prior SD models will be. The less the differentiation, the smaller the inter-speaker variance captured by the set of SD models will be. Therefore, the number of directions of inter-speaker variance extracted via the KLT will be less, resulting in poorer and fewer estimated eigenvoices.

- **Computational complexity:** Assume that a speaker model consists of H HMMs (each representing a phoneme), S states per HMM and M mixture components per state and n-dimensional observation vectors. Furthermore, assume that K eigenvoices are used. If each observation is T frames long, and P is the number of phonemes observed, the computational requirements for MLED adaptation (excluding the Baum-Welch or Viterbi segmentation and determination of accumulators) are then approximately [51]:

- **Memory:** $O(KHSMn)$
- **Computations:** $O\left(SM(SMT + H(Kn)^2)\right)$

The memory requirement of MLED is greater than that of MAP, but less than that of MLLR. To accurately compare the computational requirements of MLLR and MLED, the model complexity, number of eigenvoices and regression classes used and feature vector dimension are needed. Generally, the following holds: For simple models (where HSM is small), the computational requirement of MLED is less than that required for MLLR, but for complex models (where HSM is large) the computational requirements for MLED is greater than that required for MLLR.

- **Not convergent to the ML estimate:** Methods such as MAP and RMP converge to the ML estimate as the amount of available adaptation data is increased. MLED does not converge (unless the number of eigenvoices is the same as the number of dimensions of a SD supervector), and neither does MLLR. Experiments by Westwood [51] indicate that MLED can improve recognition performance when very little adaptation data are available. The recognition performance increases as the amount of adaptation data increases, but soon reaches a plateau. MLLR tends to have poorer performance than MLED for little adaptation data, but eventually passes the plateau MLED attains as more adaptation data are added.

- **Cannot be employed adaptively:** It is not likely that MLED could be applied adaptively, as it is dependent on a large set of SD models before it can train any new SD models.

The positive and negative attributes of MLED were valuable when planning the experiments described in this thesis, as well as for the interpretation of the results. This section concludes the discussion of MLED, and the next chapter treats WP — a new eigenvoice adaptation method.

Chapter 6

Weighted Projection

6.1 Introduction to Weighted Projection

Weighted projection is an adaptation technique employing eigenvoices introduced by Westwood in 1999 [51]. The extraction of eigenvoices and the estimation of a new mean vector as a weighted sum of eigenvoices is identical to the MLED approach. The estimation of the optimal eigenweights, however, is accomplished using a projection technique.

As stated in the MLED section, simply taking the maximum likelihood estimate in the original model space and then projecting to the eigenspace is not guaranteed to result in a maximum likelihood estimate in the eigenspace. As speaker adaptation techniques are normally applied when relatively little data are available, many of the parameters in the original space will not have a maximum likelihood estimate, which will necessitate using their initial values during the projection. By doing this something is inferred about the model that is not known, i.e. we have ML estimates for all the parameters in the original space when we do not.

Instead of using normal projection to obtain eigenweights (discounted by Nguyen [40] for good reason), Westwood does the following:

An ML estimate of parameters is formed, and a supervector $\tilde{\mu}$, is created for this ML estimate. The supervector is then mapped via a linear transformation T to a new space, $\mathcal{T}$. The mapping is such that each supervector element (an ML-estimated parameter) is divided by its variance and multiplied by a factor dependent on the amount of observed data for the parameter. The mapping accomplishes two things: One, parameters with no

observed data map to the null vector in $\mathcal{T}$. Two, parameters with lower variance receive preference over those with high variance in the new space $\mathcal{T}$.

The mapped ML-estimated supervector is now constrained to lie in the eigenspace by the following method: The eigenvectors are mapped using the same transformation as the one applied to $\tilde{\mu}$. $T\hat{\mu}$, the closest point in the mapped eigenspace to the mapped $T\tilde{\mu}$, is found. $\hat{\mu}$ then represents the new WP estimate for the speaker, and the desired eigenweights can be determined from it.

The following section explains the difference between simple projection and the WP adaptation method in more detail, and states the equation that must be solved in order to obtain eigenweights using WP.

6.2 From Simple Projection to Weighted Projection

First we will introduce the spaces used in WP: $\mathcal{D}$ is the original space for a supervector, $\mathcal{K}$ is the eigenspace, $\mathcal{T}$ is the space onto which the original supervector is mapped via the linear transform T, and $T(\mathcal{K})$ is the mapped eigenspace.

When the simple projection for eigenvoice adaptation is used, we are essentially finding the vector $\hat{\mu}$ in the eigenspace $\mathcal{K}$ that is closest to the maximum likelihood estimate $\tilde{\mu}$ in the original space $\mathcal{D}$. Thus, if $\mu_{\mathcal{K}}$ is any vector in $\mathcal{K}$, then

$$\forall \mu_{\mathcal{K}} \in \mathcal{K}\ ,\ \|\tilde{\mu} - \hat{\mu}\| \leq \|\tilde{\mu} - \mu_{\mathcal{K}}\|. \tag{6.1}$$

With weighted projection, a linear transform T is applied to the ML-estimated supervector in the original space, and to the set of eigenvectors. The original space $\mathcal{D}$ is then mapped to the space $\mathcal{T}$, and the eigenspace is mapped to the space $T(\mathcal{K})$. It should be noted that, as $\mathcal{K}$ represents a subspace of $\mathcal{D}$, $T(\mathcal{K})$ represents a subspace of $\mathcal{T}$. Having performed the necessary mappings, we wish to find the vector $T\hat{\mu}$ in the transformed eigenspace $T(\mathcal{K})$ that is closest to the T-transformed ML estimate, $T\tilde{\mu}$. Thus, if $\mu_{\mathcal{K}}$ is any vector in the eigenspace then

$$\forall \mu_{\mathcal{K}} \in \mathcal{K}\ ,\ \|T\tilde{\mu} - T\hat{\mu}\| \leq \|T\tilde{\mu} - T\mu_{\mathcal{K}}\|. \tag{6.2}$$

In order for WP to be effective, the linear transform T is chosen to satisfy two criteria:

- T is designed so that parameters that do not have ML estimates in the original space map to the zero vector in the transformed supervector space $\mathcal{T}$.
- Transform T must also satisfy $\forall \mathbf{x} \in \mathcal{K}$, $T(\mathbf{x}) = \mathbf{0} \Rightarrow \mathbf{x} = \mathbf{0}$. This means that none of the vectors in $\mathcal{K}$ are mapped to the zero vector in $\mathcal{T}$, so that the image of the basis vectors of $\mathcal{K}$ (i.e. the eigenvoices) form a set of bases for the transformed eigenspace $T(\mathcal{K})$.

Before treating the derivation of optimal WP estimates, it is necessary to address the choice of linear transformation T. This is done in the following section.

6.3 The Choice of the Linear Transformation T

The zeroth-order accumulator and first-order accumulator may be obtained from a Baum-Welch or Viterbi segmentation. The zeroth-order accumulator $A_0(s, m)$ is given by

$$A_0(s, m) = \sum_{t=1}^{T} \gamma_m^{(s)}(t), \tag{6.3}$$

and the first-order accumulator by

$$A_1(s, m) = \sum_{t=1}^{T} \gamma_m^{(s)}(t)\mathbf{o}_t. \tag{6.4}$$

The robustness of an estimate is thus related to the zeroth-order accumulator, as this gives an indication of how much data there is to estimate the mean vector of the pdf in state s and mixture component m. The higher the value of the zeroth-order accumulator, the more robust the estimate on the data is hoped to be. Thus a weight function $\omega : R \rightarrow R$ is defined that maps the zeroth-order accumulator to a weight for the parameter (in this case the mean vector) to be estimated. The properties of the weight function will be:

- $\omega(0) = 0$
- ω is a non-strictly monotonically increasing function

The transform T may now be defined as the combination of two transformations, D — a decorrelation of the original space — and Ω — a transform based on the weighting function, such that $T = \Omega D$. Strictly speaking, the decorrelation transform is unnecessary, but it will reduce the weighting in favour of distributions with large variances.

- **Matrix** D: The variance between distributions is not taken into account, and so D will be a block diagonal matrix with symmetrical blocks. Each block on the diagonal thus decorrelates a different distribution. Several options for the decorrelation are possible: the identity matrix I on each block along the diagonal results in no decorrelation, $C_m^{(s)-1/2}$ on each corresponding block will result in component-by-component decorrelation, and $\overline{C}^{-1/2}$, where $\overline{C}$ is the mean of the component covariance matrices, is a more robust decorrelation when the component covariance matrices are poorly estimated. Thus, D could be given by

$$D = \begin{bmatrix} \ddots & 0 & 0 \\ 0 & I_n & 0 \\ 0 & 0 & \ddots \end{bmatrix} \text{ or } \begin{bmatrix} \ddots & 0 & 0 \\ 0 & C_m^{(s)-1/2} & 0 \\ 0 & 0 & \ddots \end{bmatrix} \text{ or } \begin{bmatrix} \ddots & 0 & 0 \\ 0 & \overline{C}^{-1/2} & 0 \\ 0 & 0 & \ddots \end{bmatrix} \tag{6.5}$$

  where n is the dimension of the feature vectors and I_n is the $n \times n$ identity matrix. It should be noted that the above is only a list of of possible decorrelations, as many more forms for D are possible.

- **Matrix** Ω: As the D matrix is block diagonal, it is easier to think of the Ω matrix as being block diagonal too, even though it is only diagonal. Ω is now defined as

$$\Omega = \begin{bmatrix} \ddots & 0 & 0 \\ 0 & \omega\left(A_0(s,m)\right)^{1/2} I_n & 0 \\ 0 & 0 & \ddots \end{bmatrix} \tag{6.6}$$

  where $\omega\left(A_0(s,m)\right)$ is the scalar weight function and I_n is the $n \times n$ identity matrix.

6.4 Proving that $\forall \mu_{\mathcal{K}} \in \mathcal{K}\ ,\ \|T\tilde{\mu} - T\hat{\mu}\| \leq \|T\tilde{\mu} - T\mu_{\mathcal{K}}\|$

As stated before, it is necessary to find the vector in the T-transformed eigenspace that is closest to the T-transformed ML estimate in the original space. A basic linear algebra theorem (see Appendix D for the basic theorem) is adapted to create the following theorem:

Theorem: Let $E = [\mathbf{e}_1 \dots \mathbf{e}_K]$, where $\{\mathbf{e}_1, \dots, \mathbf{e}_K\}$ is the set of linearly independent eigenvectors that span the space $\mathcal{K}$. Then there exists a weight vector $\mathbf{w}$, such that $\hat{\mu} = E\mathbf{w}$, that satisfies $\forall \mu_{\mathcal{K}} \in \mathcal{K}\ ,\ \|T\tilde{\mu} - T\hat{\mu}\| \leq \|T\tilde{\mu} - T\mu_{\mathcal{K}}\|$ if and only if TE has

linearly independent columns. If TE has linearly independent columns, then $\mathbf{w}$ is given by $(TE)'(TE)\mathbf{w} = (TE)'T\tilde{\mu}$ or $\mathbf{w} = ((TE)'(TE))^{-1}(TE)'T\tilde{\mu}$.

Proof: The vector $T\hat{\mu} = TE\mathbf{w}$ closest to the projected ML estimate $T\tilde{\mu}$ will be the orthogonal projection of $T\tilde{\mu}$ onto the space $T(\mathcal{K})$, or alternately the error vector ε between the two vectors $T\tilde{\mu}$ and $TE\mathbf{w}$ will be perpendicular to the vectors $\{T\mathbf{e}_1 \dots T\mathbf{e}_k\}$ spanning the space $T(\mathcal{K})$. Thus, the error vector given by

$$\varepsilon = T\tilde{\mu} - TE\mathbf{w} \tag{6.7}$$

is perpendicular to $\{T\mathbf{e}_1 \dots T\mathbf{e}_k\}$ so that

$$(TE)'(T\tilde{\mu} - TE\mathbf{w}) = \mathbf{0}. \tag{6.8}$$

Thus

$$(TE)'(TE)\mathbf{w} = (TE)'T\tilde{\mu}. \tag{6.9}$$

Solving for $\mathbf{w}$,

$$\mathbf{w} = [(TE)'(TE)]^{-1}(TE)'T\tilde{\mu}. \tag{6.10}$$

From the above, it is seen that $\mathbf{w}$ exists if matrix $(TE)'(TE)$ is invertible. We will now prove that $(TE)'(TE)$ is invertible if and only if TE has linearly independent columns. To do this, it is first shown that the columns of TE are linearly independent (i.e. the nullspace of TE contains only the zero vector), then that TE and $(TE)'(TE)$ have the same nullspace, and therefore (as a matrix that contains only the zero vector in its nullspace has linearly independent columns) the square matrix $(TE)'(TE)$ has linearly independent columns, which, for a square matrix, implies invertibility.

Proving that the columns of TE are linearly independent: T is linear, and therefore $\{T\mathbf{e}_1, \dots, T\mathbf{e}_k\}$ spans $T(\mathcal{K})$. Let us find the possible solutions for vectors in the nullspace of TE as follows

$$TE\mathbf{x} = \mathbf{0} \tag{6.11}$$

The above is zero if the vector $E\mathbf{x}$ is zero, or if the vector $E\mathbf{x}$ is in the nullspace of T. As the columns of E are linearly independent, $E\mathbf{x}$ is the zero vector if and only if $\mathbf{x} = \mathbf{0}$.

Therefore we only need to determine the nullspace of T. The definition of T specifies that it does not map any of the vectors in the eigenspace (save the zero vector) to the zero vector (note that this is only the case if T has linearly independent columns). As $E\mathbf{x}$ is in the eigenspace, only $\mathbf{x} = \mathbf{0}$ satisfies Equation 6.11. Thus the nullspace of TE consists only of the zero vector, hence the columns of TE are linearly independent and span the transformed eigenspace $T(\mathcal{K})$.

Proving that the nullspace of TE and $(TE)'(TE)$ are the same: Suppose that $\mathbf{x}$ is a vector in the nullspace of TE

$$TE\mathbf{x} = \mathbf{0}. \tag{6.12}$$

Premultiplying by $(TE)'$,

$$(TE)'(TE)\mathbf{x} = \mathbf{0}, \tag{6.13}$$

therefore $\mathbf{x}$ is also present in the nullspace of $(TE)'(TE)$. Now suppose that $\mathbf{x}$ is a vector in the nullspace of $(TE)'(TE)$

$$(TE)'(TE)\mathbf{x} = \mathbf{0}. \tag{6.14}$$

Premultiplying by $\mathbf{x}'$,

$$\mathbf{x}'(TE)'(TE)\mathbf{x} = \mathbf{x}'\mathbf{0} \tag{6.15}$$

$$\text{or} \quad (TE\mathbf{x})'(TE\mathbf{x}) = 0 \tag{6.16}$$

$$\text{or} \quad \|TE\mathbf{x}\|^2 = 0, \tag{6.17}$$

therefore vector $TE\mathbf{x}$ has zero length and is the zero vector, $TE\mathbf{x} = \mathbf{0}$. Thus vector $\mathbf{x}$ is in the nullspace of TE.

Each vector in the nullspace of TE exists in the nullspace of $(TE)'(TE)$, and each vector in the nullspace of $(TE)'(TE)$ exists in the nullspace of TE. This implies that the nullspaces of TE and $(TE)'(TE)$ are identical.

It was shown that TE has linearly independent columns, and therefore its nullspace contains only the zero vector. The nullspace of $(TE)'(TE)$ therefore also only contains the zero vector, and thus the columns of $(TE)'(TE)$ are linearly independent. For a square matrix linearly independent columns implies invertibility, thus $(TE)'(TE)$ is invertible and vector $\mathbf{w}$ exists.

To summarise: TE has linearly independent columns, therefore $(TE)'(TE)$ is invertible, therefore $\mathbf{w}$ exists and is given by $\mathbf{w} = [(TE')(TE)]^{-1}(TE)'T\tilde{\mu}$.

Not only does the theorem prove the existence of $\mathbf{w}$, but it also supplies the equation to solve for the new estimate of the weight vector $\mathbf{w}$ and subsequently the new estimate of the model parameters $\hat{\mu} = E\mathbf{w}$.

This proof by Westwood has one irregularity: if T does not have linearly independent columns, then TE might not have linearly independent columns. If the columns of TE are not linearly independent, the inverse of $(TE)'(TE)$ will not exist. Westwood never proves linear independence of the columns (or invertibility) of T, but simply states that T must be chosen so that it does not transform any vector (save the null vector) in the eigenspace to the null vector. From the choices of the D and Ω matrices in Section 6.3, it is seen that T is block diagonal, and that each block is applied to a mean vector during adaptation. If there were no observed data for a mean vector, the weight function will make the block a zero matrix. For each mean vector with no observed data, matrix T will have n columns and n rows that are zero vectors (n is the dimension of the feature vectors or mean vectors). If more than one of the columns of T are the zero vector, T will not have linearly independent columns (i.e. T, a square matrix, will not be invertible). To summarise: if there are mean vectors for which there are no observed data, and the weighting function makes their associated blocks in T zero, then T is not invertible, and therefore TE might not have linearly independent columns and therefore $(TE)'(TE)$ might not invertible. If T is not invertible, the linear independence of the columns of TE depend on the specific T and E matrices.

It should be noted that rank deficient T matrix does not necessarily mean that the matrix TE does not have linearly independent columns. However, the greater the degree of rank deficiency, the less likely it is for TE to have linearly independent columns. Also, the higher the number of eigenvectors in E is, the more likely it will be for some of the eigenvectors to be projected to the zero vector by the non-invertible matrix T. Normally relatively few eigenvoices are used, so the chance of encountering a TE matrix with dependent columns should be relatively small.

When $(TE)'(TE)$ is not invertible — as the case might be for non-invertible T matrices — the pseudo-inverse can be used. When the pseudo-inverse is used, a least squares

solution for the weight vector $\mathbf{w}$ is obtained. The effect of using a pseudo-inverse (when $(TE)'(TE)$ is not invertible) on adaptation performance is not known, and it is our opinion that it should be used with caution.

6.5 Changing $(TE)'(TE)\mathbf{w} = (TE)'T\tilde{\mu}$ to the More Familiar $Q\mathbf{w} = \mathbf{v}$ for Eigenweight Estimation

In this section the equation for the new weight vector $\mathbf{w}$ for weighted projection will be expressed in a form that is analogue to the equation for the eigenweights for MLED.

To do so, first expand $T'T$ and equate it to a new matrix B:

$$T'T = D'\Omega'\Omega D \tag{6.18}$$

$$= \begin{bmatrix} \ddots & 0 & 0 \\ 0 & \left(D_m^{(s)}\right)' & 0 \\ 0 & 0 & \ddots \end{bmatrix} \begin{bmatrix} \ddots & 0 & 0 \\ 0 & \omega\left(A_0(s,m)\right)^{1/2} I_d & 0 \\ 0 & 0 & \ddots \end{bmatrix}$$

$$\begin{bmatrix} \ddots & 0 & 0 \\ 0 & \omega\left(A_0(s,m)\right)^{1/2} I_d & 0 \\ 0 & 0 & \ddots \end{bmatrix} \begin{bmatrix} \ddots & 0 & 0 \\ 0 & \left(D_m^{(s)}\right) & 0 \\ 0 & 0 & \ddots \end{bmatrix}. \tag{6.19}$$

Recalling that $D_m^{(s)}$ is a symmetric matrix (possibly one of I, $C_m^{(s)-1/2}$ or $\overline{C}^{-1/2}$),

$$T'T = \begin{bmatrix} \ddots & 0 & 0 \\ 0 & \omega\left(A_0(s,m)\right)\left(D_m^{(s)}\right)^2 & 0 \\ 0 & 0 & \ddots \end{bmatrix} = B. \tag{6.20}$$

The right-hand side of Equation 6.9 may now be written as

$$\mathbf{v} = (TE)'T\tilde{\mu} = E'B\tilde{\mu} \tag{6.21}$$

or, if v_i is an element of $\mathbf{v}$ and $\mathbf{e}(i)$ is the i-th column of E, then

$$v_i = \mathbf{e}(i)'B\tilde{\mu} \tag{6.22}$$

$$= \sum_{s=1}^{S}\sum_{m=1}^{M_s} \mathbf{e}_m^{(s)}(i)'\omega\left(A_0(s,m)\right)\left(D_m^{(s)}\right)^2\tilde{\mu} \tag{6.23}$$

$$= \sum_{s=1}^{S}\sum_{m=1}^{M_s} \omega\left(A_0(s,m)\right)\mathbf{e}_m^{(s)}(i)'\left(D_m^{(s)}\right)^2\tilde{\mu}_m^{(s)} \tag{6.24}$$

and, since $\tilde{\mu} = \frac{A_1(s,m)}{A_0(s,m)}$

$$v_i = \sum_{s=1}^{S} \sum_{m=1}^{M_s} \frac{\omega\left(A_0(s,m)\right)}{A_0(s,m)} \mathbf{e}_m^{(s)}(i)' (D_m^{(s)})^2 A_1(s,m). \tag{6.25}$$

Similarly, the left-hand side of Equation 6.9 may now be written as

$$E'T'TE\mathbf{w} = E'BE\mathbf{w} = Q\mathbf{w}, \tag{6.26}$$

where every element q_{ij} of Q is given by

$$q_{ij} = \mathbf{e}(j)'B\mathbf{e}(i) \tag{6.27}$$

$$= \sum_{s=1}^{S} \sum_{m=1}^{M_s} \omega\left(A_0(s,m)\right) \mathbf{e}_m^{(s)}(j)' \left(D_m^{(s)}\right)^2 \mathbf{e}_m^{(s)}(i). \tag{6.28}$$

Combining Equations 6.22 and 6.26, Equation 6.9 may be expressed as

$$Q\mathbf{w} = \mathbf{v}. \tag{6.29}$$

Looking at expressions for q_{ij} and v_i, it is seen that the weight estimation equations for weighted projection are very similar to those for MLED. In fact, in Section 6.6 it will be shown that certain choices for D and Ω will result in identical adaptation equations for weighted projection and MLED.

6.6 Comparing the Adaptation Equations for Weighted Projection and MLED

Here it will be shown that weighted projection is identical to MLED for certain choices of the decorrelation matrix D and the weighting matrix Ω.

Westwood [51] proved that when the weighting function is the identity function (i.e. $\omega(x) = x$), and component-by-component decorrelation is used, weighted projection reduces to MLED using Baum-Welch segmentation (i.e. the state-occupation probabilities $\gamma_m^{(s)}(t)$ and the accumulators were obtained via the Baum-Welch algorithm). Substituting $A_0(s,m)$ for $\omega\left(A_0(s,m)\right)$ into Equations 6.25 and 6.28 it can be shown that

$$v_i = \sum_{s=1}^{S} \sum_{m=1}^{M_s} \frac{A_0(s,m)}{A_0(s,m)} \mathbf{e}_m^{(s)}(i)' \left(C_m^{(s)-\frac{1}{2}}\right)^2 A_1(s,m) \tag{6.30}$$

$$= \sum_{s=1}^{S} \sum_{m=1}^{M_s} \sum_{t=1}^{T} \gamma_m^{(s)}(t) \mathbf{e}_m^{(s)}(i)' C_m^{(s)-1} \mathbf{o}_t \tag{6.31}$$

and

$$q_{ij} = \sum_{s=1}^{S}\sum_{m=1}^{M_s} A_0(s,m)\mathbf{e}_m^{(s)}(j)'\left(C_m^{(s)-\frac{1}{2}}\right)^2\mathbf{e}_m^{(s)}(i) \tag{6.32}$$

$$= \sum_{s=1}^{S}\sum_{m=1}^{M_s}\sum_{t=1}^{T}\gamma_m^{(s)}(t)\mathbf{e}_m^{(s)}(j)'C_m^{(s)-1}\mathbf{e}_m^{(s)}(i) \tag{6.33}$$

which are identical to Equations 5.24 and 5.25 for MLED adaptation. Note that the above derivation is also valid for the case where Nguyen's semi-Viterbi and full-Viterbi segmentation is employed [40]. From the above, Equation 6.26 and the discussion on the invertibility of $(TE)'(TE) = Q$, it is seen that standard MLED suffers from same problems as weighted projection. A check on the condition number of Q — or checking whether the columns of TE are linearly independent — should reveal potential numerical instability when inverting Q. Otherwise the pseudo-inverse of Q could be used in order to obtain a least squares solution for $\mathbf{w}$.

The identity weighting function (combined with component-by-component decorrelation) is not the only one that reduces weighted projection to MLED. Westwood considered weighting functions belonging to the family

$$\rho_q(x) = \begin{cases} 0 & x \leq 0 \\ x^q & x > 0 \end{cases}. \tag{6.34}$$

Two special cases of this weighting function were found to be of particular interest. The first was termed *occupancy* weighting, where $q = 1$ (recall that occupancy weighting combined with component-by-component decorrelation was shown to be equivalent to MLED), and the other was termed *indicator* weighting, where $q = 0$.

Consider the case where Viterbi segmentation is used to obtain state-occupation probabilities for an HMM with single Gaussian state pdfs. Here

$$\gamma^{(s)}(t) = \begin{cases} 1 & \text{if the best path goes through state } s \text{ at time } t \\ 0 & \text{otherwise} \end{cases}. \tag{6.35}$$

If occupancy weighting and component-by-component decorrelation is now used, then weighted projection reduces to MLED. Also, should indicator weighting be used, it is easy to show that weighted projection once more reduces to MLED.

From the above it is clear that though the weighted projection and MLED methods approach the adaptation problem from different perspectives, the resulting adaptation

equations are very similar, or, in some cases, identical. It is thus no surprise that Westwood found the recognition performance of weighted projection adaptation to be of the same order as the recognition performance of MLED adaptation for a variety of weighting functions and decorrelation matrices.

6.7 Recognition Performance

Westwood [51] found that for the family of weighting functions defined by Equation 6.34, the recognition performance was not highly sensitive to the choice of q. Performance was found to be of the same order for $0.2 \leq q \leq 1.8$. Also, the choice of decorrelation matrix had little influence on recognition. As expected, component-by-component decorrelation resulted in slightly better performance than other decorrelations for a monophone recognition problem, whereas for a triphone recognition problem (where the covariance matrices were poorly estimated) global decorrelation resulted in slightly better performance than other decorrelation matrices.

Recognition performance in general was of the same order as that of MLED.

6.8 Advantages of Using Weighted Projection

MLED is one of the most elegant and successful speaker adaptation techniques, and many new extensions thereof may still be created (such as using non-linear PCA to extract an eigenspace) to suit different speaker adaptation problems. Weighted Projection is also an eigen-decomposition adaptation technique, and has the same flexibility as MLED in the choice of eigenspace extraction. The adaptation equations are virtually identical for both MLED and weighted projection. Also, the recognition performance — both in terms of computational complexity, adaptation data required and the percentage of phonemes correctly recognised — of both adaptation methods is of the same order. Since MLED has a patent pending, weighted projection may prove to be a viable alternative.

Chapter 7

A Few Novel Approaches

7.1 Introduction to the Novel Approaches

We have created two novel extensions for eigenvoice methods, in particular MLED. The first method is a very simple extension to eigenvoice modelling, and the second incorporates the use of the class-based Karhunen-Loéve transform into a MLED reestimation framework. The rationale behind the development of the methods is now stated:

7.1.1 Motivation for Creating a Mean-Preserving Covariance-Based MLED Extension

The original MLED method applied PCA on the correlation matrix of the prior set of SD supervectors in order to extract the eigenvectors. When this is done, it is very likely that the first eigenvector extracted essentially models the mean of the set of SD supervectors. Alternatively, when PCA is applied to the covariance matrix, eigenvectors are extracted that do not model the mean of the set of SD models.

Thus, a correlation approach "wastes" the first eigenvector to modelling the supervector mean - a value known prior to adaptation. An eigenweight needs to be robustly estimated for this eigenvector, which places more strain on a system that must adapt using the barest amount of data.

The covariance approach, on the other hand, entirely discards the information contained in the mean of the prior SD supervectors. This mean can only be modelled if a linear combination of the chosen eigenvectors allows it. At best, the burden of modelling

the mean of the SD supervectors is spread over a few eigenvectors, and at worst, the mean of the SD supervectors cannot be modelled accurately enough by any linear combination of eigenvectors.

Shortcomings of both the correlation and covariance approaches led us to design an approach that allows the eigenvoices to exclusively model the inter-speaker variance, while the mean of the speaker-dependent models is modelled automatically. Thus the mean is not discarded, even though no eigenvoice is used to explicitly model it. In practice, both Nguyen [40] and Westwood [51] have shown the correlation method to have good results, even when using only one eigenvoice. When we attempted to use the covariance approach, however, the performance when using only a single eigenvoice was inferior to that of the correlation approach. It is our opinion that this is because the first eigenvector of the covariance approach is insufficient to model the mean of the SD supervectors. Our results are contradictory to those of Westwood, who found the two methods to have very similar performance, even for only one eigenvector. Our mean preserving extension is shown to outperform both previous approaches when few eigenvoices are used, while being on par when many eigenvoices are used.

7.1.2 Motivation for the Design of a Class-Based "Eigenvoice" Method

Accent is one of the largest contributors to inter-speaker variance (see Section 1.2.2), and in most English-speaking countries speakers belong to one of several accent classes. When designing a system to cope with speakers from various dialects, one of the following might be employed:

- For small recognition problems with a very small lexicon, a system can be designed that only uses one set of phonemes to model a word. There is thus only one global model that is employed to recognise every possible accented pronunciation of the word. Needless to say, such a system would probably have very poor performance.

- A system can be designed where all the phonemes (or triphones, or syllables, etc.) present in all the dialects are explicitly modelled. To model a word in the system, several different HMM sets are then required - one set to cope with the sequence of

phonemes used to form the word for each of the dialect classes. While this will result in an accurate model, it requires the training of more phonemes than the previous approach. Also, as there are more word models using this method, it might affect recognition performance.

We focus our attention on the first, smaller recognition task. We will now refer to the set of phonemes modelled by a single HMM as a "speech unit". Performance of this type of system could be improved by the use of eigenvoices. Using the same HMM to model different phonemes produced by speakers with different accents will make the variance of the SI model for the speech unit very large. However, a well-trained SD model should have approximately the same intra-speaker variance for a speech unit as for a single phoneme. Should eigenvoices be extracted from a set of such SD models, the large inter-speaker variance between speech units could be effectively modelled. A weighted sum of eigenvoices should then make it possible to accurately adapt HMM models of speech units for different accent classes.

Eigenvectors extracted via PCA represent the directions of greatest inter-speaker variance. The inherent assumption is thus that the directions of greatest variance are the most important and carry the most valuable information of the original space. What if another factor, such as the accent class of a speaker, contains information more relevant to HMM model adaptation than the inter-speaker variance? In this instance, a subspace modelling the greatest inter-class variance may perform better than a subspace modelling the greatest inter-speaker variance. Such a subspace may be extracted using the class-based Karhunen-Loéve transform (CBKLT).

The hypothesis regarding inter-accent-class variance was thus tested by extracting a set of subspace-spanning vectors using the CBKLT, and by implementing a CBKLT-based MLED method for accent and speaker adaptation. For the speech corpus on which the evaluation was performed, the performance of the mean-preserving CBKLT-based method was of the same order as standard MLED performance. In an experiment where male and female speaker classes were used, the performance using only a single gender-based eigenvoice indicated that adaptation with regards to speaker sex is sufficient to obtain moderate increases in recognition rate.

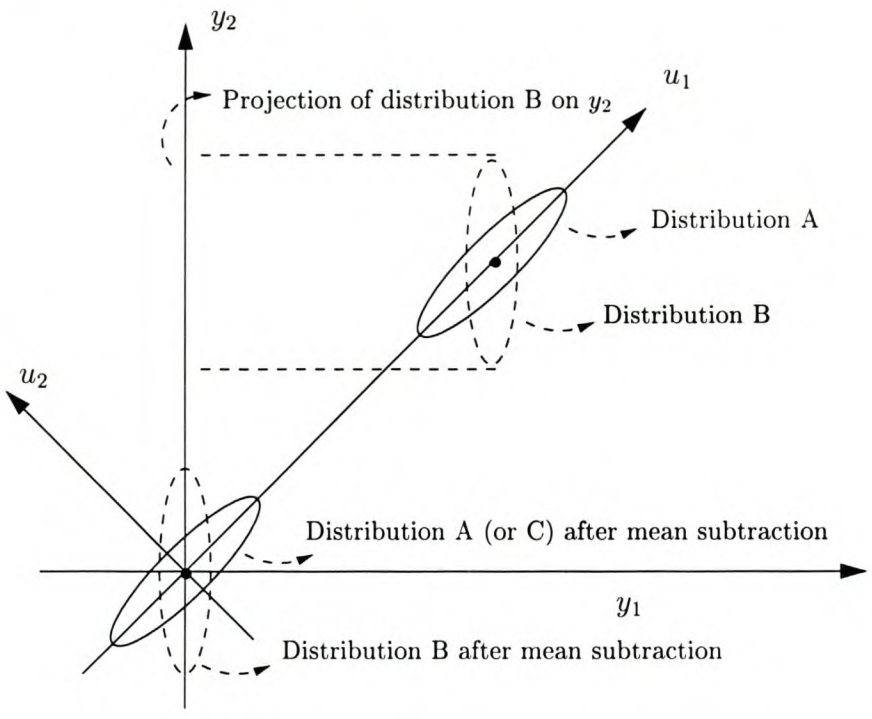

Figure 7.1: KLT on correlation matrix vs. KLT on covariance matrix.

Having discussed the motivation behind the creation of the MLED extensions, as well as giving an introduction into the methods, we now move on to an in-depth treatment.

7.2 Using a Mean-Preserving Covariance-Based Approach for MLED

A small but powerful change to eigenvoice decomposition is to model the mean separately, as this focuses the modelling of the eigenvoices solely on extracting axes representing the greatest directions in variance. Figure 7.1 is used to demonstrate the effect of various KLT implementations have when applied to different data sets. Each elliptic "cloud" on the figure represents a roughly Gaussian spread of data points with greatest variance in the direction of the long axes and lowest variance in the direction of the small axes. The following KLT implementations were analysed:

- **Applying the KLT on the correlation matrix of a data set:** First, consider eigenvector extraction for the data set A. The first eigenvector (u_1) will go through

the mean of A, and the second eigenvector (u_2) will be perpendicular to it. The error when using only one eigenvector to model points in the set is relatively small. Next consider data set B. The same eigenvectors as for data set B are extracted, but the error in representing the points in the data set using only eigenvector u_1 is much larger than before. The reason for this is that the first eigenvector extracted in this way essentially models the mean of the data set it is applied on. This KLT implementation tends to preserve the mean of the original data set, but the first eigenvector may not be in the direction of most variance.

- **Applying the KLT on the covariance matrix of a data set:** This is the same as first subtracting the mean of the distribution from the points in the data set, and then applying the KLT. First, consider distribution A. After mean subtraction, A lies around the origin, and eigenvectors u_1 and u_2 are extracted. Approximating the points in A using only the first eigenvector u_1 is good, as the eigenvoice goes through the original distribution before the mean was subtracted. Now, consider distribution B. Distribution B is shifted to lie around the origin, and the eigenvectors lie in the same direction as the original axes, i.e. y_2 is the first eigenvector and y_1 is the second eigenvector. Here, modelling of the original B distribution using only y_2 will be poor, as the error between the projection of B onto y_2 and B will be much higher than was the case for using the KLT on the correlation matrix. This KLT implementation does not preserve the mean of the original data set, but the first eigenvector is in the direction of most variance.

- **Applying the KLT and preserving the mean:** Here, a new data set is formed be first subtracting the mean of the original data set from itself. The KLT is then applied on the new mean-less dataset, and the eigenvectors are determined. To return to the original set of axes, we back-project and then add the original mean vector. Data set B was the most difficult to represent using eigenvectors of the previous two KLT implementations. Should this method be used, the mean of B is subtracted until B lies around the origin. The eigenvectors extracted will be y_2 and y_1. If only the first eigenvector y_2 is used, then the error between the mean-less dataset B its projection will be very small. The error after back-projecting and restoring the mean of B compared to the original data set B will be the same small

value. This KLT implementation preserves the mean, as well as having the first eigenvector aligned in the direction of most variance.

From the above, it is clear that only the final KLT implementation preserves the mean vector and has its most important eigenvectors aligned to the directions of most variance. As such, the mean square error in the approximation of the original data set using this KLT implementation will probably be the least when only a few eigenvectors are to be used.

In summary: When applying the KLT (PCA) to the correlation matrix of the supervectors, the first eigenvector essentially models the mean of the supervectors. If the eigenvectors are to be used exclusively for the modelling of inter-speaker variability while retaining the ability to model the mean, the third KLT implementation must be used.

Simply estimating a new model using MLED and then restoring the original mean vector will not suffice, as implementation of mean preservation must be included in the MLED framework in a maximum likelihood fashion. To do this, first subtract the mean of the supervectors. The columns Y_i of the data matrix are now given by

$$Y_i = \mu_i - \frac{1}{J}\sum_{j}^{J}\mu_j \quad \text{where} \quad i = 1 \ldots J, \tag{7.1}$$

where J is the number of SD supervectors and μ_j is a supervector.

This new data matrix is now used in Equation 5.34 and the new set of eigenvectors (the columns of the U matrix) are computed. A new estimate for a mean vector in a speaker model is now computed by adding the weighted sum of eigenvoices to the mean of the supervectors.

$$\hat{\mu} = \rho + \sum_{j=1}^{K} w_j \mathbf{e}(j), \tag{7.2}$$

where ρ is the mean of the supervectors that must be preserved. Alternatively, for each of the individual model mean vectors

$$\hat{\mu}_m^{(s)} = \rho_m^{(s)} + \sum_{j=1}^{K} w_j \mathbf{e}_m^{(s)}(j), \tag{7.3}$$

where $\rho_m^{(s)}$ is the mean of the mean vectors of mixture component m in state s for all of the SD speaker models.

Maximising the auxiliary function with respect to the weights follows the same path as before with very slight differences. The derivation is the same from Equations 5.10 through 5.12. As before, it is necessary to find the derivative $\hat{\mu}_m^{(s)}$ with respect to an eigenweight before the derivative of $h_m^{(s)}(\mathbf{o}_t)$ with respect to an eigenweight can be determined in order to solve Equation 5.12.

$$\frac{\partial}{\partial w_i}\hat{\mu}_m^{(s)} = \frac{\partial}{\partial w_i}\left\{\rho_m^{(s)} + \sum_{j=1}^{K} w_j \mathbf{e}_m^{(s)}(j)\right\} = \mathbf{e}_m^{(s)}(i). \tag{7.4}$$

Now, as before

$$\frac{\partial}{\partial w_i}h_m^{(s)}(\mathbf{o}_t) = -2\mathbf{o}_t' C_m^{(s)-1}\left(\frac{\partial}{\partial w_i}\hat{\mu}_m^{(s)}\right) + \frac{\partial}{\partial w_i}\left(\hat{\mu}_m^{(s)'} C_m^{(s)-1}\hat{\mu}_m^{(s)}\right) \tag{7.5}$$

substituting Equation 7.2 for the adapted mean vectors gives

$$= -2\mathbf{o}_t' C_m^{(s)-1}\mathbf{e}_m^{(s)}(e) + \left(\frac{\partial}{\partial w_i}\hat{\mu}_m^{(s)'}\right) C_m^{(s)-1}\hat{\mu}_m^{(s)} + \hat{\mu}_m^{(s)'} C_m^{(s)-1}\left(\frac{\partial}{\partial w_i}\hat{\mu}_m^{(s)}\right). \tag{7.6}$$

scalars are equal to their transpose, so

$$= -2\mathbf{o}_t' C_m^{(s)-1}\mathbf{e}_m^{(s)}(i) + 2\left(\frac{\partial}{\partial w_i}\hat{\mu}_m^{(s)'}\right) C_m^{(s)-1}\hat{\mu}_m^{(s)} \tag{7.7}$$

$$= -2\mathbf{o}_t' C_m^{(s)-1}\mathbf{e}_m^{(s)}(i) + 2\mathbf{e}_m^{(s)'}(i) C_m^{(s)-1}\hat{\mu}_m^{(s)} \tag{7.8}$$

again, by substituting equation 7.2

$$= 2\left[-\mathbf{o}_t' C_m^{(s)-1}\mathbf{e}_m^{(s)}(i) + \mathbf{e}_m^{(s)'}(i) C_m^{(s)-1}\left\{\rho_m^{(s)} + \sum_{j=1}^{K} w_j \mathbf{e}_m^{(s)}(j)\right\}\right]. \tag{7.9}$$

Substituting the results of Equation 7.9 in Equation 5.12 gives

$$\frac{\partial Q}{\partial w_i} = 0 = \tag{7.10}$$

$$\sum_s \sum_m \sum_t \gamma_m^{(s)}(t)\left[-\mathbf{o}_t' C_m^{(s)-1}\mathbf{e}_m^{(s)}(i) + \mathbf{e}_m^{(s)'}(i) C_m^{(s)-1}\left\{\rho_m^{(s)} + \sum_{j=1}^{K} w_j \mathbf{e}_m^{(s)}(j)\right\}\right]$$

or, equivalently,

$$\sum_s \sum_m \sum_t \gamma_m^{(s)}(t)\mathbf{e}_m^{(s)'}(i) C_m^{(s)-1}\mathbf{o}_t =$$

$$\sum_s \sum_m \sum_t \gamma_m^{(s)}(t)\left\{\rho_m^{(s)} \sum_{j=1}^{K} w_j \mathbf{e}_m^{(s)'}(j) C_m^{(s)-1}\mathbf{e}_m^{(s)}(i)\right\}, \quad i = 1 \ldots K. \tag{7.11}$$

Equation 7.11 defines the set of linear equations that must be solved in order to obtain an optimal set of eigenweights. This set of equations can also be expressed as

$$\mathbf{v} = Q\mathbf{w}, \tag{7.12}$$

where the E-dimensional vector of eigenweights is

$$\mathbf{w} = [w_1, w_2, \ldots, w_K]^T \tag{7.13}$$

and $\mathbf{v}$ is a K-dimensional vector given by

$$\mathbf{v} = \begin{bmatrix} \sum_s \sum_m \sum_t \gamma_m^{(s)}(t) \mathbf{e}_m^{(s)'}(1) C_m^{(s)-1} (\mathbf{o}_t - \rho_m^{(s)}) \\ \sum_s \sum_m \sum_t \gamma_m^{(s)}(t) \mathbf{e}_m^{(s)'}(2) C_m^{(s)-1} (\mathbf{o}_t - \rho_m^{(s)}) \\ \vdots \\ \sum_s \sum_m \sum_t \gamma_m^{(s)}(t) \mathbf{e}_m^{(s)'}(K) C_m^{(s)-1} (\mathbf{o}_t - \rho_m^{(s)}) \end{bmatrix} \tag{7.14}$$

and every element q_{ij} of the $K \times K$ matrix Q is given by

$$q_{ij} = \sum_s \sum_m \sum_t \gamma_m^{(s)}(t) w_j \mathbf{e}_m^{(s)'}(j) C_m^{(s)-1} \mathbf{e}_m^{(s)}(i). \tag{7.15}$$

Thus, as before,

$$\mathbf{w} = Q^{-1}\mathbf{v}. \tag{7.16}$$

Note that besides the eigenvoices being different, only the computation of vector $\mathbf{v}$ has changed. Mean preservation for MLED is thus a simple extension to eigenvoice decomposition, and was easily and successfully implemented. As predicted the first eigenvoice used in the original MLED essentially models the mean of the SD supervectors. The use of mean preservation resulted in having the adaptation power afforded by normal MLED using two eigenvoices for roundabout the same computational cost as using a single eigenvoice.

7.3 Using the Class-Based Karhunen-Loève Transform for Accents

R. Kuhn *et al.* [28] state that the eigenvoice concept may possibly be extended to use more complex dimension reduction techniques such as non-linear PCA for more accurate approximations of the speaker space prior information. Section 7.2 extended the use of eigenvoice decomposition to different KLT implementations, and in this section the idea of eigenvoice decomposition is extended to use the class-based Karhunen-Loève transform

(CBKLT). The CBKLT is similar to class-based linear discriminant analysis (LDA) and is also known as the Fisher method for the two-class case.

Nguyen [40] found that, for the MLED experiments on the ISOLET speech corpus, the first eigenvoice could be used to identify male and female speakers, indicating that males and females could possibly form two classes that may be used effectively using a class-based transform like CBKLT. Male-female separation in the speaker space is useful, but other interesting uses for MLED using CBKLT are also possible.

Consider the USA, where accents are prominent enough for a human listener to make an accurate guess as to the ethnicity or region of origin of the speaker. Accent groups could represent separate speaker classes for use in a CBKLT, where the resulting transformed space will have the greatest linear separation between speakers of different accent regions — assuming, of course, that the classes are indeed linearly separable.

In a country such as South Africa there is a great deal of divergent accents, as there are so many people from diverse ethnic backgrounds that speak English as their second or third language. Often such speakers' first language does not have the same phonetic richness as English. Phonemes with which they are familiar are frequently substituted in place of a more standard English phoneme pronunciation. For a small vocabulary recognition task such as a single digit number recognition problem, a different set of phonemes could be used to represent the same digit for speakers with different accents, resulting in a larger recognition problem. Alternatively, the same HMM could be used to represent a digit for speakers of all accents. SD models representing the digit would then be quite dissimilar for speakers with different accents, thus resulting in reasonably large separation between accent classes in the speaker space. Speaker adaptation using a CBKLT implementation of eigenvoice decomposition might then be used to successfully adapt the HMM for new speakers.

When the KLT is used to create an eigenspace, the inherent assumption is that the directions of greatest variance holds the most important data relevant to speaker adaptation. For the speech corpora such as ISOLET, this assumption was valid and MLED adaptation performance was excellent. What if the modelling problem needs to address speakers with different accents? Will the most important data for speaker recognition purposes still be associated with the directions of greatest inter-speaker variance, or would

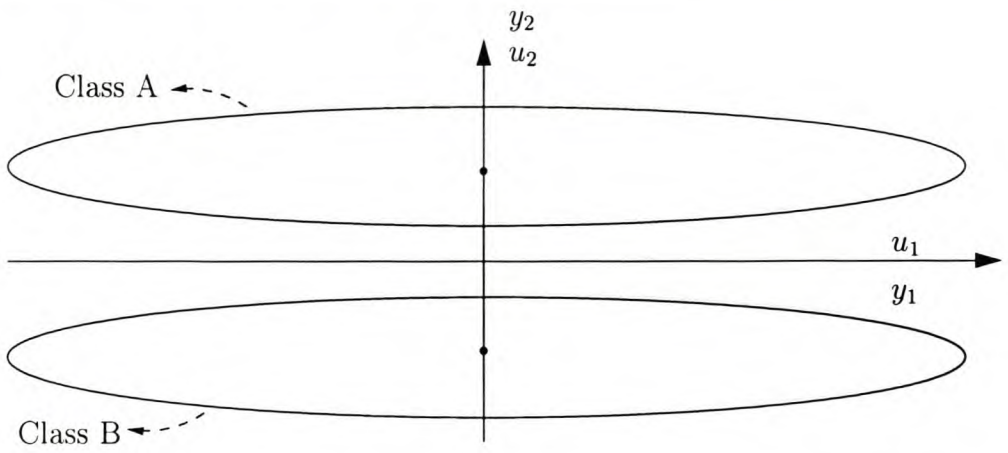

Figure 7.2: Choosing axes according to the greatest class variance vs. choosing axes according to greatest data vector variance.

the accent class of the speaker be more important? To test the hypotheses, an extension of maximum likelihood eigenvoice decomposition incorporating the well-known class-based Karhunen-Loéve transform is presented. Before the derivation of the MLED extension is continued, it is necessary to treat the class-based Karhunen-Loéve transform.

7.3.1 The Class-Based Karhunen-Loève Transform

7.3.1.1 A Brief Outline of the CBKLT

Refer to Figure 7.2. Here the two ellipses represent the data for two linearly separable classes. With normal KLT, the first or dominant axis of the transformed space will be in the direction of greatest variance of the original data. For the two classes in Figure 7.2, the principal eigenvector will be u_1 and will lie on the original y_1 axis, thereby yielding no separation between classes. CBKLT, however, chooses axes according to the greatest variance in class, resulting in maximal linear class separation along the axes in the transformed space. The first "eigenvector" (it is not entirely correct to refer to the rows of the CBKLT matrix as eigenvectors, as they are not orthonormal) obtained via the CBKLT will be u_2 and will lie on the original y_2 axis and through the means of the original classes.

The CBKLT is done as follows: First a whitening transform is applied to the data to make the average covariance of the classes the unity matrix. After this, another KL transform is applied on the class means instead of on the individual data vectors —

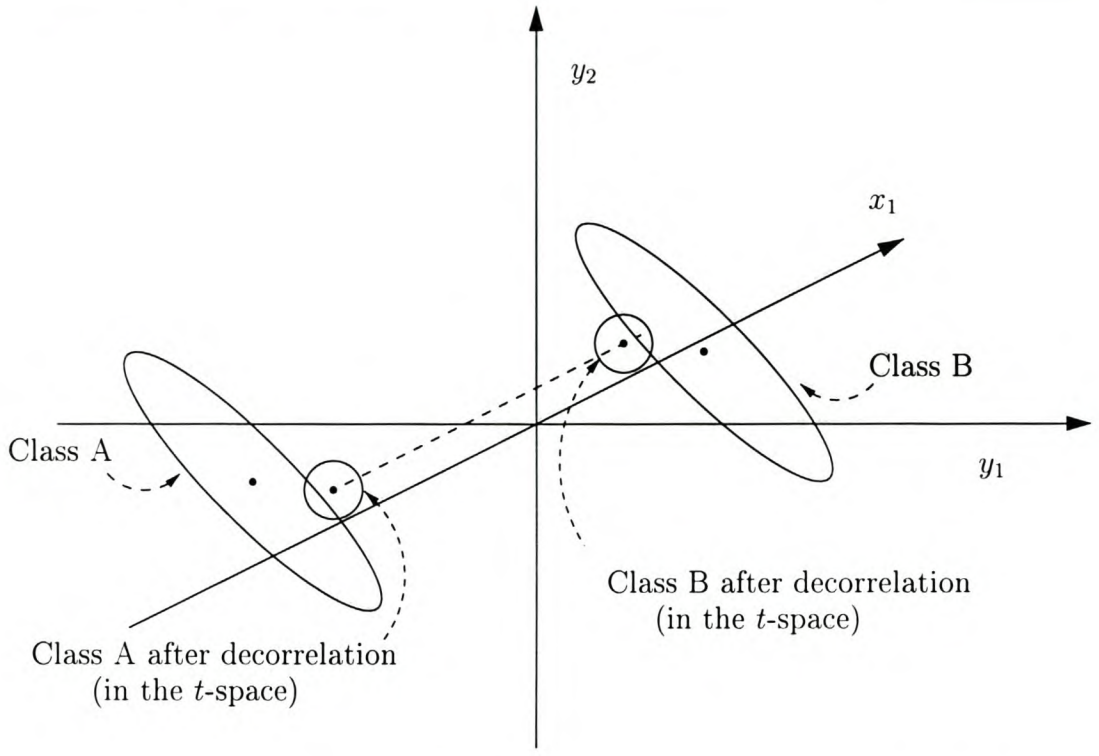

Figure 7.3: Representation of the effect of CBKLT on a two-class data set (adapted from [11]).

treating the class means as data points for a KLT results in the desired alignment of the axes in the direction of greatest variance in the class means.

7.3.1.2 Derivation of the CBKLT

As previously stated, the class-based Karhunen-Loève transform is essentially a two-step transformation. A feature vector $\mathbf{y}$ in the original space (the y-space) is transformed to a feature vector $\mathbf{t}$ in an intermediate whitened space (the t-space), which is in turn transformed to a feature vector $\mathbf{x}$ in the final space (the x-space). Equations 7.17 and 7.18 represent this two-step transformation:

$$\mathbf{t} = B'\mathbf{y}. \tag{7.17}$$

$$\mathbf{x} = V'\mathbf{t} = V'B'\mathbf{y} = W'\mathbf{y}. \tag{7.18}$$

If each $\mathbf{y}_j^{(i)}$ is used to represent a data vector number j in class i, and n_i is the number

of data vectors in class i, then the covariance matrix for a class is given by

$$\Sigma_i = \frac{1}{n_i}\sum_{j=1}^{n_i}(\mathbf{y}_j^{(i)} - \mu_i)(\mathbf{y}_j^{(i)} - \mu_i)', \tag{7.19}$$

where μ_i is the mean of class i given by

$$\mu_i = \frac{1}{n_i}\sum_{j=1}^{n_i}\mathbf{y}_j^{(i)}. \tag{7.20}$$

The average covariance matrix for all the classes is now given by

$$S_w = \sum_{i=1}^{c} P_i\Sigma_i \tag{7.21}$$

where c is the number of classes and P_i is the probability of class i occurring. This average covariance matrix is also known as the intra-class dispersion matrix and it gives an indication of how data of the classes are spread in general.

$$S_b = \sum_{i=1}^{c} P_i(\mu_i - \mu_T)(\mu_i - \mu_T)', \tag{7.22}$$

where μ_T is the mean of all the feature vectors in all the classes. Equation 7.22 is known as the inter-class dispersion matrix, and it gives an indication of how the centroids of the classes are spread.

If the probability is simply determined by the relative number of feature vectors assigned to each class, then the following may be substituted for P_i:

$$P_i = \frac{n_i}{n_T}, \tag{7.23}$$

where n_T is the total combined number of feature vectors for the classes.

For CBKLT, a transform is required that makes the intra-class dispersion matrix the unity matrix in the new space. If the intra-class dispersion matrix is the unity matrix, then it implies that the features are decorrelated. These features will also remain decorrelated after any further orthonormal transformations. If the required transform is B', and the orthonormal eigenvector matrix and diagonal eigenvalue matrix for S_w are U and Λ respectively, then the new intra-class dispersion matrix (after transformation) is given by $\hat{S_w}$:

$$\hat{S_w} = I = B'S_wB \tag{7.24}$$

$$= \Lambda^{-1/2}\Lambda^{1/2} \tag{7.25}$$

$$= \Lambda^{-1/2'}\Lambda^{1/2} \tag{7.26}$$

$$= \Lambda^{-1/2'}U'U\Lambda U^{-1}U\Lambda^{-1/2}. \tag{7.27}$$

Thus

$$B'S_wB = (U\Lambda^{-1/2})'(U\Lambda U^{-1})U\Lambda^{-1/2}. \tag{7.28}$$

The SVD of S_w is

$$S_w = U\Lambda U' = U\Lambda U^{-1}. \tag{7.29}$$

Substituting Equation 7.29 in the right side of Equation 7.28, it follows that the desired transform B' is given by

$$B' = (U\Lambda^{-1/2})' = (\Lambda^{-1/2})'U' = \Lambda^{-1/2}U'. \tag{7.30}$$

Transforming the features using B' thus makes the variance of the data in the intermediate space (represented by the $\mathbf{t}$ vectors), when viewed across all the classes, the same in all dimensions. The inter-class dispersion matrix is also transformed to become $\hat{S}_b$:

$$\hat{S}_b = B'S_bB. \tag{7.31}$$

In order to obtain optimal compression of the data, a transformation that is based on the eigenvectors of $\hat{S}_b$ is now applied. The transformation is a simple rotation of axes, so that the features will remain decorrelated after the new transformation. If the eigenvector matrix of $\hat{S}_b$ is given by V , then the total transformation applied to the original data is W', given by

$$\begin{array}{ccccc} W' & = & V' & & B' \\ m \times m & & m \times m & & m \times m \end{array} \tag{7.32}$$

where m is the dimension of the data to be transformed.

If $\mathbf{x}$ is a vector in the new transformed space, then $\mathbf{x}$ and $\mathbf{y}$ are related by

$$\mathbf{x} = W'\mathbf{y} \quad \text{or} \quad \mathbf{y} = (W')^{-1}\mathbf{x}. \tag{7.33}$$

Note that the transform W' is not orthonormal, as is the case with the normal Karhunen-Loève transform.

It is not necessary to retain all the dimensions after the first transform B', as all but the first r eigenvectors will be associated with non-null eigenvalues. The other $m - r$ eigenvalues will be zero and their eigenvectors will describe the null-space of S_w. Thus only

the first r eigenvectors and eigenvalues need to be used to form the reduced transformation matrix B'_+. As the t-space (created by applying transform B'_+ to the y-space) is only r-dimensional, the transformed inter-class dispersion matrix $\hat{S}_b$ only has dimension $r \times r$. Consequently, the transformation formed be the eigenvectors of $\hat{S}_b$ also only has dimension $r \times r$. The reduced transform W'_+ is given in Equation 7.34:

$$\underset{r \times m}{W'_+} = \underset{r \times r}{V'_+} \; \underset{r \times m}{B'_+} = \underset{r \times r}{V'_+} \; \underset{r \times r}{\Lambda_+^{-1/2}} \; \underset{r \times m}{U'_+} \tag{7.34}$$

Any dimension reduction and loss of data that have occurred at this stage are a side effect of CBKLT, and it occurs because all the data in a class are represented by the mean of the class in the calculation of the V'_+ transform. The amount of loss is greater when fewer classes are used for the same data set. Further dimension reduction is obtained at a loss by only retaining those eigenvectors corresponding to the largest eigenvalues of $\hat{S}_b$ in V_{++}, so that

$$\underset{e \times m}{W'_{++}} = \underset{e \times r}{V'_{++}} \; \underset{r \times m}{B'_+} \tag{7.35}$$

where e is the number of eigenvectors retained in transformation matrix V'_{++}.

Transforming the original features into the new e-dimensional x-space is then given by

$$\mathbf{x} = W'_{++}\mathbf{y} = V'_{++}B'_+\mathbf{y} = V'_{++}(\Lambda^{-1/2})'U'_+\mathbf{y}. \tag{7.36}$$

Therefore, the matrix used in order to describe the reduced speaker space (similar to the original concept of the "eigenspace") in MLED is then the left-inverse of (W'_{++}), so that

$$\mathbf{y} = (W'_{++})^{-1}\mathbf{x} = U_+\Lambda_+^{1/2}V_{++}\mathbf{x}. \tag{7.37}$$

Recalling that the columns of U_+ are orthonormal, the columns of V_{++} are orthonormal and that Λ is diagonal, the validity of the left-inverse matrix can be checked by simple multiplication:

$$\begin{aligned}(W'_{++})^{-1}W'_{++} &= (U_+\Lambda_+^{1/2}V_{++})(V'_{++}(\Lambda^{-1/2})'U'_+) \\ &= U_+\Lambda_+^{1/2}I\Lambda^{-1/2}U'_+ \\ &= U_+IU'_+ \\ &= I,\end{aligned} \tag{7.38}$$

where the final identity matrix I is an $e \times e$ matrix. As with the extraction of the normal eigenvoices using the KLT, the memory requirements for computing the "eigenvoices" using the CBKLT are quite large. A similar memory reduction method as the one for KLT (Section 5.4.3) was thus used for the CBKLT computation of basis vectors. The memory reduction method for the CBKLT may be found in Appendix C.

7.3.2 Extending the MLED Idea to Use CBKLT Speaker Space Reduction

During the MLED derivations, the eigenvoices used were orthonormal with respect to each other, and so the reasonable assumption was made that the eigenweights are independent of one another.

When the CBKLT is utilised, however, the transformation matrix W' is not orthonormal. The concept of eigenvoices is also not really applicable any more either, as the columns of $(W')^{-1}$ are the product of a rotation-and-scaling transform and another rotation transform, so the individual columns are not orthonormal. However, by making a simple assumption regarding independence of the weights, a derivation may be made for MLED-like adaptation of a model using a reduced-dimension speaker space.

During CBKLT, the first transformation, B', decorrelates the supervectors. The transformed supervectors will remain uncorrelated under any further purely rotational transform. Since the transformed supervectors are uncorrelated, it is reasonable to assume that the weights will be linearly independent of one another.

As in the derivation of normal MLED adaptation, the columns of the inverse of the transform used to create the "eigenspace" — previously U; now $(W')^{-1}$ — are referred to as eigenmeans ($\mathbf{e}$), and the number of eigenmeans employed is E. The new estimate of the supermean vector for a speaker is then given by

$$\hat{\mu} = \sum_{j=1}^{K} w_j \mathbf{e}_j \tag{7.39}$$

where $\mathbf{e}_j$ is the column number j of $(W')^{-1}$.

Alternatively, for each of the model mean vectors individually

$$\hat{\mu}_m^{(s)} = \sum_{j=1}^{K} w_j \mathbf{e}_m^{(s)}(j). \tag{7.40}$$

Equation 7.40 is exactly the same as Equation 5.4 of the original MLED derivation, and the assumption of linear independence of eigenweights is still valid. The rest of the derivation for MLED utilising CBKLT is identical to that of the normal MLED derivation, with the same resulting set of equations to determine the eigenweights.

It is interesting to note that even though the transformation is class-based and the columns of the $(W')^{-1}$ matrix are not really the same as eigenvoices, the same adaptation equations still apply. Save for the generation of the eigenmeans, which is different, the computational complexity and computer code is identical for the original MLED and the new CBKLT MLED. Considering that CBKLT is a more complex dimension reduction technique, the similarity of the adaptation is quite a windfall.

It should be noted that the CBKLT does not preserve the mean of the original data it transforms, resulting in similar problems as those discussed in Section 7.2. Creating a mean-preserving CBKLT MLED is the same as the creation of the mean-preserving MLED implementation derived in Section 7.2, save that the eigenvoices are created using the CBKLT instead of the KLT. As the modelling of the original mean of the SD models will be exceptionally poor when using CBKLT-extracted eigenvoices, the mean-preserving CBKLT MLED implementation will always be used.

7.4 Strengths and Weaknesses of the Novel Approaches

Both methods share the weaknesses associated with the standard MLED implementation. In addition to these weaknesses, the mean-preserving CBKLT-based MLED implementation has one more weakness: performance is linked to the number of speaker-classes, and the separation between them. If there are too few speaker classes, the number of eigenvoices will be very low. This is acceptable for robust estimation when there is an extremely small amount of adaptation data, but is unacceptable for accurate estimation when there is a moderate amount of adaptation data. For such conditions, standard MLED will outperform the class-based MLED, as a higher-dimensional eigenspace (that captures more inter-speaker variance) may be employed. When the separation between speaker classes is small (and there is little inter-class variance), standard MLED will once again outperform the mean-preserving CBKLT-based MLED implementation.

Chapter 8

Experiments using the ISOLET Speech Corpus

Several factors made the ISOLET speech corpus [37] an excellent choice for initial experimentation with speaker adaptation techniques:

- ISOLET was the speech corpus used by Nguyen [40] for MLED experimentation. Therefore, a set of results were available with which to compare the performance of the speaker adaptation software.
- MLED is one of the adaptation methods that will be tested using the ISOLET corpus, therefore the speech corpus needs to represent enough speech from enough speakers so that a robust set of eigenvoices may be determined. ISOLET contains speech from 150 speakers, which is enough for the creation of a large enough SD model set from which eigenvoices can be robustly determined. There is relatively little speech available for each speaker, however, as the set of SD models will be trained using rapid speaker adaptation methods, and thus will not present a problem.
- ISOLET is a relatively small speech corpus, making the design of small and time-efficient experiments possible. It is thus an excellent testing ground for new ideas.
- ISOLET contains high-quality speech, so the adaptation methods could be tested under highly favourable conditions. Furthermore, each utterance in the corpus contains a large voiced portion. Voiced speech is considered to contain more speaker-

specific information than unvoiced speech, so speaker adaptation experiments performed on ISOLET should have very favourable results.

The rest of the chapter is arranged as follows:

- A succinct description of the ISOLET speech corpus.
- Details on the extracted feature vectors and normaliser.
- Details on the HMM structures used to model spoken letters and silence.
- A set of ML and MAP adaptation experiments.
- A set of MLLR adaptation experiments.
- A set of MLED adaptation experiments.
- Experimental results for the repetition of Nguyen's experiments that test the effect of varying adaptation data on MLLR and MLED.
- The experiments are discussed, and conclusions are drawn.

8.1 The ISOLET Speech Corpus

ISOLET (release 1.1) speech corpus [37] comprises of utterances of the letters of the English alphabet produced in isolation. Two utterances of all the letters of the alphabet (produced in isolation of each other) are available for each of 150 speakers. This amounts to a total of 7800 spoken letters, which is approximately 1.5 hours of speech. The speech quality is considered to be high.

Speakers were between the ages 14 and 72, the average age being 35. 75 were male, and 75 female. The ISOLET corpus is subdivided into 5 subsets, ISOLET-1, ISOLET-2, ISOLET-3, ISOLET-4 and ISOLET-5, each consisting of the utterances of 30 speakers, 15 male and 15 female.

There is approximately 80ms of silence at the beginning and end of each utterance, for a total of 160ms silence per utterance.

Speech was sampled at 16kHz, and the resulting WAV files are stored using the RIFF standard. The files are 16-bit linearly encoded.

8.2 Choice of Features and Extraction Thereof

Nguyen performed his experiments using eighteen-dimensional feature vectors, where the elements comprised of eight PLP-cepstral features, the zeroth PLP-cepstral coefficient or energy, and their corresponding nine delta-parameters. We wish to use his results as a reference for some of the experiments, and we also wish to compare the performance of PLP-cepstral and LPCC feature vectors. The creation process of each of the two types of feature vectors is described next.

8.2.1 LPCC Feature Extraction used for the ISOLET Corpus

First, the speech in the WAV files is loaded, and blocked into 32 ms frames, with an overlap of 16 ms between frames and the power in the utterance is made unity. The spectrum of each frame is then preemphasised. Each frame is then Hamming windowed (the window coefficient is 0.54), and the LPCCs for each frame are computed. Nine LPCCs (including the zeroth cepstrum or energy in the spectrum) are computed. A mean value of each LPCC parameter is now removed, and a delta-parameter is computed for each. The feature vector thus consists of nine mean-compensated LPCCs, and nine delta-parameters. Finally, the features are normalised by their standard deviation.[1]

8.2.2 PLP Feature Extraction used for the ISOLET Corpus

The power in the utterance (file) is made unity. The digitised speech is blocked into 32 ms frames, with an overlap of 16 ms between frames. The spectrum of each frame is then preemphasised. Each frame is then Hamming windowed (window coefficient 0.54), and nine PLP-cepstral coefficients for each frame is determined. The first cepstral coefficient is the energy of the signal. The mean value of each PLP-cepstral parameter is now removed, and delta-parameters for all PLP-cepstral parameters are computed. Finally, the features are normalised by their standard deviation.

[1]Note that only the training data of the SI model was used to estimate the normaliser, i.e. the first utterance of each letter of the alphabet for each speaker in four of the ISOLET speaker subsets. There was thus no contamination of the adaptation training and testing data with the SI model training data.

8.3 Choice of HMM to Model ISOLET Letters

In order to model the speech present in the ISOLET corpus, 27 HMMs are needed — one HMM for each letter of the alphabet, and one HMM for silent sections present in the recorded speech.

Each letter was modelled by an 8-state, left-to-right, double skip-width HMM. Only the six middle states had pdfs associated with them. The self-loop (i.e. a state transition probability a_{ij} where $i = j$) for each emitting state was initialised with a value of 0.7, and all other transition probabilities from a state were made equal. Each emitting state has a GMM state pdf.

A five-state ergodic HMM structure was used to model the silent parts of the recorded speech. The first and last states were beginning and ending null-states, so that only the three middle states had pdfs associated with them. The initial loop-back probability (self-loop) for each non-null state was set to 0.5, and all other state transition probabilities from a state were made equal so that the sum of all transition probabilities leaving the state was equal to one. Each emitting state has a GMM state pdf.

Speaker adaptation experiments are performed on two speech modelling schemes. The first is simple modelling scheme, where each GMM has a single mixture component, i.e. single Gaussian state pdfs are used. The second is a more complex modelling scheme; GMMs with eight mixture components were used for each state pdf.

MLLR adaptation in its basic form can only be used where the GMM state pdfs have diagonal covariance matrices. As such, all HMM models were restricted to use GMMs (or single Gaussian pdfs) with diagonal covariance matrices.

8.4 General Setup for Experimentation

All experiments conducted using the ISOLET corpus are identical to that used by Nguyen [40], save for the following differences and additions:

- Nguyen only did detailed MLLR experiments for HMMs with GMMs with six mixture components, and only did detailed MLED experiments on HMMs with single Gaussian state pdfs. We perform detailed experiments of MLLR and MLED adaptation for models incorporating GMMs as well as for models incorporating single

Gaussian pdfs. In this way the results could be compared to Nguyen's (where available), and observations could be made with regard to the effect of model order on the performance of an adaptation technique.

- All experiments were done for systems using PLP-cepstral features as well as for systems using LPCC features. Experiments using PLP-cepstral features could be verified against Nguyen's, as his experiments were performed using PLP-cepstral features.

- The two novel MLED extensions were included in all experiments, so that their performance could be measured against the original MLED adaptation implementation.

- The experiments conducted to test the effect the number of eigenvoices have on recognition accuracy are done at finer intervals than Nguyen's experiments, to see if the eigenspace extracted from ISOLET speakers has the same characteristics as the eigenspace extracted from DARPA speakers in Westwood's experiments [51].

- The experiments include graphs of per-letter recognition accuracy, as this helps illustrate the difference between PLP-cepstral features and LPCC features.

The 5 speaker-subsets of the ISOLET corpus are used in a round-robin fashion so that speaker adaptation experiments can be performed on each of the sets, and the average of the recognition performance of all sets can then be determined. In this fashion speaker adaptation experiments could be performed for 150 speakers, so that taking the average gives a good indication of the adaptation performance.

For each target subset (consisting of 30 speakers), an SI model is trained using the speech from the other four subsets. The iterative procedure described in Section 2.7 is used to robustly train the SI model. An SD model is now trained for each speaker in the target subset, where the available adaptation data consisted of only the first utterance of each letter for the speaker. Note that the SD model has the same architecture as the SI model. The test data consists of the second utterance of each letter for the speaker.

The final letter error rate for an adaptation method is computed by by taking the average of the letter error rates for each of the five SD sets trained by the adaptation

method. The formula used to compute the letter error rate for each of the five ISOLET subsets is given by:

$$\text{letter error rate} = \frac{\text{(number of observed letters incorrectly recognised)}}{\text{(number of observed letters)}} \times 100\%, \tag{8.1}$$

and the formula for the average letter error rate is given by:

$$\text{average letter error rate} = \frac{1}{5}\sum_{i=1}^{5} \text{letter error rate for ISOLET-}i. \tag{8.2}$$

The recognition rate is given by

$$\text{average letter recognition rate} = 100 - \text{average letter error rate}. \tag{8.3}$$

Henceforth the average letter error rate will be referred to as ER, and the average recognition rate will be referred to as RR.

For all SI and SD model training and adaptation procedures, the Viterbi algorithm was used to obtain the state and mixture occupation probabilities. For scoring purposes, the Viterbi algorithm was again used. Viterbi was preferred to Baum-Welch, because the Baum-Welch algorithm was computationally very expensive when compared to Viterbi, and preliminary experiments indicated that there was little difference between error rates for the two segmentation approaches.

Adaptation methods were only allowed to reestimate the mean vectors of Gaussian pdfs in the HMMs, and all other HMM parameters were fixed.

Any interesting results found when comparing different adaptation methods were verified using the McNemar test [19]. The McNemar test is a is a method used to check whether the difference in error rates between two algorithms tested on the same data set is statistically significant, i.e. to check whether the difference between the error rates is due to the algorithms, or simply due to the (limited) data set. The McNemar test is used to determine the probability of the following hypotheses: The number of utterances which the first algorithm classifies correctly and the second algorithm classifies incorrectly is equal to the number of utterances the first algorithm classifies incorrectly and the second algorithm classifies correctly. Thus, if the McNemar probability is close to one, it

Single pdf PLP-cepstra	Single pdf LPCC	Mixture pdf PLP-cepstra	Mixture pdf LPCC
16.15%	16.18%	11.76%	13.22%

Table 8.1: SI recognition results for various speech modelling systems. Single pdf refers to a system using HMMs with single Gaussian state pdfs. Mixture pdf refers to a system using HMMs with eight-component GMM state pdfs. PLP-cepstra refers to systems using PLP-cepstral features, and LPCC refers to systems employing LPCC features.

is very likely that the algorithms had equivalent performance and any difference in error rate was simply due to the limited data set. If the McNemar probability is close to zero, it is likely that the algorithms are not equivalent, and any difference in error rate is due to the algorithms and not the data set. Typically a significance level is chosen, and if the McNemar probability is less than the significance level, the hypotheses that the two algorithms are equivalent is rejected.

8.5 Letter Error Rates for SI models

Table 8.1 gives the average letter error rates for the SI models for the various modelling systems tested. As expected, systems employing mixture pdfs have lower error rates than systems employing single pdfs. Also, SI systems using PLP-cepstral features outperform systems using LPCC features. In all subsequent sections, the following notation will be used to denote the various systems:

- "SG PLP" will refer to a system that has HMMs using single Gaussian state pdfs, and uses PLP-cepstral features.
- "SG LPCC" will refer to a system that has HMMs using single Gaussian state pdfs, and uses LPCC features.
- "GMM PLP" will refer to a system that has HMMs using GMM state pdfs (each with eight mixture components), and uses PLP-cepstral features.
- "GMM LPCC" will refer to a system that has HMMs using GMM state pdfs (each with eight mixture components), and uses LPCC features.

	System			
	SG PLP	SG LPC	GMM PLP	GMM LPC
SI ER	16.15%	16.18%	11.76%	13.22%
ML ER	15.66%	15.30%	15.94%	15.94%

Table 8.2: Error rates for various systems after ML-reestimation. ER is the letter error rate as defined in Section 8.4.

8.6 ML and MAP Adaptation Experiments

This section treats ML and MAP adaptation on the ISOLET corpus. The specific experimental setup and the results of the experiments are treated in the next section. In Section 8.6.2 the ML and MAP adaptation results are discussed in depth.

8.6.1 Experimental Setup and Results for ML and MAP Adaptation

ML adaptation is the normal method used to train and reestimate the model parameters of an HMM. As such, comparing the recognition performance of ML-reestimated SD models with speaker adapted SD models will give a good indication of the abilities of the speaker adaptation methods.

To prevent non-robust ML-estimation, a limit was placed on the minimum number of observation vectors needed for the reestimation of a mean vector. Previous experience with PatrecII code suggested that a minimum observation count of 8 (per mixture component) was sufficient for robust estimation of a mean vector. Henceforth the notation O_{min} will be used to refer to the minimum observation count.

Table 8.2 summarises the performance of ML-adaptation when using all available adaptation data (i.e. the first utterance of each letter of the alphabet) for each speaker.

MAP-reestimation of HMM parameters can be seen as a more robust version of ML-reestimation. In addition to the τ parameter, the confidence measure in the new ML-estimate, a minimum observation count (as with ML-estimation) of one, four or eight observation vectors is used. There are thus two factors controlling the "robustness" of a new estimate: the τ parameter, and the minimum observation count.

Adaptation method	SG PLP	SG LPCC
SI	16.15%	16.18%
MAP, $\tau = 100$, $O_{\text{min}} = 8$	11.91%	10.54%
MAP, $\tau = 40$, $O_{\text{min}} = 4$	13.76%	11.93%
MAP, $\tau = 30$, $O_{\text{min}} = 1$	12.52%	10.23%
MAP, $\tau = 40$, $O_{\text{min}} = 1$	11.98%	10.50%

Table 8.3: A comparison of best letter error rates achieved for SG systems after MAP-reestimation.

Adaptation method	GMM PLP	GMM LPCC
SI	11.76%	13.22%
MAP, $\tau = 30$, $O_{\text{min}} = 8$	10.29%	8.33%
MAP, $\tau = 10$, $O_{\text{min}} = 4$	11.13%	11.95%
MAP, $\tau = 10$, $O_{\text{min}} = 1$	10.29%	8.83%
MAP, $\tau = 20$, $O_{\text{min}} = 1$	9.77%	8.94%
MAP, $\tau = 30$, $O_{\text{min}} = 1$	9.80%	9.16%

Table 8.4: A comparison of best letter error rates for GMM systems after MAP-reestimation.

The heuristic parameter was chosen to be the same for all the mean vectors in all the HMMs of a speaker model. Various values of this single heuristic parameter τ were then tried to obtain an empirical indication of which values yield the best results. Table 8.3 contains the lowest error rates achieved using MAP-reestimation of the parameters of the SG PLP and SG LPCC systems. Table 8.4 contains the lowest error rates achieved using MAP-reestimation of the parameters for the GMM PLP and GMM LPCC systems. For more results, see Tables E.1 and E.2.

8.6.2 Discussion on ML and MAP adaptation results

From Table 8.2 it is seen that ML-adaptation results in a small improvement in the letter error rate. This is as the amount of adaptation is too small for robust and accurate ML

adaptation.

From Tables 8.3 and 8.4, it is seen that MAP adaptation always resulted in an improvement in error rate, and that the improvement was considerable for certain choices of τ and minimum observation count. It appears that using a minimum observation count for "doubly robust" MAP adaptation results in poorer performance than when no minimum observation count was used. For an SG LPCC system, the improvement was as high as 5.95%, which is a relative reduction of 36.77% of the SI error rate of 16.18%. For an GMM LPCC system, the improvement was as high as 4.83%, which is a relative reduction of 33.21%.

For the case of SG systems, the error rates after MAP adaptation are comparable to the error rates achieved using MLLR or MLED adaptation (see Tables 8.5 and 8.7). This indicates that the number of parameters to adapt given the amount of available adaptation data is such that parameter spreading is unnecessary for adequate reestimation of the speaker model (this is only true when there was adaptation data for each letter).

For the case of GMM systems, the error rates after MAP adaptation are — while better than ML adaptation — not comparable to the rapid adaptation techniques. The reason for this is that the number of parameters that need to be robustly estimated from the available adaptation data have increased eightfold. The rapid adaptation techniques have made more robust estimation possible by effectively spreading the available data.

The relative improvement in error rate was always greater for LPCC systems than for PLP systems. MAP-adapted SG LPCC systems always had lower error rates than SG PLP systems. MAP-adapted GMM LPCC systems often outperformed MAP-adapted GMM PLP systems, even though the initial PLP SI model has a 1.46% advantage over the initial LPCC SI model.

Contrary to the findings of Nguyen [40], we found that MAP adaptation performance was rather sensitive to the value of τ. This suggests that a better performance could be achieved if a genetic algorithm (or similar method) were used on the set of training speakers to determine the best τ values. Furthermore, the performance could probably be enhanced by using a separate τ for each mean vector. The iterative Bayesian method [18] may also be used to obtain good τ values.

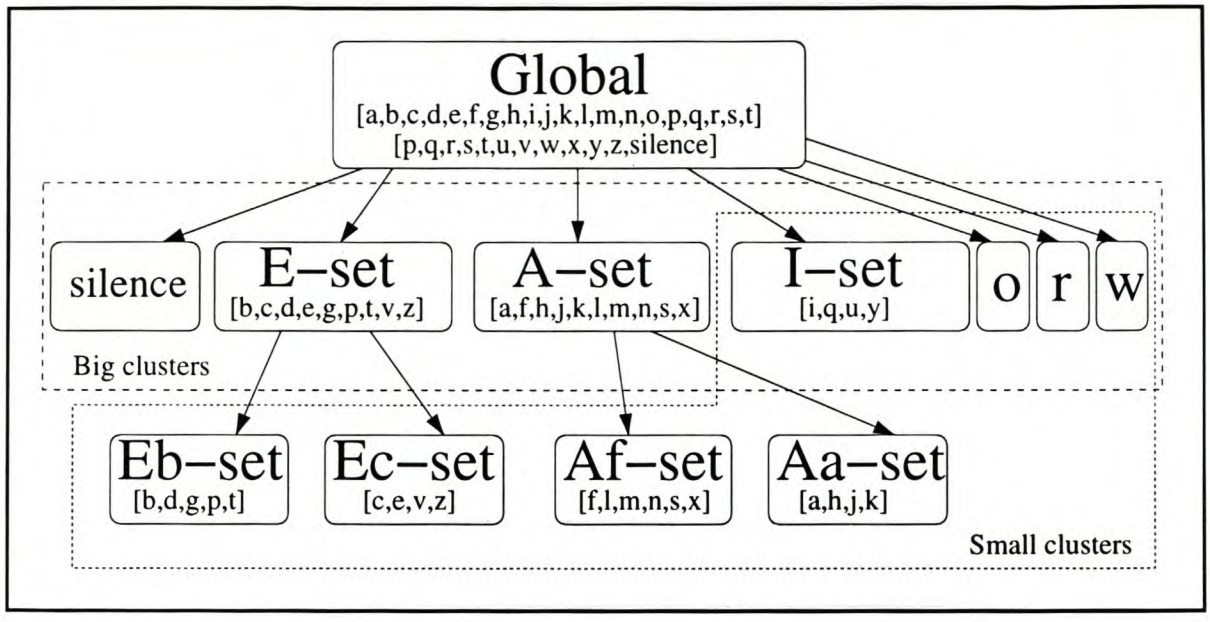

Figure 8.1: Regression tree used for the ISOLET corpus (adapted from Nguyen [40]).

8.7 Setup for the MLLR Adaptation

The regression tree used during the experiment is identical to the one proposed by Nguyen [40] for his ISOLET experiments on MLLR, and is reproduced in Figure 8.1.

Nguyen never used the entire tree during his MLLR experiments, but tested the following clusterings for four different experiments:

- Global regression: this regression class binds all the letters' models as well as the silence model.
- Big clusters: phonetically similar letters are grouped according to the following large regression classes:
  - A set containing only the silence model.
  - An E-set composed of the set [b, c, d, e, g, p, t, v, z].
  - An A-set composed of the set [a, f, h, j, k, l, m, n, s, x].
  - Separate sets for each of the letters o, r and w, to be adapted by the global matrix.

- Small clusters: big clusters are split in order to exploit further phonetic similarities between constituent letters.
  - A plosive subset of the E-set (referred to as the Eb-set), composed of the letters [b, d, g, p, t].
  - The remainder of the E-set (referred to as the Ec-set), composed of the letters [c, e, v, z].
  - A subset of the A-set (referred to as the Af-set), composed of the letters [f, l, m, n, s, x].
  - A second subset of the A-set (referred to as the Aa-set), composed of the letters [a, h, j, k].
  - The I-set and separate sets for the letters o, r and w remain the same as they were for the big clusters.
- Per letter: a separate regression class for each letter on its own.

As some results will be compared to Nguyen's, initial ISOLET experiments are to be done using the same clustering schemes. Subsequent experiments use the entire phonetically determined regression class tree.

Nguyen included the silence model in his global regression class, but it is not known whether the silence model was adapted using the global regression matrix. It was decided to altogether remove the silence model from the adaptation, for two reasons. Firstly, it does not contribute any speaker-specific knowledge that could be helpful during adaptation, as the recording conditions were identical. Secondly, the other letter models are vastly different from the silence model, both in structure and in the feature data used to train them. As such, adapting the silence model using a global regression matrix formed from letter data would not improve the SI silence model. Furthermore, initial experimentation indicated that the inclusion or exclusion of the silence model in the adaptation procedure had little effect on recognition performance. The exclusion of a silence model is not advisable for all adaptation tasks. For instance, using the "silence" feature vectors produced by motor-car noise could assist MLLR adaptation of a speaker model to compensate for the recording environment.

	SG PLP	GMM PLP	$O_{\min}$
SI	16.15%	11.76%	-
MLLR, global	10.37%	7.66%	Gl: 200
MLLR, big clusters	10.80%	7.67%	Gl, BC: 200
MLLR, small clusters 1	13.65%	12.02%	Gl, o, r, w: 200 & SC: 8
MLLR, small clusters 2	13.04%	7.83%	Gl, o, r, w: 200 & SC: 100
MLLR, per letter 1	96.30%	—	Gl: 200 & PL: 8
MLLR, per letter 2	10.37%	7.66%	Gl: 200 & PL: 100
MLLR, tree 1	15.68%	8.02%	Gl, BC: 200 & SC, o, r, w: 50
MLLR, tree 2	13.36%	8.02%	Gl, BC, o, r, w: 200 & SC: 100
MLLR, tree 3	10.85%	8.02%	Gl, BC, SC, o, r, w: 200

Gl: Global; BC: Big Clusters (excluding o, r and w)

SC: Small Clusters (excluding o, r ,and w)

PL: per letter (o, r and w are included)

The silence model was entirely excluded from adaptation.

Table 8.5: A Comparison of MLLR Letter Error Rates for PLP systems.

Results for an extensive set of experiments testing MLLR adaptation performance on PLP systems using various regression class schemes are contained in Table 8.5. A second set of experiments for MLLR adaptation of LPCC systems using only the best regression class schemes is contained in Table 8.6. All of these experiments are done with all the available training data for each speaker, i.e. the first utterances of each letter of the alphabet for the speaker is used.

8.7.1 Discussion of MLLR Adaptation Results

For the SG models, the MLLR using the best regression class schemes only had slightly better performance than MAP adaptation, indicating that there was enough adaptation data per parameter to make robust MAP estimates. For the GMM models, however, most MLLR regression class schemes outperform MAP adaptation, indicating that parameter reduction is becomes necessary for robust parameter estimation.

	SG LPCC	GMM LPCC	$O_{\min}$
SI	16.18%	13.22%	-
MLLR, global	9.36%	7.62%	Gl: 200
MLLR, big clusters	9.36%	6.87%	Gl, BC: 200
MLLR, small clusters 1	9.31%	9.76%	Gl, o, r, w: 200 & SC: 8
MLLR, tree 1	13.14%	7.59%	Gl, BC: 200 & SC, o, r, w: 50
MLLR, tree 3	9.36%	7.59%	Gl, BC, SC, o, r, w: 200

Table 8.6: A Comparison of MLLR Letter Error Rates for LPCC systems.

In all cases the post-adaptation error rates for LPCC systems are lower than their counterpart PLP systems.

For the "per letter 1" implementation of MLLR, the error rate was 96.30% — by far the highest error rate for any MLLR implementation. This is because the minimum occupancy (eight) was far lower than the dimension of the feature vectors (eighteen), so that the resulting $G^{(i)}$-matrix is rank deficient, and therefore not invertable (as discussed in Section 4.3.2). A pseudo-inverse is therefore calculated (as suggested by Leggetter [33]) in place of the inverse to obtain a least-squares approximation of the true solution. However, the extremely poor adaptation performance indicates that using a pseudo-inverse when the $G^{(i)}$-matrix is very rank deficient is not good practice. It is therefore our suggestion that the pseudo-inverse be used with caution (or not at all) when the regression class occupancy is low.

The error rates for MLLR adaptation are generally two to three percent lower than those achieved by Nguyen. This is probably because our SI system had a slightly lower error rate (1% lower), and as a result of the different limitations placed on the minimum observations needed to estimate a regression matrix.

Even though several regression class schemes were employed, few had better results than a simple global regression class. This is probably due to the very basic phonetic regression tree employed. Using more complex clustering schemes — be it phonetically or distance based — performance better than that of the global regression class scheme should be possible.

8.8 Setup for MLED Adaptation

Before MLED adaptation can be performed, a set of eigenvoices is needed. These, in turn, have to be calculated via PCA from supervectors extracted from a set of well-trained SD models.

Adaptation experiments are to be performed on one ISOLET speaker-subset at a time. As stated before, the SI model for the current target subset is trained using the data from the other four subsets. For MLED adaptation, a set of 120 SD models are determined for each of the speakers in the other four subsets. These SD models are trained by first using MLLR adaptation, and then using MAP-reestimation as a computationally light-weight post-adaptation step, to increase the SD models' likelihood slightly. For instance, if ISOLET-1 is the targeted speaker subset on which MLED adaptation is to be tested, the data from ISOLET-2 to ISOLET-5 will be used to train the SI model. The same training data is then used to adapt this SI model to an SD model for each speaker in subsets ISOLET-2 to ISOLET-5. Eigenvoices are then determined from the SD models trained for the speakers in ISOLET-2 to ISOLET-5, so that MLED adaptation may be performed on ISOLET-1.

Note that the HMM modelling silence was included in the computation of the eigenvoices as well as in the final adaptation, unlike MLLR where the silence model was entirely excluded from the adaptation process.

The silence model is excluded from MLLR adaptation, because we are not concerned with environmental adaptation. Adaptation data from the silence sections is therefore not likely to make a positive contribution to the MLLR adaptation of non-silence letter models. Furthermore, as the recording environment is the same, the silence portions will be highly similar. It is therefore not viable to poorly adapt the silence models using a small amount of similar adaptation data, because a well-trained SI silence model, which is robustly estimated using a large amount of similar adaptation data, is available.

The silence model is included for MLED adaptation, as it is not likely to affect adaptation performance. The recording environment is the same for all speakers, and so the silence sections will be the same for all speakers. This means that the inter-speaker variance that can be extracted from the silence data of various speakers will be very small, and will thus have a very small role in the estimation of MLED eigenvoices. Inclusion of

silence data for MLED adaptation is thus not likely to have a great effect on adaptation performance. This is only true as the recording environment is the same for all speakers. If the recording environment is not the same, the silence portions could play an important role in environmental adaptation.

Three types of MLED adaptation were performed in each of the experiments:

1. Normal MLED, where PCA is used to extract the eigenvoices from the correlation matrix of the set of SD supervectors. This MLED implementation will be labelled "MLEDcor".

2. Mean-preserving MLED, where PCA is used to extract the eigenvoices from the covariance matrix of the set of SD supervectors. This MLED implementation will be labelled "MLEDcov".

3. Mean-preserving MLED with CBKLT-derived bases or eigenvoices. For ISOLET, CBKLT was performed using two classes: one for male speakers, and one for female speakers. As there are only two classes, there is at most one eigenvoice that can be used for adaptation. This MLED implementation will be labelled "MLEDcbklt".

Results for MLED adaptation of the various SI systems are contained in Table 8.7. All of these results were obtained using all the available training data for each speaker, i.e. the first utterance of each letter of the alphabet produced by the speaker.

Figures 8.2 to 8.5 show that the letter recognition rate increases as the number of eigenvoices employed is increased.

Only one adaptation iteration was used in the MLED experiments listed thus far. To test the effect of using more than one iteration on recognition performance, a simulation using three adaptation iterations of a mean-preserving covariance-based MLED implementation on an LPCC GMM system was completed, and the results are shown in Figure 8.6.

8.8.1 Discussion of the MLED Adaptation Results

From Table 8.7, and Figures 8.2 to 8.5, it is clear that mean-preserving covariance-based MLED implementation (MLEDcov) always outperforms standard MLED (MLEDcor)

	SG PLP	SG LPCC	GMM PLP	GMM LPCC
SI	16.15%	16.18%	11.76%	13.22%
MLEDcor, $K = 1$	15.21%	15.93%	11.03%	13.23%
MLEDcor, $K = 10$	11.30%	11.64%	7.83%	9.04%
MLEDcor, $K = 20$	11.15%	10.63%	7.46%	8.66%
MLEDcov, $K = 1$	13.28%	14.13%	8.81%	12.00%
MLEDcov, $K = 10$	11.32%	11.34%	7.96%	9.53%
MLEDcov, $K = 20$	11.15%	10.53%	7.34%	8.54%
MLEDcbklt, $K = 1$	13.46%	14.55%	—	—

Table 8.7: A comparison of letter error rates for SG and GMM systems after adaptation. K is the number of eigenvoices used.

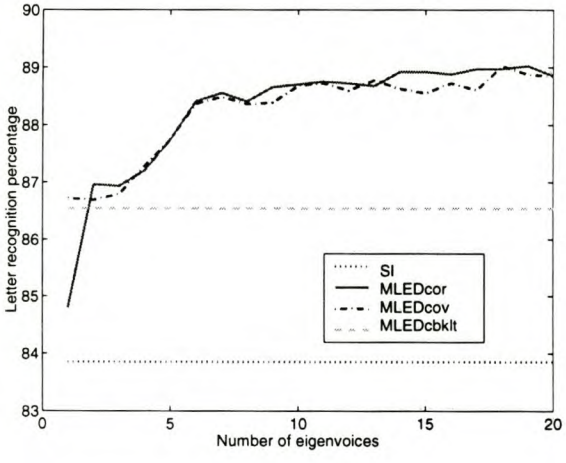

Figure 8.2: The effect of the number of eigenvoices on the performance of MLED adaptation for SG PLP systems.

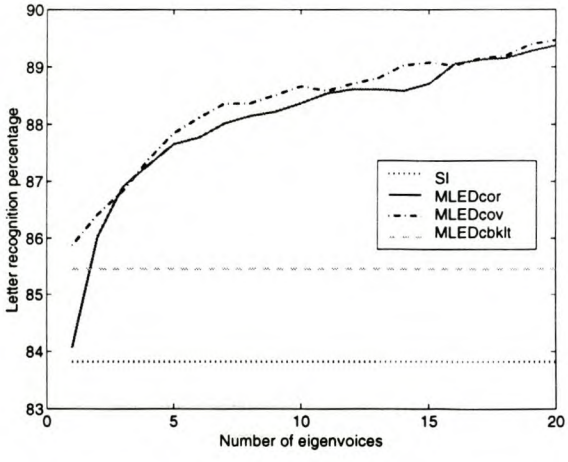

Figure 8.3: The effect of the number of eigenvoices on the performance of MLED adaptation for SG LPCC systems.

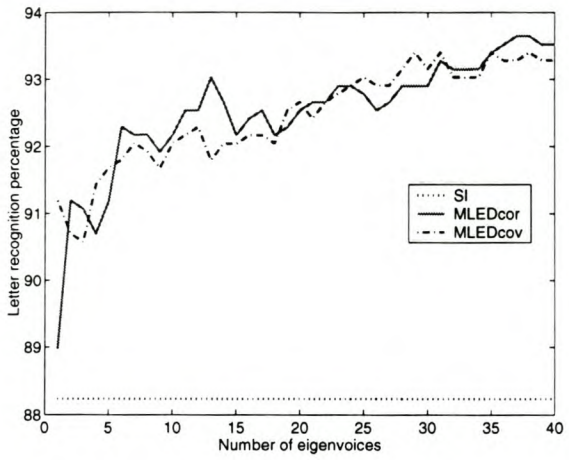

Figure 8.4: The effect of the number of eigenvoices on the performance of MLED adaptation for GMM PLP systems.

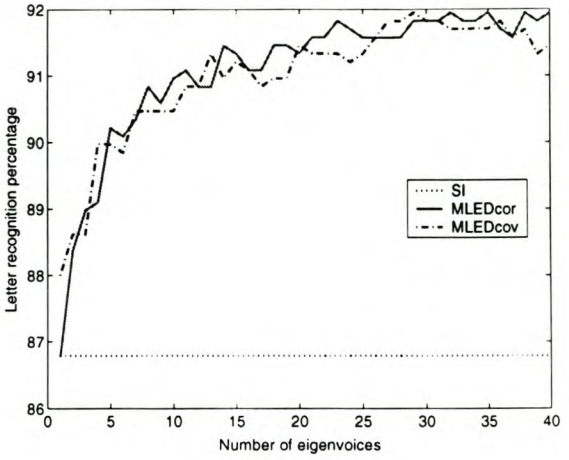

Figure 8.5: The effect of the number of eigenvoices on the performance of MLED adaptation for GMM LPCC systems.

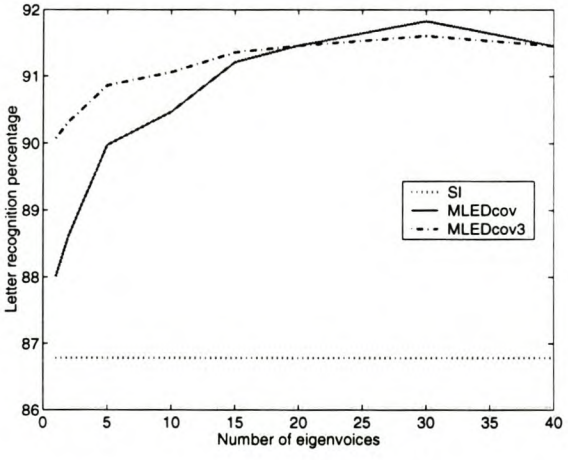

Figure 8.6: The effect of using more than one adaptation iteration on recognition performance. The MLEDcov plot was obtained using a single adaptation iteration, and the MLEDcov3 plot was obtained using three adaptation iterations.

when only one eigenvoice is used. When more than one eigenvoice is employed, however, there is little difference in performance between the two MLED implementations.

For the case where one eigenvoice was used, the CBKLT-based MLED implementation (MLEDcbklt) had slightly better performance than the standard MLED implementation (MLEDcor), and similar performance to the mean-preserving covariance-based MLED implementation. Only one eigenvoice could be used, as the classes used for the CBKLT computation of eigenvoices consisted of a male and female speaker class. The CBKLT-based MLED performance indicates two things: one, mean-preservation enhances adaptation performance, and two, an eigenvoice aligned to the inter-class variance between male and female speaker classes gives the same performance as an eigenvoice aligned to the direction of greatest inter-speaker variance.

Nguyen showed that increasing the number of eigenvoices used, results in increased post-adaptation recognition performance. He only showed the results for using one, two, three, five, eight, ten, fifteen, twenty and thirty eigenvoices, and the resulting recognition rate versus eigenspace dimension curve was relatively smooth and almost monotonically increasing. Westwood, however, plotted recognition rate versus eigenspace dimension graphs for systems of varying complexity — ranging from monophones modelled by HMMs using single Gaussians, to triphones modelled by HMMs using six component GMMs, all for the DARPA Resource Management corpus [38] — for an eigenspace dimension of one to thirty. For the simple models, the recognition rate increased as the number of eigenvoices were increased up to fifteen eigenvoices, after which the recognition rate increased and decreased erratically. We suspect that this is possibly due to numerical instability, as Westwood estimates many eigenvoices, and each eigenvoice has a very large dimension, using a very small amount of adaptation data (see Section 6.4). Another possibility is that the pseudo-inverse is being used. Use of a pseudo-inverse will give the least-squares solution to the problem, which is not necessarily close to the true solution. For the complex models, the recognition rate increased to five eigenvoices at best, after which the recognition rate once more became very erratic. In all cases, the performance behaviour became more unpredictable when less adaptation data was available. Westwood cites non-robust estimation of the eigenvoices as the reason for the unpredictable behaviour. We wanted to see if eigenvoices for the ISOLET corpus displayed similar traits, as Nguyen's

plots were incomplete.

Figures 8.2 to 8.5 of this thesis are plots of recognition rate versus eigenspace dimension, where the dimension ranges from one to twenty for SG systems, and from one to forty for GMM systems. For SG systems, the recognition rate does not behave unpredictably as the number of eigenvoices are increased, and the curve is smooth and increasing. For the GMM systems, the curve is still increasing and relatively smooth for eigenvoices one to forty. The curve is less smooth than is the case for SG systems, but does not display the same unpredictable behaviour observed by Westwood, where the recognition rate would rise and drop with up to 3% as the eigenspace dimension is increased by one.

For the SG PLP system, the recognition rate versus eigenspace dimension curve climbs rapidly, and reaches a plateau after the eighth eigenspace dimension, after which the increase (if any) is very small. For the SG LPCC system, the recognition rate initially increases slower than is the case for the SG PLP system. However, the recognition rate shows no indication of having reached a plateau for the eigenvoices one to twenty. At twenty eigenvoices, the recognition rate of MLED-adapted SG LPCC systems is higher than the recognition rate of MLED-adapted SG PLP systems. This hints that the largest part of the inter-speaker variability is captured in very few eigenvoices by the SG PLP system, but that the eigenvoices of the SG LPCC system has more true dimensions of inter-speaker variability. It would seem that the use of PLP-cepstral features removes some of the dimensionality of inter-speaker variability, and concentrates the remaining variability into very few dimensions.

Moving to the GMM systems, it is seen that the curve for the GMM PLP system is generally higher than the curve for the GMM LPCC system. The LPCC curve, however, tends to be smoother than the PLP curve. Neither curve shows any indication of reaching a plateau in recognition performance. Of the adaptation methods tested, MLED adaptation of a GMM PLP system is the only PLP system that outperforms its LPCC counterpart, even though PLP systems have better SI models than LPCC systems.

Figure 8.6 indicates that using more adaptation iterations is beneficial to recognition rate when the eigenvoice dimension is less than fifteen.

8.9 Adaptation for Varying Amounts of Data

A set of four experiments testing the effect of varying amounts of adaptation data on the performance of different adaptation methods was completed. The four experiments are:

1. **Sub-experiment 1:** In this experiment, the following nine letters are removed from the training set:

   - A, H, K and L from the A-set.
   - B, E, P and T from the E-set.
   - O.

   The remaining training data was considered to be reasonably balanced.

2. **Sub-experiment 2:** All nine letters from the E-set were removed in order to create a sparse unbalanced training data set. This should have an adverse impact on MLLR adaptation. MLED adaptation should remain unaffected, apart from having less adaptation data.

3. **Sub-experiment 3:** Adaptation methods were only allowed to use the training data of letter A.

4. **Sub-experiment 4:** Adaptation methods were only allowed to use the training data of letter S.

Table 8.8 summarises the results for PLP-cepstral features.

8.9.1 Discussion of Adaptation for Varying Amounts of Data

The MLLR adaptation results are discussed first. From Table 8.8, we observe that MLLR adaptation is adversely affected by the first reduction in the amount of available adaptation (sub-experiment one). When unbalanced adaptation data was used in sub-experiment two, MLLR adaptation becomes a little worse, as is expected. For the last two sub-experiments, where adaptation of only one letter are used, no MLLR adaptation takes place as there are too few observation vectors to robustly estimate a global regression matrix.

	Exp. 1	Exp. 2	Exp. 3	Exp. 4
SI	16.15%	16.15%	16.15%	16.15%
MLLR, global	12.03%	13.40%	16.15%	16.15%
MLLR, big clusters	12.05%	13.34%	16.15%	16.15%
MLEDcor, $K = 1$	15.16%	14.99%	15.38%	15.34%
MLEDcor, $K = 10$	11.32%	11.66%	14.61%	15.83%
MLEDcor, $K = 20$	11.02%	12.08%	20.07%	19.38%
MLEDcov, $K = 1$	13.33%	13.21%	13.06%	13.53%
MLEDcov, $K = 10$	11.47%	11.79%	14.52%	16.02%
MLEDcov, $K = 20$	11.22%	12.19%	19.33%	19.97%
MLEDcbklt, $K = 1$	13.51%	13.48%	13.85%	13.89%

Table 8.8: Error rates for sub-experiments one to four on the SG PLP system.

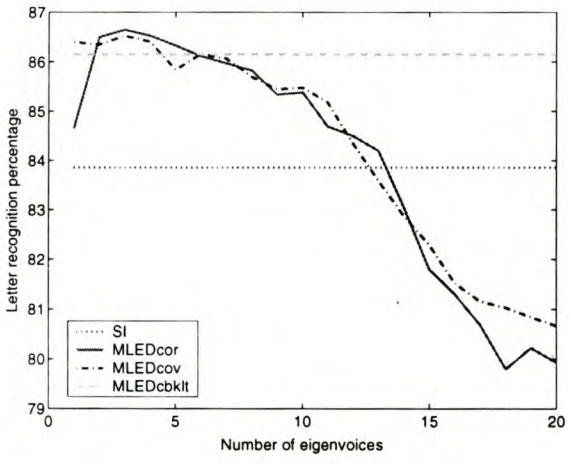

Figure 8.7: The effect of eigenspace dimension on the recognition rate for the SG PLP systems when the amount of adaptation data is very small. Only the data of the letter A were used as adaptation data.

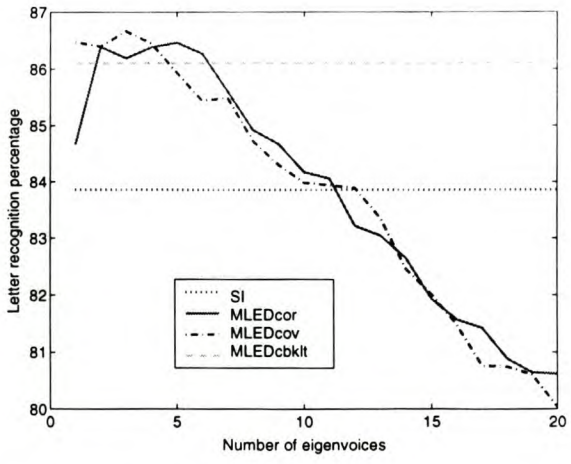

Figure 8.8: The effect of eigenspace dimension on the recognition rate for the SG PLP systems when the amount of adaptation data is very small. Only the data of the letter S were used as adaptation data.

MLED adaptation, on the other hand, yields reduction in error rate even when the adaptation data consists of the observation vectors from one letter. Apart from the mean-preserving CBKLT-based MLED implementation, MLED always outperformed MLLR for certain sizes of eigenspace dimension. This is a reversal of the situation found when all letters are used as adaptation data, where MLLR outperforms MLED.

For the following two reasons, it was hoped that the MLEDcov implementation would allow superior performance:

1. Applying the KLT to the covariance matrix instead of the correlation matrix of the prior set of SD supervectors should result in a more targeted modelling of the directions of highest inter-speaker variance. It was hoped that adaptation would perform better for at least the first few eigenvoices, as the mean is preserved, and the directions of inter-speaker variance are represented better.

2. Preserving the mean of the prior set of SD supervectors means that it does not have to be estimated from the available adaptation data. It was reasoned intuitively that, if the data were not wasted on the estimation of a quantity already known (i.e. the mean of the prior set of SD supervectors), then more eigenweights could be robustly estimated. It was hoped that the MLEDcov implementation would outperform the standard MLED implementation for the first few eigenvoices when the amount of adaptation data was very small.

The mean-preserving covariance-based MLED implementation (MLEDcov) always outperformed the standard MLED implementation (MLEDcor) when only a single eigenvector was used. McNemar [19] tests were done to verify this result. McNemar tests to verify the MLEDcov vs. MLEDcor results for the cases where only one eigenvoice was used were much lower than a significance level of 0.005 — typically of the order 1×10^{-6}. This indicates that the superior performance of MLEDcov is in fact due to the method and not simply due to the specific data set.

When enough adaptation data are available for the robust estimation of many eigenweights, Figures 8.2 to 8.5 show that the performance of the two MLED implementations is very similar, and the more accurate eigenspace modelling of MLEDcov does not give improvement for all eigenspace dimensions. When a very small amount of adaptation

data is available, Figures 8.7 and 8.8 show that the intuition that mean-preservation would allow the robust estimation of more eigenvoices was false, as the performance of standard MLED (MLEDcor) and the mean-preserving covariance-based MLED implementation (MLEDcov) are similar when more than one eigenvoice is employed. The mean-preserving CBKLT-based MLED implementation (MLEDcbklt) could only employ one eigenvoice, and was consequently very robust. It shows very little degradation in performance as the amount of adaptation data is decreased. However, MLEDcov still outperforms MLEDcbklt for one eigenvoice, indicating that a one-dimensional eigenspace based on inter-speaker variance is superior to a one-dimensional eigenspace based on gender-class variance. MLEDcbklt performance is not that much poorer than MLEDcov, indicating that much of the inter-speaker variance is due to gender. This concurs with the results of Nguyen, who found that PCA (KLT) identifies gender as the source of greatest inter-speaker variance.

8.10 Discussion of ISOLET Experiments

From the set of experiments on the ISOLET corpus the following conclusions could be drawn with regards to the behaviour of the adaptation methods:

- **MAP:** When all adaptation data are used (i.e. every first utterance of the alphabet), MAP has similar performance to the rapid speaker adaptation techniques MLLR and MLED for the SG systems. There are relatively few parameters in the SG system and the amount of data was therefore enough that parameter spreading had very little effect on adaptation performance. When GMM systems were reestimated using all adaptation data, MAP was outperformed by MLLR and MLED. The eightfold increase in the number of parameters to be estimated using the same amount of adaptation data made parameter spreading necessary for robust estimation of parameters.

  For the SG experiments where less and less adaptation data were available, MAP adaptation degrades rapidly, and cannot compete with MLLR and MLED.

- **MLLR:** MLLR adaptation gave the best results when all the available training data was used for adaptation, with the exception of the GMM PLP system, where

MLED had slightly better performance. Few regression schemes gave better results than the simple global regression class. For the SG models, this is because the mean vector occupancy of other regression class schemes is very low, making numerical instability determination of regression matrices commonplace. The implementation in this thesis checks the condition numbers of the matrices that are inverted to create the transformation matrix, and if they are too high, adaptation using the particular regression class is not allowed. For the GMM models, the amount of adaptation data is relatively small, making robust estimation of non-global regression matrices difficult.

- **MLED:** MLED adaptation had better results than the other methods when the amount of adaptation data was very small. The parameter reduction of the method is so great that successful robust reestimation of model parameters is possible with as little as one letter utterance. When a great deal of data is available, adaptation performance tends to be on the same order (or slightly less) than the performance of the set of prior SD models from which the eigenspace is created.

  In all instances, the MLEDcov implementation resulted in significantly lower error rate than standard MLED (MLEDcor) for the case where a single eigenvoice is used. When there are very little adaptation data (one letter utterance), the error rate never becomes significantly greater than when a single eigenvoice is used. MLEDcor needs two eigenvoices before it performs as well as MLEDcov, which is computationally more expensive. The MLEDcov implementation may thus be used as a robust computationally light adaptation method when little adaptation data is available.

  MLEDcbklt could only employ one "eigenvoice". However, the adaptation was robust, and indicated that a significant improvement over the baseline SI model performance was possible by compensating for inter-gender speech variance.

On the performance of PLP-cepstral features versus LPCC features, the following observations were made: (a) SI models using PLP-cepstral features had lower error rates, and (b), post-adaptation error rates were lower for systems using LPCC features in all but one case: MLED adaptation of GMM PLP systems was better than MLED adaptation of GMM LPCC systems. Even in this last case, the decrease in the error rate from the

SI system was 4.68% for the LPCC system, which is higher than the 4.42% decrease for the PLP system. McNemar [19] statistical significance tests were used to validate the performance of systems using PLP-cepstral and LPCC feature vectors. The results are summarised in Tables 8.9 and 8.10. The closer the McNemar probability is to one, the more likely it is that any difference between error rates for systems is likely to be due to the specific data set used, and not due to differences between the performance of the systems. The closer the McNemar probability is to zero, the more likely it is that the difference between error rates for the systems is likely to be due to the differences between the systems, and not due to the specific data set used.

First, the results of the McNemar tests for the SG PLP and SG LPCC systems will be discussed (refer to Table 8.9). The McNemar probability for the two SI systems indicates that the differences in performance between the two systems was statistically insignificant. The same goes for the comparison between MLEDcor adapted SG LPCC and SG PLP systems. The McNemar test for MAP indicates that the differences in performance between the LPCC and PLP systems is significant, and that the LPCC system outperforms the PLP system. The McNemar test for the global MLLR does not have such a low probability, indicating that it is quite likely (but not certain) that global MLLR adaptation for LPCC systems has better performance than global MLLR adaptation for PLP systems. The McNemar test for MLEDcor indicates that there is statistical significance between the performance of the LPCC and PLP systems.

The results of the McNemar tests for the GMM PLP and GMM LPCC systems will now be discussed (refer to Table 8.10). The McNemar probability for the SI systems indicate there is a small chance that the difference in performance between the SI systems is due to the data set, even though the error rate for the PLP system is somewhat lower than the error rate for the LPCC system. The McNemar probabilities for the MAP and MLLR ("big clusters") adapted LPCC and PLP systems indicate that the chance that the difference in performance between the LPCC and PLP systems is due to the specific data set is relatively small. The global MLLR adaptation of LPCC and PLP systems is likely to be equivalent. The McNemar probabilities obtained for both SG and GMM systems were relatively high, as the ISOLET corpus is small, and there are few utterances with which to compare the methods. The McNemar tests for the GMM systems generally

	Error rate		
Model	SG LPCC	SG PLP	McNemar probability
SI	16.18%	16.15%	0.87
MAP, $\tau = 40$, $O_{\text{min}} = 1$	10.50%	11.98%	<0.005
MLLR, global	9.36%	10.37%	0.08
MLEDcor, $K = 20$	10.63%	11.15%	0.47

Table 8.9: Error rates and McNemar tests for SG PLP vs SG LPCC systems. The closer the McNemar probability is to zero, the less likely it is that the performance of the PLP and LPCC systems are equivalent. The closer the McNemar probability is to one, the more likely it is that the performance of the PLP and LPCC systems are equivalent, regardless of the difference in error rates.

had larger values than the McNemar tests for the SG systems. This is because the error rates for the GMM systems are lower, and there are fewer instances where the decoding of the utterances differ.

	Error rate		
Model	GMM LPCC	GMM PLP	McNemar probability
SI	13.22%	11.76%	0.11
MAP, $\tau = 40$, $O_{\text{min}} = 1$	10.50%	11.98%	0.03
MLLR, global	7.62%	7.66%	0.53
MLLR, big clusters	6.87%	7.67%	0.04

Table 8.10: McNemar tests for GMM PLP vs GMM LPCC systems. The closer the McNemar probability is to zero, the less likely it is that the performance of the PLP and LPCC systems are equivalent. The closer the McNemar probability is to one, the more likely it is that the performance of the PLP and LPCC systems are equivalent, regardless of the difference in error rates.

SI systems trained on PLP-cepstral features have lower error rates, because PLP-features tend to reduce the inter-speaker variance. When training SD models or adapting SI systems, lower inter-speaker variance means that less specialisation for each SD system

is possible, and higher inter-speaker variance means that more specialisation for each SD system is possible. Intuitively, higher specialisation for a SD system should lead to a higher recognition rate. This explains why LPCC features are generally preferable to PLP-cepstral features when supervised adaptation is performed.

Nothing can be said for the case of unsupervised adaptation, as such experiments were not performed. The higher SI recognition rate for PLP systems may lead to better classification and segmentation of data. For the case where moderate amounts of adaptation data is available, this would certainly lead to better MLLR adaptation. MLED adaptation gives almost constant error rates for moderate to large amounts of adaptation data, therefore better segmentation and classification is unlikely to have a positive influence on adaptation.

Chapter 9

Experiments using the TIMIT Speech Corpus

TIMIT is a larger corpus than ISOLET, in terms of the number of speakers, the amount of recorded speech for each speaker, and the variety of phonemes present in the corpus. The recorded speech for each speaker consists of ten sentences, and TIMIT therefore has a much wider variety of phonemes. Furthermore, the speech is continuous. The phonetic diversity and continuity of the speech makes the TIMIT corpus more complex, and it can therefore be employed to evaluate the performance of the speaker adaptation techniques in a more complex environment.

Furthermore, we wished to test the performance of the mean-preserving CBKLT-based MLED implementation when dialect classes were used. TIMIT contains speech from speakers of seven major dialect regions, and an eighth consisting of speakers not belonging to any of the dialect regions, making it a good choice of speech corpus to test the performance of our class-based MLED implementation.

The TIMIT database was also employed by other researchers working on concurrent projects, so that that the speaker adaptation and speech recognition results of this thesis could be compared against their speech and speaker recognition results.

The rest of the chapter is arranged as follows:

- A succinct description of the TIMIT speech corpus.
- General experimental setup for all TIMIT experiments.

- Detail extracted feature vectors and normaliser.
- Detail of the employed HMM structures.
- Detail of the regression tree used for MLLR adaptation.
- Three experiments testing the various MLED implementations.
- Conclusion of the experiments performed on TIMIT.

9.1 The TIMIT speech corpus

The DARPA TIMIT acoustic-phonetic continuous speech corpus [39] consists of 6300 sentences (roughly five hours and twenty-two minutes of speech), ten sentences spoken by each of 630 speakers from seven major dialect regions of the United States of America, and an eighth group of speakers not belonging to any of the seven regions.

The ten sentences for each speaker are divided into the following groups:

- **2 sa sentences:** Two sentences were designed to expose the dialectal variants of the speakers. These two sentences were spoken by all 630 speakers, and are the sentences labelled "sa" on the CD-ROM. Together these sentences represent an average of 6.2 seconds of speech for a speaker.
- **5 sx sentences:** 450 phonetically-compact sentences were designed to provide good coverage of pairs of phones, with extra occurrences of phonetic contexts of special interest. Each speaker read five of these sentences, labelled as "sx" on the CD-ROM. Together these sentences represent an average of 14.6 seconds of speech for a speaker.
- **3 si sentences:** 1890 phonetically-diverse sentences were selected to add diversity to the sentence types and phonetic contexts. Each speaker reads three of these sentences, labelled as "si" on the CD-ROM. Together these sentences represent an average of 9.9 seconds of speech for a speaker.

The suggested training and testing speaker sets on the CD-ROM were used. The training set consists of 462 speakers, and the test set consists of 168 speakers. The training and

test speaker sets are such that the proportion of male and female speakers, as well as the proportion of speakers from different dialect groups are approximately the same for both speaker sets.

Instead of using the transcriptions for the full set of phonemes, transcriptions for a reduced set of 39 phonemes, as proposed by Lee [32], was used.

The speech in TIMIT was sampled at 16 kHz.

9.2 Choice of Features and Extraction Thereof

For experiments on the TIMIT database, the feature vectors comprised of eleven PLP-cepstral features, the zeroth PLP-cepstral coefficient or energy, and their corresponding twelve delta-parameters.

9.2.1 PLP Feature Extraction used for the TIMIT Corpus

First the power of each utterance (file) is made unity. The digitised speech is blocked into 32 ms frames, with an overlap of 16 ms between frames. The spectrum of each frame is then preemphasised. Each frame is then Hamming windowed (window coefficient 0.54), and nine PLP-cepstral coefficients for each frame is determined. The first cepstral coefficient is the energy of the signal. The mean value of each PLP-cepstral parameter is now removed, and delta-parameters for all PLP-cepstral parameters are computed. Finally, the features are normalised by their standard deviation.

9.3 Choice of HMM to Model TIMIT Phonemes

We employed 39 HMMs to model the 39 phonemes of the reduced phoneme set for the TIMIT corpus. 38 of the phonemes represent speech sections, and the remaining phoneme (labelled "epi") represents silent sections.

The 38 "spoken" phonemes were each modelled by a 5-state, left-to-right, single skip-width HMM. Only the three middle states had pdfs associated with them. The self-loop (i.e. a state transition probability a_{ij} where $i = j$) for each emitting state was initialised with a value of 0.7, and all other transition probabilities from a state were made equal.

Each emitting state has a GMM state pdf.

A five-state ergodic HMM structure was used to model the silent parts of the recorded speech. The first and last state were beginning and ending null-states, so that only the three middle states have pdfs associated with them. The initial loop-back probability (self-loop) for each non-null state was set to 0.5, and all other state transition probabilities from a state were made equal, so that the sum of all transition probabilities leaving the state was equal to one. Each emitting state has a GMM state pdf.

Speaker adaptation experiments are performed on two speech modelling schemes. The first is a simple modelling scheme, where each GMM had a single mixture component, i.e. single Gaussian state pdfs are used. This system will be referred to as the SG system. The second is a more complex modelling scheme; GMMs with eight mixture components were used for each state pdf. This system will be referred to as the GMM system.

As with the experiments on the ISOLET corpus, all the Gaussian pdfs were restricted to have diagonal covariance matrices.

9.4 Regression Class Tree used for MLLR

The regression class tree that was used for MLLR adaptation is phonetically based, and is depicted in Figure 9.1. Based on this regression class tree, the following regression schemes were used:

- **(Fake) global:** Here all mean vectors belong to a single global regression class (the "fake global" node in the regression tree). The minimum observation count was set to 200.

- **Big Clusters:** Here the "fake global", "silence", "mostly voiced" and "mostly unvoiced" nodes were given very high minimum observation counts, so that regression matrices for these classes would never be estimated or employed. The "global" node was retained with a minimum observation count of 1400. The nodes "vowels", "semi-vowels and glides", "nasals", "affricates", "stops" and "fricatives" were retained, each with a minimum observation count of 25 times the number of phonemes in the regression class. It was later found that the node "semi-vowels and

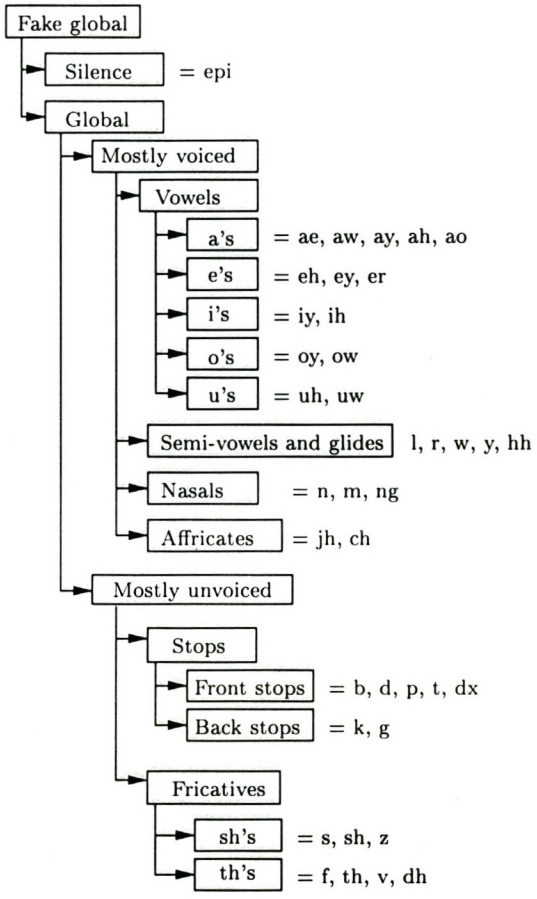

Figure 9.1: Regression tree used for MLLR on the TIMIT corpus.

glides" adapted poorly, and therefore the minimum observation class for this node was made very high, so that it was adapted using the "global" regression matrix.

- **Small clusters 1:** Here all the nodes except leaf nodes were given very high minimum observation counts. After initial experiments it was found that adaptation with nodes "stops", "fricatives" and "nasals" were poor, and these nodes were also given very high minimum observation counts. The remaining nodes were given minimum observation counts of 25 times the number of phonemes in the regression class.

- **Small clusters 2:** The same as the regression scheme "small clusters 1", except that the node "semi-vowels and glides" was given a very high minimum observation count.

- **Small clusters 3:** The same as the regression class scheme "small clusters 2", except that minimum observation count for each remaining node after removing undesirable nodes (by setting their minimum observation counts very high) was set to 144 times the number of phonemes in the regression class.

- **Tree 1:** All nodes are given a minimum observation count of between 25 and 50 times the number of phonemes in the regression class, except the nodes "fake global", "silence", "stops", "fricatives", "nasals" and "semi-vowels and glides" which are removed.

- **Tree 2:** All the nodes remaining in scheme "tree 1" are used, except that their minimum observation count is set to 500. Additionally, node "a's" is removed.

- **Tree 3:** All the nodes remaining in scheme "tree 2" are used, except that the "global" node is removed. All phonemes belonging to the "mostly unvoiced" node will thus not be adapted, and neither will their data be used in the estimation of a regression matrix with which phonemes belonging to the "mostly voiced" node might be adapted. This is to investigate whether unvoiced sounds should be adapted, and whether or not unvoiced data have a positive effect on MLLR adaptation.

It should be noted that for the SG systems, there are only three Gaussian pdfs for each phoneme, therefore the largest leaf node (nodes "a's" and "front stops") can at most have

fifteen mean vectors assigned to them. This number is too small for matrix $G^{(i)}$ to have an inverse, and therefore no regression class can be estimated (see the last paragraph of Section 4.3.2).

9.5 Experiment One: Performance of Adaptation Methods for Different Training and Test Sets

The aim of this experiment was to check the performance of three MLED implementations — standard MLED, mean-preserving covariance-based MLED and mean-preserving CBKLT-based MLED — on the TIMIT speech corpus, using an SG system and the transcription for the reduced set of phonemes proposed by Lee [32].

The experiment was completed in the following steps:

1. An SI model was created using all available data from the set of training speakers in the TIMIT corpus.

2. An iterative combination of MLLR and MAP adaptation was used to create a set of SD models for the set of training speakers in the corpus.

3. ML, MLLR and MLED adaptation were performed for the set of test speakers in the TIMIT corpus. Adaptation for five different training and testing data sets were completed and the results compared. The segmentation of the available data into training and testing data for the five sub-experiments were as follows:

   (a) In the first sub-experiment, the training data consisted of the first sa sentence, and the testing data consisted of the remaining sa, si and sx sentences.

   (b) In the second sub-experiment, the training data consisted of one of the si sentences, and the testing data consisted of the remaining sa, si and sx sentences.

   (c) In the third sub-experiment, the training data consisted of both sa sentences, and the testing data consisted of the remaining si and sx sentences.

   (d) In the fourth sub-experiment, the training data consisted of all three si sentences, and the test data consisted of the remaining sa and sx utterances.

(e) In the fifth sub-experiment, the training data consisted of all five sx utterances, and the testing data consisted of the remaining sa and si sentences.

The performance of the SI model for each training and testing data segmentation is also given. In every case but ML adaptation, only one adaptation iteration was employed.

9.5.1 Creating the SI and SD Initial Models

First, an SI model and set of SD models were created that are necessary for MLED adaptation. The SI model was trained using all utterances from the set of 462 speakers in the training set of TIMIT. Using MLLR adaptation and the SI model as an initial model, a set of 462 SD models was created. All the available data for each speaker were used during the training of the individual's SD model, in the hope of obtaining the most accurate speaker model possible. The more accurate the models for the training speakers, the more likely we are to obtain a representative speaker-space from which the eigenspace can be determined. The same data were used to score the individual's model. The scores for SD models created using various adaptation methods are contained in Tables 9.1 and 9.2.

Note that these results were obtained by testing the models using the same data set that was used for training the models. As the scores are not obtained from a test data set that is independent from the training data set, they should not be seen as a good indication of the true performance of the adaptation methods, nor of the true recognition performance of the resulting trained set of SD models. Accurate scores can only be obtained by using a large amount of test data which is independent of the training data. The only truth that can be learned by comparing the post-adaptation scores of the SD models to the pre-adaptation scores of the SI model for the case of using the same data set for training and testing, is that the lower error rates of the SD models indicate that the adaptation methods adapt model parameters is such a way that they maximise the likelihood of the data given the model parameters.

At best, when the training data set is very large, such scoring gives an indication of the best possible adaptation performance. At worst, when the training data set is very small (as in this case), such scoring could easily lead to error rates much lower than

	ER
SI	41.97%
ML, 1 iteration	21.86%
ML, 10 iterations	20.74%
MLLR, global	33.54%
MLLR, big clusters	55.76%
MLLR, small clusters 1	61.39%
MLLR, small clusters 2	51.77%
MLLR, small clusters 3	34.94%
MLLR, tree 1	34.98%
MLLR, tree 2	27.49%
MLLR, tree 3	30.17%
ML on MLLR (tree 2)	18.25%

Table 9.1: Error rates for initial SI and MLLR trained SD models (for MLED adaptation). All available data were used, both for training and scoring.

they would have been, had an independent test data set been available. This is because overtraining could have occurred, i.e. specialising the models too much for the small amount of available training data. Scoring the highly specialised models with the small, non-representative test (and training) data set will then yield seemingly good, but highly unreliable error rates.

The scores obtained for SD models created via ML, MLLR global and MLLR tree adaptation looked very promising, considering that only an SG system was used. Unfortunately, it was not possible to determine whether overtraining had occurred, as all available data were used to train the models. If overtraining had occurred, the error rates are probably not representative at all, and, in truth, very poor SD models may have been trained. Such poor training would result in a non-representative speaker-space, so that the extracted eigenspace will not model inter-speaker variance accurately. A poorly determined eigenspace will degrade the performance of MLED adaptation.

The SD models with the lowest error rates were obtained via iteratively alternating

	Error rate for iteration				
	1	2	3	4	5
MLLR, tree 2	27.49%	17.80%	17.34%	17.10%	16.90%
ML	18.25%	17.62%	17.27%	17.07%	16.98%

Table 9.2: Error rates for initial SD models obtained via alternating between MLLR and ML adaptation. The SI error rate was 41.97%.

between MLLR tree and ML adaptation. Five iterations were used (see Table 9.2). These models were thus used to create the set of SD supervectors from which the eigenvoices were determined.

9.5.2 Discussion of the Results

Table 9.3 contains the results for the five sub-experiments. From the results, it is clear that there was too little data for robust ML estimation. The only instance where ML estimation reduced the error rate of the SI system was for the fifth sub-experiment, where five adaptation sentences were used. MLLR adaptation had even poorer results. For the first three sub-experiments, there were too few observations to estimate a global regression matrix. For the fourth and fifth experiments, MLLR adaptation did take place, but it had an adverse effect on performance. The degradation for the fourth and fifth sub-experiments are a 5.25% and 2.97% increase in phoneme error rate, indicating that the increase in adaptation data from three to five adaptation sentences was making the MLLR estimate more robust. For the speech corpora and transcriptions Leggetter [34, 33] used for his tests of the MLLR adaptation method, he found that approximately ten seconds (corresponding to roughly 10000 frames of adaptation data) of speech was required for each regression class for the robust estimation of transformation matrices. In total there is five hours and 22 minutes of speech on the TIMIT corpus, which means there is about 30 seconds of speech available for each of the 630 speakers, and for each speaker parameters for a set of 39 HMMs (one per phoneme) must be reestimated. The sa1 sentence is 3.4 seconds long (averaged over the speakers), and the sum of the duration of the sx sentences is 14.6 seconds (averaged over the speakers. As only 3.4 to 14.6 seconds of

Adapted model	Error rates for sub-experiment				
	1	2	3	4	5
SI	44.93%	42.69%	45.52%	42.20%	42.00%
ML, 10 iterations	46.69%	44.28%	48.33%	45.03%	41.52%
MLLR, tree 2	44.93%	42.69%	45.72%	47.46%	44.97%
MLEDcor, $K = 1$	45.08%	42.27%	45.78%	42.10%	42.10%
MLEDcor, $K = 7$	41.65%	39.20%	41.94%	38.43%	38.31%
MLEDcor, $K = 10$	41.59%	39.13%	41.81%	38.08%	37.97%
MLEDcor, $K = 20$	42.67%	41.25%	42.21%	38.51%	38.03%
MLEDcov, $K = 1$	41.94%	39.55%	42.49%	38.97%	38.95%
MLEDcov, $K = 7$	41.68%	39.13%	41.87%	38.37%	38.03%
MLEDcov, $K = 10$	41.65%	39.44%	41.90%	38.15%	37.87%
MLEDcov, $K = 20$	42.75%	41.26%	42.12%	38.53%	37.87%
MLEDcbklt, $K = 1$	44.73%	42.10%	45.23%	41.57%	41.56%
MLEDcbklt, $K = 7$	44.22%	42.05%	44.76%	41.13%	40.31%
Sub-experiment 1: Train on first sa sentence, test on remainder					
Sub-experiment 2: Train on one si sentence, test on remainder					
Sub-experiment 3: Train on both sa sentences, test on remainder					
Sub-experiment 4: Train on all three si sentences, test on remainder					
Sub-experiment 5: Train on all five sx sentences, test on remainder					

Table 9.3: Adaptation results for the five sub-experiments of experiment one, where the performance of each adaptation method is evaluated for different training and test data sets.

speech per speaker was available for for MLLR adaptation on the TIMIT corpus, and the low occupancy of regression classes due to the SG models used, it is not surprising that MLLR adaptation performed so poorly.

MLED adaptation was the only adaption method to yield decreases in the error rate. As before, MLEDcov had better results than MLEDcor when only one eigenvoice was used. In fact, MLEDcor using one eigenvoice sometimes had poorer (higher) error rates than the SI model, whereas MLEDcov never performed worse than the SI model, regardless of the number of eigenvoices used. This result emphasises that the inclusion of the mean of the SD models adds the modelling power of one extra robustly estimated eigenvoice, for which no eigenweight needs to be determined. MLEDcov will thus be useful for situations where eigenweights for very few eigenvoices may be robustly estimated.

The same behaviour regarding the effect of eigenspace dimension was observed as for the ISOLET experiments: the number of eigenweights that can be robustly estimated decreases as the amount of adaptation data decreases. For the fifth sub-experiment, where five adaptation utterances where used, the error rate decreased up to about the ninth eigenvoice, after which the error rate effectively remained constant. For the first sub-experiment, where one adaptation utterance was used, the error rate decreased up to about the fifth eigenvoice, remained constant up to about the eleventh eigenvoice, after which the error rate began to climb.

MLEDcbklt generally yielded small decreases in the error rate, but always performed poorer than MLEDcov, and generally performed poorer than MLEDcor. The only times MLEDcbklt had better performance than MLEDcor, was for the cases where a single eigenvoice is used. This is because MLEDcbklt has the same mean-preserving capability as MLEDcov. The results indicate that there was very little benefit to be gained from eigenvoices based on the inter-class variance from speaker dialect classes for the current set of phonemes. Furthermore, the results are indicative that one of the underlying assumptions of standard MLED — that eigenvoices in the direction of greatest inter-speaker variance carry the most useful or important information for speaker adaptation — is better than the underlying assumptions of MLEDcbklt — that base vectors ("eigenvoices") in the direction of greatest inter-dialect variance carry the most useful information for speaker adaptation. The dialect class a speaker belongs to is thus less important than the

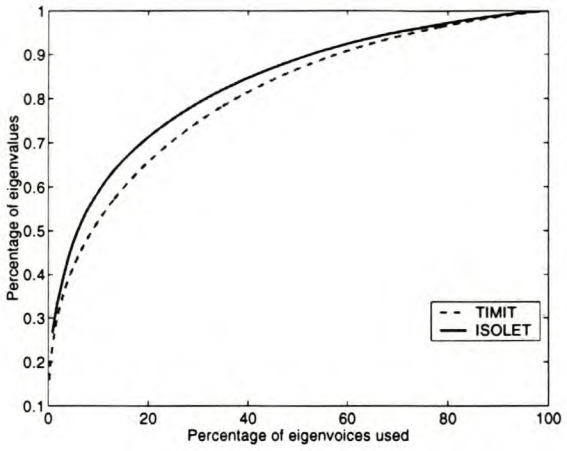

Figure 9.2: The percentage of the sum of eigenvalues versus the percentage of the eigenvoices used for the ISOLET and TIMIT corpora. 119 eigenvoices are available for the ISOLET corpus, and 461 eigenvoices are available for the TIMIT corpus.

location of the speakers supervector in the speakerspace.

The error rates obtained for the MLLR adapted models of the set of training speakers in TIMIT were vastly superior to the error rates achieved for adaptation on the set of test speakers in TIMIT. This indicates that overtraining had probably occurred during the creation of the set of prior SD models for MLLR, and that the low error rates for these models are wildly inaccurate and misleading. MLED is expected to have performance of the order of the SD models spanning the eigenspace, indicating that the error rates for the prior SD models would probably have been on the order of 39% instead of 17%. The test results for the prior SD models were thus not a reliable indication of their recognition performance. As previously stated, the poor performance of MLLR adaptation for both the set of training and test speakers is not only due to a lack of sufficient adaptation data, but also because of the simplicity of the SG system. Using an SG system means that there are fewer different mean vectors that are assigned to a regression class, making the chance of numerical instability and the inability to estimate regression matrices greater. The performance of MLED adaptation performance on the ISOLET corpus was superior to the performance of MLED adaptation on the TIMIT corpus. The first reason for this is the overtrained initial set of SD models used for the TIMIT corpus. The second reason has to do with the amount of inter-speaker variance captured by the eigenvoices used

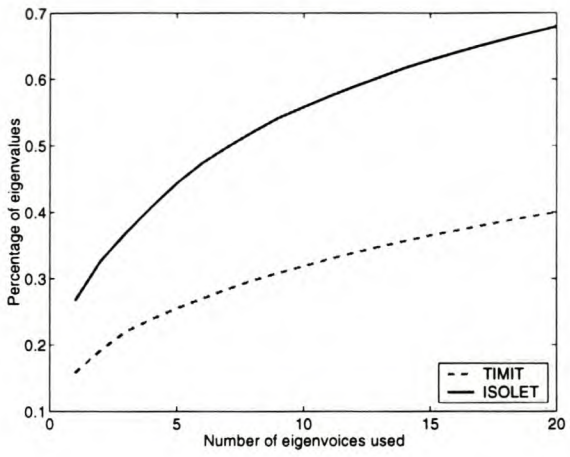

Figure 9.3: The percentage of the sum of eigenvalues versus the number of the eigenvoices used for the ISOLET and TIMIT corpora.

during the adaptation. The eigenvalue associated with an eigenvoice is related to the amount of inter-speaker variance in the direction of the eigenvoice. Figures 9.2 and 9.3 are plots of the sum of a number of eigenvalues relative to the total sum of eigenvalues for the eigenvoices used in ISOLET and TIMIT SG PLP systems. These eigenvoices were determined from the covariance matrix of the set of SD models for each corpus. 120 speakers' models were trained for ISOLET, therefore 119 eigenvoices could be extracted. 462 speaker models were trained for TIMIT, therefore 461 eigenvoices could be extracted. From these figures, it is clear that a much larger amount of inter-speaker variance is captured by the first few eigenvoices of ISOLET. There are three reasons for this:

1. TIMIT is more complex than ISOLET, because it contains more phonetic diversity and context than ISOLET. More phonemes were used to represent the speech on TIMIT.

2. The MLLR-trained prior set of SD models for the TIMIT corpus were overtrained.

3. Very little data were available for the training of the prior TIMIT SD models, resulting in little specialisation of the SD models. The smaller degree of specialisation has the following effects on the inter-speaker variance represented by the set of SD models: fewer directions of inter-speaker variance may be extracted by the KLT, and each of these directions are associated with smaller variances than those for ISOLET.

It is therefore not surprising that MLED adaptation was more successful for the experiments on the ISOLET corpus.

Even though the MLLR estimated prior SD models were poor, the differences between them still captured a good deal of inter-speaker variance, and MLED still managed modest improvement in error rates for the adapted speakers. This speaks volumes for the adaptation capabilities of the method. It seems to indicate that the inter-speaker variance represented by the set of prior SD models has a greater effect on MLED performance than the recognition performance of the SD models. This echos Kuhn *et al.* [29], who found that an eigenspace determined from a number of speakers with a set amount of data gave inferior performance to an eigenspace determined from double the number of speakers with half of the amount of data.

9.6 Experiment Two: Testing MLED on a Dialect-Independent Transcription

Experiment One indicated that most of the inter-dialect variance was compensated for by the transcription of the phonemes, and that there was very little benefit in using an MLEDcbklt implementation. This experiment is designed to examine MLEDcbklt for the case where a set of phonemes representing different pronunciations of the same utterance for speakers from the different dialect regions, are transcribed as the same symbol (this "phoneme" thus comprises of several true phonemes). The experiment was conducted as follows:

1. A dialect-independent transcription was created for each of the dialect emphasising sentences sa1 and sa2.

2. An initial SI model was created from the data of both sa sentences for the set of training speakers in TIMIT.

3. A set of SD models was created by iteratively alternating between MLLR and ML adaptation, using the SI model as the initial model and the data of both sa sentences for the set of training speakers in TIMIT. This set of SD models was then used to create the different sets of eigenvoices for the three MLED implementations.

4. MLLR and MLED adaptation were then used to create SD models for the set of testing speakers in TIMIT. Two training and testing data sets were used: in the first, the training data consist of sa1 and the testing data of sa2, and in the second the training data consist of sa2 and the testing data of sa1.

The various stages in the experiment will now be treated, after which the results will be discussed.

9.6.1 Creating a Dialect-Independent Set of Transcriptions

The creation of a set of dialect-independent transcriptions entails the selection of a reference transcription for each sa utterance, and then finding an optimal fit for the sa transcriptions of each speaker (target transcriptions) to the reference transcriptions. In this way, the phonemes representing the same speech segment are transcribed by a single symbol. The task is not straightforward, as the same number of phonemes might not be present in the reference and target transcription. A speaker may have omitted phonemes present in the reference transcription, or may have added phonemes that are not present in the reference transcription.

It was felt that the Viterbi-algorithm could be employed to solve the problem, as the problem entails the determination of an optimal match between two sequences of events (phonemes). In order to employ the Viterbi-algorithm, a discrete-density HMM was created to represent the reference transcription. Each of the phonemes (the symbols of our new dialect-independent transcription set) are now represented by a state. Each state pdf (strictly a pmf) is such that if a phoneme belongs to the same class as the state's reference phoneme, the probability is 0.99; if not, the probability is $\{(1 - 0.99)/A\}$, where A is the number of phonetic classes.

To accommodate omissions and additions of phonemes in target transcriptions, a double-skipwidth left-to-right HMM was used. For each emitting state j, the transition probabilities were as follows: $a_{i,i} = 0.1$, $a_{i,i+1} = 0.5$, $a_{i,i+2} = 0.3$, and $a_{i,i+3} = 0.1$. The initial probabilities were such that the sequence must begin in the first state, i.e. $\pi_1 = 1$ and all other $\pi_i = 0$, $i \neq 1$.[1]

[1] In retrospect it may have been better to allow the sequence to begin in more than only the first state.

Seven phonetic classes were used, one for each of the following: stops; affricates; fricatives; nasal; semi-vowels and glides; vowels; and silence.

Using the HMM and the Viterbi-algorithm, the optimal match (given the choice of HMM parameters) between the reference transcription and target transcription could be determined. The new transcription for the target transcription was then created by going through the optimal state sequence, and determining the probabilities of the target phoneme for the state it was assigned to. If the probability is equal to 0.99, the new symbol for the target phoneme transcription is the reference phoneme transcription. If the probability is not equal to 0.99 (i.e. the target phoneme does not belong to the same phonetic class as the reference phoneme), the new transcription for the target phoneme is "ARB". Speech labelled by "ARB" is not used for training or testing.

The transcriptions of speaker "fadg0" from the test set of dialect region one were (randomly) selected to form reference transcriptions for the sa sentences. Below we list the reference transcription for sa1, the initial target transcription of sa1 for speaker "mcmj0" (from the test set of dialect region six) and the final dialect-independent transcription of sentence sa1 for speaker "mcmj0".

- **Reference:** epi, sh, iy, hh, ae, d, y, er, d, ao, r, k, s, uw, t, ih, n, g r, iy, s, iy, w, ao, sh, epi, w, ao, dx, er, t, ao, l, y, ih, er, epi
- **Target:** epi, sh, iy, hh, eh, epi, y, er, d, ao, r, k, s, uw, t, ih, n, g, r, iy, s, iy, w, ao, sh, epi, w, ao, dx, er, t, ao, l, y, iy, er, epi
- **New:** epi, sh, iy, hh, ae, ARB, y, er, d, ao, r, k, s, uw, t, ih, n, g, r, iy, s, iy, w, ao, sh, epi, w, ao, dx, er, t, ao, l, y, ih, er, epi

Only 26 "phonemes" remain in the dialect-independent transcription, as opposed to the 39 in the original set. For a list of these phonemes, see Appendix E.2.

9.6.2 Creating the Initial SI and SD Models

The same HMM models' structures as those used in Experiment One were used for spoken phonemes and silence. The initial SG SI model was trained using all the data from the sa utterances of the set of training speakers in TIMIT. Subsequently, a set of SD models

Adapted model	ER
SI	13.39%
MLLR, big clusters	13.15%
MLLR, tree 2	13.13%

Table 9.4: Error rates of the SI model and MLLR adapted models.

Adapted model	ER for iteration				
	1	2	3	4	5
MLLR, tree 2	13.13%	8.42%	8.42%	8.41%	8.41%
ML	8.43%	8.42%	8.43%	8.42%	8.42%

Table 9.5: Iterative ML and MLLR adaptation to improve SD models for the dialect experiment. The SI error rate was 13.93%.

were trained for the set of training speakers using all the data from the sa utterances. All the available training data were used in order to estimate the SD models as robustly as possible. Tables 9.4 and 9.5 contain the error rates for the models. The models were scored using the same data on which they were trained. As the amount of adaptation data is very small (about 6.2 seconds of speech for each speaker) and the MLLR regression class occupancy of SG models are very low, overtraining had probably occurred and the error rates could be misleading.

9.6.3 Post-Adaptation Error Rates for Experiment Two on TIMIT

Tables 9.6 and 9.7 contain the results for MLED adaptation. The error rates are much smaller than those for Experiment One, but that is simply because the number of phonemes that are modelled (and need to be recognised) have been reduced from 39 to 26. MLEDcor and MLEDcov each only managed to reduce the error rate by about 1%, which is less than what is expected of adaptation of such simple models with the available adaptation data. The MLEDcbklt implementation could not even manage a 1% reduction in error rate. As with with Experiment One, the MLEDcbklt implementation is once more shown to be slightly worse than standard MLED (MLEDcor), even though the dialect-independent

Adapted model	ER						
	$K = 1$	$K = 2$	$K = 3$	$K = 4$	$K = 5$	$K = 6$	$K = 7$
MLEDcor	11.60%	11.22%	11.35%	11.25%	11.25%	11.25%	11.07%
MLEDcov	11.21%	11.38%	10.37%	11.27%	11.28%	11.20%	11.06%
MLEDcbklt	11.40%	11.40%	11.46%	11.40%	11.48%	11.44%	11.36%

Table 9.6: Error rates after MLED adaptation for Experiment Two on TIMIT. sa1 was used for training, and sa2 was used for testing. The SI error rate was 11.67%.

Adapted model	ER						
	$K = 1$	$K = 2$	$K = 3$	$K = 4$	$K = 5$	$K = 6$	$K = 7$
MLEDcor	10.58%	9.55%	9.63%	9.62%	9.81%	9.88%	9.87%
MLEDcov	9.73%	9.71%	9.66%	9.69%	9.65%	9.71%	9.73%
MLEDcbklt	10.41%	10.04%	10.02%	9.81%	10.05%	9.94%	10.10%

Table 9.7: Error rates after MLED adaptation for Experiment Two on TIMIT. sa2 was used for training, and sa1 was used for testing. The SI error rate was 10.76%.

transcription was used. The poorly trained set of SD models from which the eigenvoices were determined is the most likely cause of the relatively poor performance of MLED.

9.7 Experiment Three: A Further Dialect Experiment Using a Larger Amount of Training Data

The biggest problem with the previous section was the poor estimation of the set of prior SD models, resulting in a non-representative speakerspace (and eigenspace) and poor adaptation performance. Furthermore, a relatively small amount of data was available with which to perform MLED adaptation. The aim of Experiment Three was to reduce the problems associated with Experiment Two, so that a better analysis of MLEDcbklt for dialect classes could be made.

First, a better prior set of SD models needed to be trained. These models are trained using alternating iterative MLLR and ML adaptation. There are two ways in which the MLLR portion of the training process can be improved. The first is by increasing the amount of adaptation data, which is not possible in this case. The second is to increase the amount of mean vectors assigned to a regression class. As even a global regression class for the SG system does not have a high mean vector occupancy, it is likely to be poorly estimated due to numerical instability. Therefore, it was decided to change to a GMM system, as this will result in an eightfold increase in the number of mean vectors assigned to each regression class, making the regression matrix estimation less prone to numerical instability.

Secondly, the quantity of adaptation data used for MLED adaptation was increased by using round robin training and scoring on the two sa sentences. All the phonemes of both sentences save one are used for training, and scoring is then done using the remaining phoneme. The round robin training is done until scoring has been done with each phoneme. The scores are then tallied, and the error rate computed.

Table 9.8 contains the results after MLED adaptation. The decreases in error rate for all MLED implementations are modest — on the order of 2%, which is a relative decrease of about 30% in the error rate of the SI model. The decrease in error rate as the eigendimension is increased is very small. This is due to the rather poorly MLLR

adapted speaker models from which the eigenvoices were created. Poor adaptation results in smaller differentiation between speakers, so that fewer directions of inter-speaker variance can be robustly determined. When we compare this experiment with Experiment Two, we see that the amount of adaptation data increased twofold and the number of model parameters to reestimate increased eightfold. This means that a greater degree of parameter spreading is necessary for speaker adaptation in this experiment. Nonetheless, the relative decrease in error rate for Experiment Three (around 30%) was much greater than the relative decrease in error rate for Experiment Two (which was around 10%). This is possibly due to the effect of adapting GMM systems with MLLR adaptation, as the occupancy of regression matrices is greater for GMM systems than for SG systems. Normally it would be expected that the greater the parameter spreading necessary for MLLR adaptation, the poorer the results would be. However, the GMM systems are adapted better here, as the regression class occupancy for SG systems is simply too low, and is likely to result in numerical instability regardless of the amount of adaptation data. In this instance, therefore, the GMM systems allow better MLLR adaptation, so that the eigenspace determined from the MLLR estimated set of prior SD models is more representative of the true speakerspace than the eigenspace determined for Experiment Two.

MLEDcbklt performance was slightly worse than that MLEDcor and MLEDcov. Once more it is found that eigenvoices based on inter-speaker variance outperform eigenvoices based on inter-dialect variance. To further evaluate the performance of the mean-preserving CBKLT-based MLED implementation, a speech corpus containing more speech per speaker — maybe twenty to thirty sentences (each with an average duration of 3 seconds per sentence) per speaker — is required. In addition to this, the speakers in the corpus must come from distinct speaker classes.

9.8 Discussion of TIMIT Experiments

This section lists and discusses the most important results of the experiments on the TIMIT speech corpus.

Adapted model	ER						
	$K = 1$	$K = 2$	$K = 3$	$K = 4$	$K = 5$	$K = 6$	$K = 7$
MLEDcor	3.87%	3.69%	3.68%	3.62%	3.56%	3.46%	3.46%
MLEDcov	3.71%	3.66%	3.71%	3.63%	3.51%	3.38%	3.53%
MLEDcbklt	3.90%	3.88%	3.83%	3.80%	3.70%	3.64%	3.64%

Table 9.8: Error rates after MLED adaptation for Experiment Three on TIMIT. The SI error rate was 5.89%.

- From the experiments it is seen that the performance of MLED is linked to the quality of the prior SD models. This dependency is not so much on the recognition performance of the SD models, but rather on how well the true inter-speaker variance is represented by the set of SD models.

- The mean-preserving covariance-based MLED implementation (MLEDcov) always performs better than standard MLED (MLEDcor) for the case where a single eigenvoice is used.

- The results of Experiment One — where MLED adaptation of an SG system for various training and test data sets was examined — indicate that the number of eigenvoices used should depend on the amount of available adaptation data. If too many eigenvoices are used, the degree of parameter spreading becomes too great and results in non-robust adaptation. The amount of eigenvoices that can effectively be employed thus depends on the following:

  - The true inter-speaker variance, which is dependent on such factors as the language, phonemes and speech of the speakers that need to be modelled.
  - The set of prior SD models. The better the SD models are trained and the more speakers are represented, the higher the dimension of the eigenspace that is an accurate representation of inter-speaker variance.
  - The amount of available adaptation data relative to the number of parameters to be estimated. The larger the amount of adaptation data, the more eigenweights may be robustly estimated.

As an efficient eigenspace dimension to use is dependent on so many variable factors, it is hard to suggest empirical guidelines.

- During the ISOLET experiments, the MLEDcbklt implementation was used to adapt for inter-gender variance, and was shown to have poorer performance than standard MLED. The TIMIT experiments tested the use MLEDcbklt on dialect classes.

  In Experiment One, where a dialect-dependent transcription was used, MLEDcbklt has poorer performance than MLEDcor (standard MLED) and MLEDcov. Using the dialect-independent transcription (in Experiment Two and Experiment Three) benefited MLEDcbklt performance. In both Experiment Two and Experiment Three, MLEDcbklt performance was much closer — though still slightly worse — to the performance of MLEDcor and MLEDcor.

  From the results we can deduce that eigenvoices based on inter-speaker variance are superior to eigenvoices based on inter-dialect variance. Perhaps different classes would lead to a MLEDcbklt implementation with better performance, but for gender-based and dialect-based classes the performance was worse than that of standard MLED (apart from the instances where mean-preservation boosted MLEDcbklt performance past that of standard MLED).

  It should be emphasised that the only advantage of using the CBKLT would be to obtain a few eigenvoices that might have a better effect on adaptation than eigenvoices based on inter-speaker variance. When many eigenvoices are used, the KLT obtained eigenvoices (based on inter-speaker variance) will capture the inter-class variance — as it is part of the inter-speaker variance — making the use of the CBKLT (LDA) pointless. The CBKLT can thus only shift the focus of the first few eigenvoices, but will not provide a better approximating subspace for the speakerspace than the KLT when many eigenvoices are used. The CBKLT (LDA) is very useful as a clustering or segmentation device, as it provides the greatest linear separation between classes in the underlying data along the extracted base vectors. If this separation is not useful for adaptation purposes, it will perform worse for MLED speaker adaptation than the KLT (PCA).

It should be noted that eigenvoices were recently successfully employed for speaker identification and verification [49]. For this application LDA is shown perform better than PCA for certain speech corpora.

Chapter 10

Conclusions and Recommendations

This chapter covers the achievements and conclusions of this thesis, as well as suggestions for future research on speaker adaptation. First, the major conclusions are discussed.

10.1 Conclusions

- The best adaptation method for a problem depends on the expected amount of available adaptation data. The performance of a method for varying amounts of adaptation data is strongly linked to the parameter reduction capabilities of the method. The higher the parameter reduction, the more robust the method will be for sparse data conditions. Lower parameter reduction makes more specialised adaptation of parameters possible, which will lead to lower error rates provided that the estimates were robust.

  If large amounts of adaptation data is available for most of the parameters in the system, MAP adaptation will be a good choice. For moderate amounts of adaptation, an MLLR scheme would be a good choice. For very small amounts of adaptation, MLED adaptation would be a good choice.

  These findings are in accord with the findings of other researchers [51, 40, 33].

- The complexity of the model impacts on the performance of adaptation methods. In general, lower model complexity estimation allows more robust estimation of parameters and higher model complexity allows more specific adaptation of parameters.

MLLR displayed more complex behaviour for the model order. The smaller the number of seen mean vectors in the regression class, the less robust the estimate is likely to be, and the more likely it is to encounter numerical instability. The number of seen mean vectors depends on the amount and spread of adaptation data, as well as the number of mean vectors belonging to the regression class. For the SG models, the number of mean vectors belonging to a node in the regression tree was so low that virtually only the global node could be used for adaptation, regardless of the amount of available adaptation data. Even greater parameter reduction was required for MLLR adaptation of GMM systems. Even though the number of parameters for GMM systems was higher, the results were better than for SG systems as the regression class occupancy was higher. This behaviour was also remarked on by [34, 33].

- The mean-preserving covariance-based MLED implementation always had better adaptation than standard MLED when a single eigenvoice was used. It is unfortunate, as it was hoped that the explicit inclusion of the mean of the prior set of SD models would allow an eigenspace that is a more accurate approximation of the true speaker-space, and that this would lead to lower error rates for at least the first few eigenvoices. However, it seems that the inter-speaker variance in the speakerspace is such that the eigenvoices extracted by the KLT on the correlation matrix of the SD supervectors have much the same effect as eigenvoices extracted by the KLT on the covariance matrix of the SD supervectors.

  The mean-preserving CBKLT-based MLED implementation was also shown to have better performance than standard MLED when a single eigenvoice was used. This indicates that mean-preservation is a good method of boosting MLED performance when a single eigenvoice is used.

- The performance of the CBKLT MLED implementation for gender classes in ISOLET indicated that a great deal of inter-speaker variance is a result of gender. This finds correlation with Nguyen's work, who found that PCA identifies the greatest source of inter-speaker variance as the different sexes of the speakers [40].

For the case of a reduced-dialect independent phoneme set, the mean-preserving CBKLT-based MLED implementation was shown to have slightly poorer performance than standard MLED. If the inter-dialect variance was greater, CBKLT-based MLED might have had better performance, but such a test must still be completed. We thus implemented CBKLT (LDA) as a method of forming eigenvoices (as suggested by [28, 40, 51]), and found that it had poorer results than standard MLED using the KLT (PCA) for gender-based or dialect-based speaker classes. Perhaps a different clustering method might yield better speaker classes, though it is unlikely to yield better performance than standard MLED unless it extracts better initial eigenvoices. As stated in Section 9.8: When many eigenvoices are used, the KLT obtained eigenvoices (based on inter-speaker variance) will capture the inter-class variance — as it is part of the inter-speaker variance — making the use of the CBKLT (LDA) pointless.

- The performance of MLED adaptation is highly dependent on the size and quality of the set of SD models from which the eigenspace was extracted. This seems to be the greatest limiting factor in MLED performance, even more so than the available amount of adaptation data.

- In general, using PLP-features is superior to LPCC-features for SI systems, but inferior for SD models and adaptation of SI models. It is our opinion that this is a direct consequence of reduced inter-speaker variance when using PLP-features. This conclusion is of some significance, as many researchers employ PLP-cepstral (or MFCC) features for speaker adaptation purposes, simply because PLP-cepstral features have better performance than LPCC features for SI systems.

10.2 Achievements

- Speaker adaptation was incorporated in the in-house DSP and pattern recognition toolkit, PatrecII. This forms a base that will facilitate further speaker and environmental adaptation experiments driven by PatrecII.

- Two new MLED implementations were designed and implemented. The first novel implementation — preservation of the mean of the prior set of SD models for MLED — resulted in better adaptation than standard MLED when a single eigenvoice was used. Mean-preservation has the effect of adding an extra robustly estimated eigenvoice to MLED adaptation.

- The second novel MLED extension was the use of the CBKLT (LDA) to determine the eigenvoices used for adaptation. It was shown to have performance that was slightly worse than that of standard MLED. The use of CBKLT-extracted (LDA-extracted) eigenvoices was suggested by other researchers [28, 40, 51] as an alternative to KLT-extracted eigenvoices. The results in this thesis indicate that it is unlikely for CBKLT-extracted eigenvoices to outperform KLT-extracted eigenvoices for the purposes of speech recognition.[1]

- LPCC features were shown to be preferable to PLP-cepstral features for most adaptation implementations and models of varying complexity, notwithstanding the fact that PLP-cepstral features generally outperform LPCC features for SI models. This is something to be aware of when designing a recognition system. If a SI system is to be used, then PLP-cepstral features will be better. However, should the recognition system employ SD models and the use of speaker adaptation methods, LPCC features might very well be a better choice.

10.3 Recommendations

The use of eigenvoice decomposition for speaker adaptation is still relatively new, and research in the following has not been done to date:

- MLED (or another eigenvoice decomposition technique) should be extended to adapt other HMM parameters besides the mean vectors. It is our opinion that post-adaptation recognition rates may be improved for the case where moderate amounts of data are available if the covariance matrices (and possibly the transition probabilities) could be incorporated into the same MLED framework. Other researchers

[1] Recent work in the field of speaker identification and verification using eigenvoices has shown LDA to outperform PCA for certain speech corpora. Here a class consisted of all the speech for a speaker.

have also commented on this, but to date MLED has not been extended to include the adaptation of anything but mean vectors.

- The number of eigenvoices needed to effectively cover the inter-speaker variability is generally quite high. For instance, Westwood found that around twenty eigenvoices are needed to account for half the inter-speaker variability for a set of 109 speakers [51]. Non-linear methods should be analysed, to see whether the number of bases needed to accurately model inter-speaker variability can be reduced. The bases thus derived might also prove to result in more robust adaptation given the same amount of adaptation data as normal MLED bases, as they will match the inter-speaker variance more closely.

- Further research may be done to determine the usefulness of the CBKLT-based MLED implementation, though it is our opinion that it is unlikely that such an implementation would outperform standard MLED. A speech corpus with greater dialect or accent variance than TIMIT should be used so that a better comparison of the CBKLT-based MLED implementation and standard MLED may be made. Furthermore, different speaker classes may be used. A simple clustering technique could be used to form speaker classes, after which CBKLT may be used to extract eigenvoices.

- As remarked by Westwood [51], no research has to date (to our knowledge) been done on the effectivity of eigenvoice decomposition for environmental adaptation. Westwood further states that such a method should include examples for SI or SD models trained under various environmental conditions in the set of prior speech models, so that the extracted eigenvoices would model the directions of greatest inter-environmental variance or inter-speaker-and-environmental variance. Given the non-linear changes that noise and other environmental effects cause on feature vectors, it remains to be seen if the linearly extracted eigenvoices could cope with this task for large changes in noise and environment. Non-linearly extracted eigenvoices may have better results for a speaker-and-environment space that is expected to be highly non-linear.

Further research into the effect of different types of feature vectors on SI, SD, speaker and environmentally adapted models is also necessary. Such research would facilitate the optimisation of a recognition system in terms of model complexity and feature vector type for a certain speaker adaptation technique.

To date, many detailed comparisons have been drawn between adaptation methods. In most work on eigenvoice decomposition methods, including this thesis, the method is compared to a relatively basic MLLR implementation where only the mean vectors are adapted and a simple (typically global) regression scheme is employed. This is understandable, since eigenvoice decomposition techniques are typically tested for conditions of extreme to moderate data scarceness, where few MLLR regression matrices may be robustly estimated. A detailed comparison of MLED or WP with a high-end MLLR implementation must still be completed so that an accurate assessment may be made of which method is more effective for any given amount of adaptation data.

Bibliography

[1] S. M. Ahadi and P. C. Woodland, "Rapid Speaker Adaptation Using Model Prediction", *Proceedings ICASSP 95*, 1995.

[2] S. M. Ahadi and P. C. Woodland, "Rapid Speaker Adaptation Using Model Prediction", *Proc. IEEE Int. Conf. Acoust. Speech, Signal Processing*, Vol. 1, Detriot, May, 1995.

[3] S. M. Ahadi-Sarkani, "Bayesian and Predictive Techniques for Speaker Adaptation", PhD Dissertation, Cambridge University, 1996.

[4] L. E. Baum, "An Inequality and Associated Maximisation Technique in Statistical Estimation for Probabilistic Functions of Markov Processes", *Inequalities 3*, pp. 1-8, 1972.

[5] F. Bimbot and L. Mathan, "Text-Free Speaker Recognition using an Arithmetic Harmonic Shericity Measure", Proc. Eurospeech, Vol. 1, pp. 169-172, 1993.

[6] F. Bimbot, I. Magrin-Chagnolleau, and L. Mathan, "Second-Order Statistical Measures for Text-Independent Speaker Identification", *Speech Communication*, Vol. 17, No. 1-2, pp. 177-192, August 1995.

[7] D. R. Cox and D. V. Hinkley, "Theoretical Statistics", Chapman and Hall, 1986.

[8] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, pp. 357-366, August. 1980.

[9] J. R. Deller, Jr., J. H. L. Hansen, J. G. Proakis, "Discrete-Time Processing of Speech Signals", IEEE Press, 2000.

[10] A. P. Dempster, N. M. Laird and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society*, Ser. B, vol. 39, pp. 1-38, 1977.

[11] J. A. du Preez, Notes for the course "Pattern Recognition 813", sec. High Dimensional Feature Spaces and the Karhunen Loéve Transform, Stellenbosch University Engineering Department, DSP Group, 2001.

[12] A. Elfessi and D. M. Reyneke, "A Bayesian Look at Classical Estimation: The Exponential Distribution," *Journal of Statistics Education*, vol. 9, number 1, 2001, available as http://www.amstat.org/publications/jse/v9n1/elfessi.html.

[13] K. Fukunaga, "Introduction to Statistical Pattern Recognition," Academic Press, 1990.

[14] M. J. F. Gales, "Cluster Adaptive Training for Speech Recognition", *Proceedings ICSLP 98*, 1998.

[15] M. J. F. Gales, "Maximum Likelihood Linear Transformations for HMM-based Speech Recognition - TR 291", Technical Report, Cambridge University Engineering Department, May 1997.

[16] M. J. F. Gales, "Semi-tied Full-covariance Matrices for Hidden Markov Models - TR 287", Technical Report, Cambridge University Engineering Department, April 1997.

[17] M. J. F. Gales, "The Generation and Use of Regression Class Trees for MLLR Adaptation - TR263", Technical Report, Cambridge University Engineering Department, August 1996.

[18] J.-L. Gauvain and C.-H. Lee, "Maximum *a Posteriori* Estimation for Multivariate Gaussian Mixture Observations of Markov Chains", *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 2, pp. 291-298, April 1994.

[19] L. Gillick and S. J. Cox, "Some Statistical Issues in the Comparison of Speech Recognition Algorithms," *Proceedings ICASSP 1989*, vol. 1, pp. 532-535, Glasgow, UK, 1989.

[20] X. D. Huang and K. F. Lee, "On Speaker-Independent, Speaker-Dependent and Speaker Adaptive Speech Recognition", *Proc. ICASSP*, Vol. 2, pp. 877-880, 1991.

[21] M. Jamshidian and R. I. Jennrich, "Conjugate Gradient Acceleration of the EM Algorithm", *Journal of the American Statistical Association*, vol. 88, no. 412, pp. 221-228.

[22] J. Jaschul, "Speaker adaptation by a linear transformation with optimized parameters", *Proceedings of the ICASSP* 3, pp. 1657-1670, 1982.

[23] S. E. Johnson and P. C. Woodland, "Speaker Clustering using Direct Maximisation of the MLLR-adapted Likelihood", Cambridge University Engineering Department.

[24] B.-H. Juang, "Maximum Likelihood Estimation for Mixture Multivariate Stochastic Observations of Markov Chains", *AT&T Technical Journal*, 64, pp. 1235-1249, 1985.

[25] M. Kirby and L. Sirovich, "Application of the Karhunen-Loève Procedure for the Characterization of Human Faces", *IEEE PAMI*, V. 12, no. 1, pp. 103-108, Jan. 1990.

[26] L. Knohl and A. Rinscheid, "Speaker Normalization and Adaptation Based on Feature Map Selection", *Proc. Eurospeech*, Vol. 1, pp. 367-370, 1993.

[27] R. Kuhn, "Eigenvoices for Speaker Adaptation", Technical Report, Speech Technology Laboratory (STL), July 1997.

[28] R. Kuhn, P. Nguyen, J.-C. Junqua, L. Goldwasser, N. Niedzielski, S. Fincke, K. Field and M. Contolini, "Eigenvoices for Speaker Adaptation", *Proc. ICSLP 98*, 1998.

[29] R. Kuhn, P. Nguyen, J.-C. Junqua, N. Niedzielski, S. Fincke, K. Field and M. Contolini, "Fast Speaker Adaptation using *a Priori* Knowledge", *Proc. ICASSP 99*, 1999.

[30] R. Kuhn, F. Perronnin, P. Nguyen, J.-C. Junqua and L. Rigazio, "Very Fast Adaptation with a Compact Context-Dependent Eigenvoice Model", *Proc. ICASSP 2001*, 2001

[31] R. Kuhn, F. Perronnin, P. Nguyen, J,-C. Junqua and L. Rigazio, "Very Fast Adaptation with a Compact Context-Dependent Eigenvoice Model (Corrected Version)", Technical Report, Panasonic Speech Technology Laboratory, 2001.

[32] K. F. Lee, "Speaker-Independent Phone Recognition Using Hidden Markov Models", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 37, no. 11, pp. 1641-1648, November 1989.

[33] C. J. Leggetter, "Improved Acoustic Modelling for HMMs using Linear Transformations", PhD Thesis, Cambridge University, 1995.

[34] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models", *Computer Speech and Language*, 9, pp. 171-185, 1995.

[35] L. A. Liporace, "Maximum Likelihood Estimation for Multivariate Observations of Markov Sources", *IEEE Transactions on Information Theory IT-28*, pp. 729-734, 1982.

[36] N. Muller, L. Magaia and B. Herbst, "Singular Value Decomposition and Facial Recognition", Technical Report, Department of Applied Mathematics, University of Stellenbosch, February 8, 2001.

[37] Y. Muthusamy, Ronald A. Cole and Mark Fanty, "The ISOLET Spoken Letter Database", Tech. Report, Oregon Graduate Institute of Science and Technology (OGI), 19600 N.W. von Neumann Drive, Beaverton, OR 97006, November 1994, available as http://www.cse.ogi.edu/CSLU/copora/isolet.html.

[38] National Institute of Standards and Technology, DARPA Resource Management Continuous Speech Database, Documentation on CD-ROM "NIST Disc 2-3.1 and 2-4.2", 1992.

[39] National Institute of Standards and Technology, The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus, Documentation on CD-ROM "NIST Speech Disc CD1-1.1", December 1990.

[40] P. Nguyen, "Fast Speaker Adaptation", Industrial Thesis Report, Institut Eurécom, June 17, 1998.

[41] T. Niesler, Notes for the course "Digital Signal Processing 823", sec. Speech Analysis, Slides 3.1-3.11, Stellenbosch University Engineering Department, DSP Group, 2001.

[42] F. Nolan, "The Phonetic Bases of Speech Recognition", Cambridge University Press, 1983.

[43] J. J. Odell, "The Use of Context in Large Vocabulary Speech Recognition", DPhil Dissertation, University of Cambridge, March 1995.

[44] J. J. Odell, P. C. Woodland, and S. J. Young, "Tree-Based State Clustering for Large Vocabulary Speech Recognition", *Proc. International Symposium on Speech, Image Processing and Neural Networks*, Vol. 2, pp. 690-693, Hong Kong, April 1994.

[45] Y. Ono, H. Wakita and Y. Zhao, "Speaker Normalization using Constrained Spectra Shifts in Auditory Filter Domain", *Proc. Eurospeech*, Vol. 1, pp. 355-358, 1993.

[46] K. K. Paliwal and W. A. Ainsworth, "Dynamic Frequency Warping for Speaker Adaptation in Automatic Speech Recognition", *Journal of Phonetics*, (13):123-134, 1985.

[47] Rabiner and Juang, "Fundamentals of Speech Recognition", Prentice Hall, Englewood Cliffs, NJ, 1994.

[48] G. Strang, "Introduction to Linear Algebra", Wellesley-Cambridge Press, 1993.

[49] O. Thyes, R. Kuhn, P. Nguyen and J.-C. Junqua, "Speaker identification and Verification using Eigenvoices", Technical Report, Panasonic Speech Technology Laboratory, 2001.

[50] H. Wakita, "Normalisation of Vowels by Vocal Tract Length and its Application to Vowel Identification", *IEEE Trans. ASSP*, 25(2):183-192, April 1977.

[51] R. Westwood, "Speaker Adaptation Using Eigenvoices", MPhil Thesis, Cambridge University Engineering Department, August 1999.

[52] C. Wu, "On the Convergence Properties of the EM Algorithm", *The Annals of Statistics*, vol. 11, no. 1, pp. 95-103, 1983.

[53] S. J. Young, J. J. Odell, and P.C. Woodland, "Tree-Based State Tying for High Accuracy Acoustic Modelling", *Proc. ARPA Human Language Technology Conference*, Plainsboro, NJ, March 1994.

Appendix A

Information Pertaining to MAP Adaptation

A.1 Posterior Distributions and Conjugate Families of Prior Densities

Let Θ and Y be continuous random variables and let $f_\Theta(\theta)$ be the prior density of Θ and $g(y|\theta)$ be the conditional density of Y given Θ. Bayes' Theorem for continuous random variables then can be represented by

$$g(\theta|\, y) = \frac{f_\Theta(\theta)g(y|\,\theta)}{\int_{-\infty}^{\infty} f_\Theta(\theta)g(y|\,\theta)d\theta}\ , \tag{A.1}$$

where $g(\theta|\, y)$ is called the posterior density function of Θ.

In Bayesian statistics, conjugate priors are often used. Let $\mathcal{F}$ denote a class of density functions. A class $\mathcal{C}$ of prior distributions is a conjugate family for $\mathcal{F}$ if the posterior distribution is also in the class $\mathcal{C}$ for all density functions in $\mathcal{F}$ and all prior density functions in $\mathcal{C}$. When using conjugate priors, the posterior distribution is normally easily calculated.

A.2 Prior Densities Referred to During MAP Adaptation Discussions

According to Gauvain and Lee [18] a practical candidate to model the prior knowledge of the mixture gain parameter vector is the Dirichlet density, given by

$$g(c_1, \ldots, c_M | v_1, \ldots, v_M) = \prod_{m=1}^{M} c_m^{v_m - 1}, \tag{A.2}$$

where c_m are the mixture component weights, and $v_m > 0$ are the parameters for the Dirichlet density.

The Dirichlet density can also be used as prior probability for the initial probability vector π and each row of the transition probability matrix A of an HMM (see Section 2.4).

The prior probability used for each individual Gaussian pdf in a mixture pdf is a normal-Wishart density, given by

$$\begin{aligned} & g(\mu_m, C_m | \tau_m, q_m, \alpha_m, u_m) \\ &= |C^{-1}|^{(\alpha_m - n)/2} \exp\left[-\frac{\tau_m}{2}(\mu_m - q_m)' C_m^{-1}(\mu_m - q_m)\right] \exp\left[-\frac{1}{2} tr(u_m C_m^{-1})\right], \end{aligned} \tag{A.3}$$

where n is the dimension of the Gaussian pdf, μ_m is the mean vector, C is the covariance matrix, $(\tau_m, q_m, \alpha_m, u_m)$ are the parameters of the normal-Wishart pdf such that $\alpha_m > n - 1$, $\tau_m > 0$, q_m is a vector of dimension n, and u_m is a $p \times p$ positive definite matrix.

If independence between mixture weights and parameters of individual mixture pdf components is assumed, the prior density for a GMM is a joint pdf obtained from the product of A.2 and A.3:

$$g(\theta) = g(c_1, \ldots, c_M) \prod_{m=1}^{M} g(\mu_m, C_m). \tag{A.4}$$

From here, Gauvain and Lee go on to prove that the prior density for all HMM parameters can be modelled by

$$G(\lambda) = \prod_{i=1}^{N} \left[\pi_i^{\eta_i - 1} g(\theta_i) \prod_{j=1}^{N} a_{ij}^{\eta_{ij} - 1} \right], \tag{A.5}$$

where $\{\eta_i\}$ is the parameter set for the prior density of the initial probabilities $\{\pi_i\}$, and $\{\eta_{ij}\}$ is the parameter set for the prior density of the transition probabilities $\{a_{ij}\}$, and N is the number of states.

Appendix B

Supplementary Mathematical Derivations for MLLR

B.1 Differentiation of a Scalar Function with Respect to a Matrix

When the term "matrix differentiation" is used in this thesis, the definition introduced below will be followed.

The derivative of a scalar function $f(A)$ with respect to an $m \times n$ matrix A is defined as:

$$\frac{df}{dA} = \begin{bmatrix} \frac{\partial f}{\partial a_{11}} & \frac{\partial f}{\partial a_{12}} & \cdots & \frac{\partial f}{\partial a_{1n}} \\ \frac{\partial f}{\partial a_{21}} & \frac{\partial f}{\partial a_{22}} & \cdots & \frac{\partial f}{\partial a_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial a_{m1}} & \frac{\partial f}{\partial a_{m2}} & \cdots & \frac{\partial f}{\partial a_{mn}} \end{bmatrix} \tag{B.1}$$

The dimension of the derivative is thus always the same as that of A.

A typical example of matrix differentiation will now be given:

To determine the derivative of $f(A) = \mathbf{b}'A\mathbf{x}$ the scalar function $f(A)$ is expanded into summation form:

$$f(A) = \sum_{i=1}^{m} b_i \sum_{j=1}^{n} a_{ij} x_j \tag{B.2}$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} b_i a_{ij} x_j \tag{B.3}$$

function of A	derivative with respect to A
$\mathbf{b}'A\mathbf{x}$	$\mathbf{x}\mathbf{b}'$
$\mathbf{x}'A'MA\mathbf{x}$	$2MA\mathbf{x}\mathbf{x}'$
$tr(A)$	I

Table B.1: A small table of matrix differentiation equations.

Now determine the derivative with respect to one of the elements of A, a_{pq}:

$$\frac{\partial f}{\partial a_{pq}} = \sum_{i=1}^{m}\sum_{j=1}^{n}\frac{\partial}{\partial a_{pq}}b_i a_{ij} x_j \tag{B.4}$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{n} b_i x_j \frac{\partial a_{ij}}{\partial a_{pq}}. \tag{B.5}$$

The elements of A are independent of each other, so that $\frac{a_{ij}}{\partial a_{pq}} = 0$, unless $i = p$ and $j = q$. Thus

$$\frac{\partial f}{\partial a_{pq}} = b_p x_q \tag{B.6}$$

or

$$\frac{df}{dA} = \mathbf{b}\mathbf{x}'. \tag{B.7}$$

As another example, the differentiation of $h_m^{(s)}(\mathbf{o}_t)$ with respect to transformation matrix W_α needed in order to solve Equation 4.10 for MLLR auxiliary function maximisation will be done. First, $h_m^{(s)}(\mathbf{o}_t)$ is expanded:

$$h_m^{(s)}(\mathbf{o}_t) = \left[(\mathbf{o}_t)'C_m^{(s)-1}\mathbf{o}_t - (\mathbf{o}_t)'C_m^{(s)-1}W_\alpha\xi_m^{(s)}\right. \tag{B.8}$$

$$\left.-(\xi_m^{(s)})'(W_\alpha)'C_m^{(s)-1}\mathbf{o}_t + (\xi_m^{(s)})'(W_\alpha)'C_m^{(s)-1}W_\alpha\xi_m^{(s)}\right] \tag{B.9}$$

Remembering that the transpose of a scalar is equal to the scalar, and that the covariance matrix is diagonal, and using the first two matrix differentiation rules in Table B.1, the expansion of $h_m^{(s)}(\mathbf{o}_t)$ is differentiated with respect to W_α.

$$\frac{\partial}{\partial W_\alpha}h_m^{(s)}(\mathbf{o}_t) = \frac{\partial}{\partial W_\alpha}\left\{(\mathbf{o}_t)'C_m^{(s)-1}\mathbf{o}_t\right\} - 2\frac{\partial}{\partial W_\alpha}\left\{(\mathbf{o}_t)'C_m^{(s)-1}W_\alpha\xi_m^{(s)}\right\}$$

$$+\frac{\partial}{\partial W_\alpha}\left\{(\xi_m^{(s)})'(W_\alpha)'C_m^{(s)-1}W_\alpha\xi_m^{(s)}\right\} \tag{B.10}$$

$$= 0 - 2C_m^{(s)-1}\mathbf{o}_t(\xi_m^{(s)})' + 2C_m^{(s)-1}W_\alpha\xi_m^{(s)}(\xi_m^{(s)})' \tag{B.11}$$

$$= -2C_m^{(s)-1}\left[\mathbf{o}_t - W_\alpha\xi_m^{(s)}\right](\xi_m^{(s)})' \tag{B.12}$$

B.2 Using the Least or Highest Possible Amount of MLLR Regression Classes

Though there are many possible regression class schemes, there are two extremes that are of particular interest:

- global clustering, where all mean vectors in the model are placed in the same single regression class.
- assigning each mean vector to its own exclusive cluster.

Global clustering is closely related to speaker normalisation using spectral shift transformations [22]. Also, because it affords the most general transformation possible, it is possibly the safest clustering method when little is known about the speech modelling problem, or when the smallest amount of adaptation data is available. Using a cluster exclusively for one mean vector is the extreme opposite of global clustering. It is the clustering scheme with the highest likelihood given the data, as it allows more freedom to the adaptation and affords a finer estimation of parameters. It is, in fact, identical to ML reestimation of mean vectors. This is easily seen when Equation 4.12 (restated below for convenience) is examined:

$$\sum_{(s,m)\in R_\alpha} \sum_{t=1}^{T} \gamma_m^{(s)}(t) C_m^{(s)-1} \mathbf{o}_t \xi_m^{(s)\prime} = \sum_{(s,m)\in R_\alpha} \sum_{t=1}^{T} \gamma_m^{(s)}(t) C_m^{(s)-1} W_\alpha \xi_m^{(s)} \xi_m^{(s)\prime}. \tag{B.13}$$

Only one mean vector is assigned to regression class α, and so the summation over states and mixture components disappears so that

$$C_m^{(s)-1} \left(\sum_{t=1}^{T} \gamma_m^{(s)}(t) \mathbf{o}_t \right) \xi_m^{(s)\prime} = C_m^{(s)-1} \left(\sum_{t=1}^{T} \gamma_m^{(s)}(t) \right) W_\alpha \xi_m^{(s)} \xi_m^{(s)\prime} \tag{B.14}$$

$$\sum_{t=1}^{T} \gamma_m^{(s)}(t) \mathbf{o}_t = \left(\sum_{t=1}^{T} \gamma_m^{(s)}(t) \right) W_\alpha \xi_m^{(s)}. \tag{B.15}$$

Thus

$$W_\alpha \xi_m^{(s)} = \hat{\mu_m^{(s)}} = \frac{\sum_{t=1}^{T} \gamma_m^{(s)}(t) \mathbf{o}_t}{\sum_{t=1}^{T} \gamma_m^{(s)}(t)} \tag{B.16}$$

which is identical to the equation for ML mean vector reestimation.

B.3 Distance Measures Used During Acoustic Clustering

Assigning mean vectors to regression classes for MLLR can be done using acoustic distance clustering methods. Several distance measures may be employed, each with its own characteristics. In this section, the properties of a distance measure between two clusters of data will be introduced, after which specific distance measures will be stated (some will be derived).

B.3.1 Representing Clusters Using Gaussian pdfs

We want a distance measure between two data clusters, $\mathcal{X}$ and $\mathcal{Y}$. Let $\{\mathbf{x}_t\}_{1\leq t\leq M}$ be a sequence of M vectors from cluster $\mathcal{X}$, and $\{\mathbf{y}_t\}_{1\leq t\leq N}$ a sequence of N vectors from cluster $\mathcal{Y}$. The vectors from both clusters are n-dimensional. Under the hypothesis the clusters are Gaussian distributions and the data sequence can be summarised by its mean vector and covariance matrix. For example, the mean vector and covariance matrix given by

$$\mu_x = \frac{1}{M}\sum_{t=1}^{M}\mathbf{x}_t \qquad \text{and} \qquad C_x = \frac{1}{M}\sum_{t=1}^{M}(\mathbf{x}_t - \mu_x)(\mathbf{x}_t - \mu_x)' \tag{B.17}$$

respectively summarise sequence $\{\mathbf{x}_t\}$.

B.3.2 Properties of Distance Measures Based on Second-Order Statistical Measures

All the distance measures treated here can be expressed completely using the second-order statistics of the data. Each distance measure $d(\mathcal{X},\mathcal{Y})$ between two data clusters $\mathcal{X}$ and $\mathcal{Y}$ can thus be expressed as a function [6]

$$d(\mathcal{X},\mathcal{Y}) = \phi(\mu_x, C_x, M, \mu_y, C_y, N). \tag{B.18}$$

The distance measures are non-negative

$$d(\mathcal{X},\mathcal{Y}) \geq 0, \tag{B.19}$$

and satisfy the property

$$d(\mathcal{X},\mathcal{X}) = 0. \tag{B.20}$$

B.3.3 The Gaussian Divergence Distance Measure

Two ways of approaching the Gaussian divergence distance measure will now be presented. This section covers the Gaussian divergence distance measure as it is treated by Bimbot *et al.* [6], and Appendix B.3.6 covers the measure as treated by Leggetter [34].

The likelihood of an observation from sequence $\{\mathbf{y}_t\}$ on the distribution of $\mathcal{X}$ is

$$b(\mathbf{y}_t|\mathcal{X}) = \frac{1}{(2\pi)^{n/2}|C_x|^{1/2}} e^{-\frac{1}{2}(\mathbf{y}_t-\mu_x)'C_x^{-1}(\mathbf{y}_t-\mu_x)}. \tag{B.21}$$

If all the vectors $\mathbf{y}_t$ are assumed to be independent observations, then the average log-likelihood of sequence $\{\mathbf{y}_t\}_{1\leq t\leq N}$ for distribution $\mathcal{X}$ is

$$\begin{aligned}
\mathcal{L}_{\mathcal{X}}(\mathbf{y}_1^N) &= \frac{1}{N}\ln\left[b(\mathbf{y}_1\ldots\mathbf{y}_N|\mathcal{X})\right] && \text{(B.22)}\\
&= \frac{1}{N}\sum_{t=1}^{N}\ln\left[b(\mathbf{y}_t|\mathcal{X})\right] && \text{(B.23)}\\
&= -\frac{1}{2}\left[n\ln 2\pi + \ln|C_x| + \frac{1}{N}\sum_{t=1}^{N}(\mathbf{y}_t-\mu_x)'C_x^{-1}(\mathbf{y}_t-\mu_x)\right]\\
&= -\frac{1}{2}\left[n\ln 2\pi + \ln|C_x| + \frac{1}{N}\sum_{t=1}^{N}(\mathbf{y}_t-\mu_y+\mu_y-\mu_x)'C_x^{-1}(\mathbf{y}_t-\mu_y+\mu_y-\mu_x)\right]
\end{aligned}$$

The following property:

$$\frac{1}{N}\sum_{t=1}^{N}(\mathbf{y}_t-\mu_y)'C_x^{-1}(\mathbf{y}_t-\mu_y) = \text{tr}\,(C_yC_x^{-1}) \tag{B.24}$$

is now substituted so that

$$\mathcal{L}_{\mathcal{X}}(\mathbf{y}_1^N) + \frac{n}{2}\ln 2\pi = -\frac{1}{2}\left[\ln|C_x| + \text{tr}\,(C_yC_x^{-1}) + (\mu_y-\mu_x)'C_x^{-1}(\mu_y-\mu_x)\right] \tag{B.25}$$

or

$$\begin{aligned}
&\frac{2}{n}\mathcal{L}_{\mathcal{X}}(\mathbf{y}_1^N) + \ln 2\pi + \frac{1}{n}\ln|C_y| + 1\\
&\quad= \frac{1}{n}\left[\ln\left(\frac{|C_y|}{|C_x|}\right) - \text{tr}\,(C_yC_x^{-1}) - (\mu_y-\mu_x)'C_x^{-1}(\mu_y-\mu_x)\right] + 1. && \text{(B.26)}
\end{aligned}$$

If the Gaussian likelihood measure d_{G} is defined as

$$d_{\text{G}}(\mathcal{X},\mathcal{Y}) = \frac{1}{n}\left[\text{tr}\,(C_yC_x^{-1}) - \ln\left(\frac{|C_y|}{|C_x|}\right) + (\mu_y-\mu_x)'C_x^{-1}(\mu_y-\mu_x)\right] - 1, \tag{B.27}$$

then

$$\underset{\mathcal{X}}{\text{argmax}}\,\mathcal{L}_{\mathcal{X}}(\mathbf{y}_t^N) = \underset{\mathcal{X}}{\text{argmin}}\,d_{\text{G}}(\mathcal{X},\mathcal{Y}). \tag{B.28}$$

B.3.3.1 Properties of the Gaussian Likelihood Measure

The measure as it is defined is not symmetric, in other words $d_{\mathrm{G}}(\mathcal{X},\mathcal{Y}) \neq d_{\mathrm{G}}(\mathcal{Y},\mathcal{X})$. The measure is only equal to zero when both the covariance and mean of both sequences are equal, i.e. $d_{\mathrm{G}}(\mathcal{X},\mathcal{Y}) = 0$ if and only if $C_x = C_y$ and $\mu_x = \mu_y$.

B.3.3.2 A Variant of the Gaussian Likelihood Measure

According to Bimbot *et al.* [6] the mean vectors μ_x and μ_y are more sensitive to channel characteristics and noise present in a distorted speech signal than the covariance matrices C_x and C_y. The difference terms $(\mu_y - \mu_x)$ in Equation B.27 may thus make the Gaussian likelihood measure inconsistent when the speech is very distorted. A variant Gaussian distance measure determined completely by the more noise-robust covariance matrices is given by

$$d_{\mathrm{G}}(\mathcal{X},\mathcal{Y}) = \frac{1}{n}\left[\mathrm{tr}\,(C_y C_x^{-1}) - \ln\left(\frac{|C_y|}{|C_x|}\right)\right] - 1. \tag{B.29}$$

This alternative measure has the same properties as the Gaussian likelihood measure, i.e. it is non-symmetric and zero only when both the covariance matrices and mean vectors are equal.

B.3.3.3 Symmetrisation

When one cluster $(\mathcal{X})$ is a well estimated reference set and the other $(\mathcal{Y})$ is a cluster estimated using only a limited amount of data, then the directed Gaussian likelihood measure $d_{\mathrm{G}}(\mathcal{X},\mathcal{Y})$ defined by Equation B.27 will be an accurate measure. In practice, however, both reference and test clusters are estimated from a limited amount of data, and neither are exact. Both directed distance measures $d_{\mathrm{G}}(\mathcal{X},\mathcal{Y})$ and $d_{\mathrm{G}}(\mathcal{Y},\mathcal{X})$ could be used in order to check the validity of a measure, but the discrepancy between $d_{\mathrm{G}}(\mathcal{X},\mathcal{Y})$ and $d_{\mathrm{G}}(\mathcal{Y},\mathcal{X})$ becomes great when the number of test and reference vectors is greatly disproportionate (i.e. when N/M is very different from 1). Symmetrisation of a distance measure can improve classification compared to both directed asymmetric terms taken individually. Constructing a symmetric measure by taking the average of the two asymmetric measures is the simplest method.

$$d_{\mathrm{sym}}(\mathcal{X},\mathcal{Y}) = \frac{1}{2}d_{\mathrm{asym}}(\mathcal{X},\mathcal{Y}) + \frac{1}{2}d_{\mathrm{asym}}(\mathcal{Y},\mathcal{X}) = d_{\mathrm{sym}}(\mathcal{Y},\mathcal{X}). \tag{B.30}$$

Other methods of symmetrisation taking into account the relative numbers of vectors for the respective clusters are possible [6].

The symmetrised Gaussian likelihood measure d_{Gs} is then given by

$$\begin{aligned} d_{\mathrm{Gsym}} &= \frac{1}{2}d_{\mathrm{G}}(\mathcal{X},\mathcal{Y}) + \frac{1}{2}d_{\mathrm{G}}(\mathcal{Y},\mathcal{X}) \\ &= \frac{1}{2}\left[\frac{1}{n}\mathrm{tr}\,(C_y C_x^{-1})\right. \\ &\quad \left. + \frac{1}{n}\mathrm{tr}\,(C_x C_y^{-1}) + \frac{1}{n}(\mu_y-\mu_x)'(C_x^{-1}+C_y^{-1})(\mu_y-\mu_x)\right] - 1 \end{aligned} \tag{B.31}$$

B.3.4 Arithmetic-Geometric Sphericity Measure

According to Bimbot [6] a likelihood measure of the proportionality of covariance matrix C_y given covariance matrix C_x is defined by

$$S(C_y|C_x) = \left\{ \frac{\left|C_x^{-\frac{1}{2}} C_y C_x^{-\frac{1}{2}}\right|}{\left[\frac{1}{n}\mathrm{tr}\,\left(C_x^{-\frac{1}{2}} C_y C_x^{-\frac{1}{2}}\right)\right]^p} \right\}^{\frac{N}{2}}. \tag{B.32}$$

The average log-likelihood $\mathcal{L}_{\mathcal{X}}(\mathbf{y}^N)$ for the sphericity test is then

$$\mathcal{L}_{\mathcal{X}}(\mathbf{y}^N) = \frac{1}{N}\ln\left[S(C_y|C_x)\right]. \tag{B.33}$$

Defining the arithmetic-geometric sphericity measure to be

$$d_{\mathrm{S}}(\mathcal{X},\mathcal{Y}) = \ln\left[\frac{\frac{1}{n}\mathrm{tr}\,(C_y C_x^{-1})}{\left(\frac{|C_y|}{|C_x|}\right)^{\frac{1}{n}}}\right], \tag{B.34}$$

we have

$$\underset{\mathcal{X}}{\mathrm{argmax}}\ \mathcal{L}_{\mathcal{X}}(\mathbf{y}^N) = \underset{\mathcal{X}}{\mathrm{argmin}}\ d_{\mathrm{S}}(\mathcal{X},\mathcal{Y}). \tag{B.35}$$

B.3.4.1 Properties of the Arithmetic-Geometric Sphericity Measure

The measure is non-negative:

$$d_{\mathrm{S}}(\mathcal{X},\mathcal{Y}) \geq 0 \tag{B.36}$$

and is zero if and only if C_x and C_y are proportional (i.e. all the eigenvalues of $X^{-\frac{1}{2}}YX^{-\frac{1}{2}}$ are equal to 1). The measure is not symmetric:

$$d_{\mathrm{S}}(\mathcal{X},\mathcal{Y}) \neq d_{\mathrm{S}}(\mathcal{Y},\mathcal{X}). \tag{B.37}$$

B.3.4.2 Symmetrisation of the Arithmetic-Geometric Sphericity Measure

Equation B.30 is used once more to provide a symmetric measure.

$$\begin{aligned} d_{\text{Ssym}}(\mathcal{X},\mathcal{Y}) &= \frac{1}{2}d_{\text{S}}(\mathcal{X},\mathcal{Y}) + \frac{1}{2}d_{\text{S}}(\mathcal{Y},\mathcal{X}) \\ &= \frac{1}{2}\left\{\ln\left[\text{tr }(C_yC_x^{-1})\text{ tr }(C_xC_y^{-1})\right] - 2\ln n\right\}. \end{aligned} \tag{B.38}$$

B.3.5 Expressing Distance Measures in Terms of Eigenvalues

It is possible to express the Gaussian likelihood and arithmetic-geometric sphericity measures in terms of the eigenvalues of $\Gamma = C_x^{-\frac{1}{2}}C_yC_x^{-\frac{1}{2}}$, where the eigenvalues (λ_i, where $1 \leq i \leq n$) are the roots of equation

$$|\Gamma - \lambda I| = 0. \tag{B.39}$$

The distance measures treated thus far may be expressed in terms of three functions of the eigenvalues:

1. The arithmetic mean:

$$a(\lambda_1,\ldots,\lambda_n) = \frac{1}{n}\sum_{i=1}^{n}\lambda_i \tag{B.40}$$

2. The geometric mean:

$$g(\lambda_1,\ldots,\lambda_n) = \left(\prod_{i=1}^{n}\lambda_i\right)^{\frac{1}{n}} \tag{B.41}$$

3. The harmonic mean:

$$h(\lambda_1,\ldots,\lambda_n) = \left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\lambda_i}\right)^{-1} \tag{B.42}$$

It should be noted that

$$a \geq g \geq h \tag{B.43}$$

where equality holds only when all eigenvalues are equal.

When swapping $\mathcal{X}$ and $\mathcal{Y}$, a becomes $1/h$, g becomes $1/g$ and h becomes $1/a$ (also Γ becomes Γ^{-1}, C_x becomes C_y etc.). These three functions may also be expressed as

$$a(\lambda_1, \ldots, \lambda_n) = \frac{1}{n}\text{tr}\,\Lambda = \frac{1}{n}\text{tr}\,\Gamma = \frac{1}{n}\text{tr}\,(C_y C_x^{-1}) \tag{B.44}$$

$$g(\lambda_1, \ldots, \lambda_n) = (|\Lambda|)^{\frac{1}{n}} = (|\Gamma|)^{\frac{1}{n}} = \left(\frac{|Y|}{|X|}\right)^{\frac{1}{n}} \tag{B.45}$$

$$h(\lambda_1, \ldots, \lambda_n) = \frac{n}{\text{tr}\,(\Lambda^{-1})} = \frac{n}{\text{tr}\,(\Gamma^{-1})} = \frac{n}{\text{tr}\,\left(C_x C_y^{-1}\right)} \tag{B.46}$$

where Λ is the diagonal eigenvalue matrix.

Expression for the distance measures may now be found in terms of the eigenvalues. For instance, the asymmetric Gaussian likelihood measure (Equation B.27) may thus be expressed as

$$d_{\text{G}}(\mathcal{X}, \mathcal{Y}) = a - \ln g + \frac{1}{n}(\mu_y - \mu_x)' C_x^{-1}(\mu_y - \mu_x) - 1. \tag{B.47}$$

Similarly, the asymmetric arithmetic-geometric sphericity measure may be expressed as

$$d_{\text{S}}(\mathcal{X}, \mathcal{Y}) = \ln\left(\frac{a}{g}\right) \tag{B.48}$$

and the symmetric arithmetic-geometric sphericity measure is

$$d_{\text{Ssym}}(\mathcal{X}, \mathcal{Y}) = \frac{1}{2}\ln\left(\frac{a}{h}\right). \tag{B.49}$$

The arithmetic-harmonic sphericity measure, a new measure that is proportional to the symmetric arithmetic-geometric sphericity measure [5] may now be defined as

$$d_{\text{H}}(\mathcal{X}, \mathcal{Y}) = \ln\left(\frac{a}{h}\right) \tag{B.50}$$

$$= \ln\left(\text{tr}\,(C_y C_x^{-1})\text{tr}\,(C_x C_y^{-1})\right) - 2\ln n. \tag{B.51}$$

The properties of the arithmetic-harmonic sphericity measure are the same as those for the symmetric arithmetic-geometric sphericity measure.

B.3.6 Gaussian Divergence as Defined by Leggetter [33]

Divergence is a distance measure of the separability of two distributions based on the difference between their means.

If there are two Gaussian distributions, distribution $b_1(\mathbf{y})$ for class w_1, with mean μ_1 and covariance C_1 and distribution $b_2(\mathbf{y})$ for class w_2, with mean μ_2 and covariance C_2 then the log-likelihood ratio is

$$\Lambda(\mathbf{y}) = \ln\left[\frac{b_1(\mathbf{y})}{b_2(\mathbf{y})}\right] \tag{B.52}$$

The expectation of the log-likelihood ratio for $i = 1$ or $i = 2$ is then given by

$$E[\Lambda(\mathbf{y})|w_i] = \int_{-\infty}^{\infty} \ln\left[\frac{b_1(\mathbf{y})}{b_2(\mathbf{y})}\right] b_i(\mathbf{y})d\mathbf{y}. \tag{B.53}$$

If $H(i,j)$ is defined by

$$H(i,j) = \int_{-\infty}^{\infty} \ln\left[\frac{b_i(\mathbf{y})}{b_j(\mathbf{y})}\right] b_i(\mathbf{y})d\mathbf{y} = E\left[\ln\left(\frac{b_i(\mathbf{y})}{b_j(\mathbf{y})}\right)\middle| w_i\right], \tag{B.54}$$

the directed divergence is then given by

$$d_{\text{dir}} = H(w_1, w_2) \tag{B.55}$$

and the symmetric divergence is given by

$$d_{\text{sym}} = H(w_1, w_2) + H(w_2, w_1). \tag{B.56}$$

In order to expand these two divergence measures, the log-likelihood ratio is first expanded.

$$\begin{aligned}\ln\left(\frac{b_1(\mathbf{y})}{b_2(\mathbf{y})}\right) = \quad & \frac{1}{2}(\mathbf{y}-\mu_2)'C_2^{-1}(\mathbf{y}-\mu_2) \\ & -\frac{1}{2}(\mathbf{y}-\mu_1)'C_1^{-1}(\mathbf{y}-\mu_1) + \frac{1}{2}\ln\frac{|C_2|}{|C_1|}.\end{aligned} \tag{B.57}$$

Now

$$\begin{aligned}H(w_1, w_2) = \quad & \frac{1}{2}E\left[(\mathbf{y}-\mu_2)'C_2^{-1}(\mathbf{y}-\mu_2)|w_1\right] \\ & \frac{1}{2}E\left[(\mathbf{y}-\mu_1)'C_1^{-1}(\mathbf{y}-\mu_1)|w_1\right] + \frac{1}{2}\ln\frac{|C_2|}{|C_1|}.\end{aligned} \tag{B.58}$$

Using the properties

$$(\mathbf{y}-\mu_i)'C_i^{-1}(\mathbf{y}-\mu_i) = \text{tr}\ \left[(\mathbf{y}-\mu_i)'C_i^{-1}(\mathbf{y}-\mu_i)\right] \tag{B.59}$$

and

$$\text{tr}\ (AB) = \text{tr}\ (BA), \tag{B.60}$$

we have

$$\begin{aligned} E\left[(\mathbf{y}-\mu_1)'C_1^{-1}(\mathbf{y}-\mu_1)|w_1\right] &= E\left[\text{tr}\left((\mathbf{y}-\mu_1)'C_1^{-1}(\mathbf{y}-\mu_1)\right)|w_1\right] \\ &= E\left[\text{tr}\left(C_1^{-1}(\mathbf{y}-\mu_1)(\mathbf{y}-\mu_1)'\right)|w_1\right] \\ &= \text{tr}\left(C_1^{-1}C_1\right) \\ &= \text{tr}\left(I\right) \end{aligned} \tag{B.61}$$

and

$$\begin{aligned} E\left[(\mathbf{y}-\mu_2)'C_2^{-1}(\mathbf{y}-\mu_2)|w_1\right] &= E\left[\text{tr}\left((\mathbf{y}-\mu_2)'C_2^{-1}(\mathbf{y}-\mu_2)\right)|w_1\right] \\ &= E\left[\text{tr}\left(C_2^{-1}(\mathbf{y}-\mu_2)(\mathbf{y}-\mu_2)'\right)|w_1\right] \\ &= E\left[\text{tr}\left(C_2^{-1}(\mathbf{y}-\mu_1+\mu_1-\mu_2)(\mathbf{y}-\mu_1+\mu_1-\mu_2)'\right)|w_1\right] \\ &= \text{tr}\left(C_2^{-1}\left(C_1^{-1}+(\mu_1-\mu_2)(\mu_1-\mu_2)'\right)\right) \\ &= \text{tr}\left(C_2^{-1}C_1^{-1}+(\mu_1-\mu_2)'C_2^{-1}(\mu_1-\mu_2)\right). \end{aligned} \tag{B.62}$$

Given the above, the directed divergence d_{dir} thus becomes

$$\begin{aligned} d_{\text{dir}} &= H(w_1,w_2) \\ &= \frac{1}{2}\text{tr}\left(C_2^{-1}C_1-I\right)+\frac{1}{2}(\mu_1-\mu_2)'C_2^{-1}(\mu_1-\mu_2)+\frac{1}{2}\ln\frac{|C_2|}{|C_1|} \end{aligned} \tag{B.63}$$

and the symmetric divergence d_{sym} is given by

$$\begin{aligned} d_{\text{sym}} &= H(w_1,w_2)+H(w_2,w_1) \\ &= \frac{1}{2}\text{tr}\left(C_1^{-1}C_2+C_2^{-1}C_1-2I\right)+\frac{1}{2}(\mu_1-\mu_2)'(C_1^{-1}+C_2^{-1})(\mu_1-\mu_2). \end{aligned} \tag{B.64}$$

Note that the directed divergence is the same as the asymmetric Gaussian likelihood measure (Equation B.27), and the symmetric divergence is the same as the symmetric Gaussian likelihood measure (Equation B.27).

B.3.7 Likelihood Measure

The likelihood measure is used to measure the similarity between distributions for the purpose of tree-based state tying of HMMs [44, 43, 53, 34].

Executing a complete maximum likelihood training pass is computationally expensive, and so an estimate of the log-likelihood for the training data given a particular set of state distributions is used. According to Odell [43] the following assumptions are made when the simplified estimate is used:

- The state-occupation probabilities $\gamma_s(t)$ are not altered during the clustering procedure.

- The contribution of the state transition probabilities to the total likelihood are ignored. Though the state transition probabilities have a significant impact on the total likelihood, their contribution only changes when there are changes in the state-occupation probabilities. Since the state-occupation probabilities — and thus also the state sequence — are assumed fixed, the state transition probabilities will remain constant during clustering.

- It is assumed that the total likelihood can be estimated by the average of the log-likelihoods of the state pdfs weighted by the state-occupation probabilities.

From the above assumptions, the approximation of the total likelihood of the group of distributions S generating observation sequence O is then given by:

$$\begin{aligned} \mathcal{L} &= \sum_{t=1}^{T} \sum_{s \in S} \gamma_s(t) \ln \left[P(\mathbf{o}_t, \mu_s, C_s) \right] && \text{(B.65)} \\ &\approx \ln \left[P(O, S) \right] && \text{(B.66)} \end{aligned}$$

where $P(\mathbf{o}_t, \mu_s, C_s)$ is the probability of state pdf s for observation $\mathbf{o}_t$, $P(O, S)$ is the probability of the group of state pdfs S generating O and $\gamma_s(t)$ is the state-occupation probability of state s at time t.

Suppose we have an HMM that has single Gaussian pdfs of dimension n for each emitting state. The log-likelihood of a Gaussian pdf in the HMM generating a single observation $\mathbf{o}$ is then

$$\begin{aligned} \ln \left[b_s(\mathbf{o}) \right] &= \ln \left[\frac{1}{(2\pi)^{n/2} \left| C_s \right|^{1/2}} e^{-\frac{1}{2}(\mathbf{o}-\mu_s)' C_s^{-1} (\mathbf{o}-\mu_s)} \right] && \text{(B.67)} \\ &= -\frac{1}{2} \left[n \ln(2\pi) + \ln(|C_s|) + (\mathbf{o} - \mu_s)' C_s^{-1} (\mathbf{o} - \mu_s) \right]. && \text{(B.68)} \end{aligned}$$

The likelihood of the set of Gaussian pdfs S generating an entire sequence of T observations $\{\mathbf{o}_t\}_{t=1 \ldots T}$ thus becomes

$$\mathcal{L} = \sum_{t=1}^{T} \sum_{s \in S} -\frac{1}{2} \left[n \ln(2\pi) + \ln(|C_s|) + (\mathbf{o}_t - \mu_s)' C_s^{-1} (\mathbf{o} - \mu_s) \right] \gamma_s(t). \qquad \text{(B.69)}$$

To reestimate the covariance of a Gaussian pdf given the new observation sequence, the following formula is applied

$$C_s = \frac{\sum_{t=1}^{T} \gamma_s(t) \left(\mathbf{o}_t - \mu_s\right) \left(\mathbf{o}_t - \mu_s\right)'}{\sum_{t=1}^{T} \gamma_s(t)}. \tag{B.70}$$

Thus

$$\begin{aligned}
C_s \sum_{t=1}^{T} I\gamma_s(t) &= \sum_{t=1}^{T} \gamma_s(t) \left(\mathbf{o}_t - \mu_s\right) \left(\mathbf{o}_t - \mu_s\right)' \\
\sum_{t=1}^{T} I\gamma_s(t) &= \sum_{t=1}^{T} \gamma_s(t) C_s^{-1} \left(\mathbf{o}_t - \mu_s\right) \left(\mathbf{o}_t - \mu_s\right)' \\
\text{tr} \left(\sum_{t=1}^{T} I\gamma_s(t) \right) &= \text{tr} \left(\sum_{t=1}^{T} \gamma_s(t) C_s^{-1} \left(\mathbf{o}_t - \mu_s\right) \left(\mathbf{o}_t - \mu_s\right)' \right) \\
n \sum_{t=1}^{T} \gamma_s(t) &= \text{tr} \left(\sum_{t=1}^{T} \gamma_s(t) \left(\mathbf{o}_t - \mu_s\right)' C_s^{-1} \left(\mathbf{o}_t - \mu_s\right) \right) \\
n \sum_{t=1}^{T} \gamma_s(t) &= \sum_{t=1}^{T} \gamma_s(t) \left(\mathbf{o}_t - \mu_s\right)' C_s^{-1} \left(\mathbf{o}_t - \mu_s\right).
\end{aligned} \tag{B.71}$$

Thus the likelihood in Equation B.69 becomes

$$\mathcal{L} = \sum_{s \in S} -\frac{1}{2} \left\{ n \left[1 + \ln(2\pi)\right] + \ln(|C_s|) \right\} \sum_{t=1}^{T} \gamma_s(t). \tag{B.72}$$

The difference in likelihood if a single distribution e is used to represent the set of distributions D would be

$$\delta\mathcal{L} = \mathcal{L}_e - \mathcal{L}_D \tag{B.73}$$

$$\begin{aligned}
&= -\frac{1}{2} \left\{ n \left[1 + \ln(2\pi)\right] + \ln(|C_e|) \right\} \sum_{t=1}^{T} \gamma_e(t) \\
&\quad + \sum_{d \in D} \frac{1}{2} \left\{ n \left[1 + \ln(2\pi)\right] + \ln(|C_d|) \right\} \sum_{t=1}^{T} \gamma_d(t) \\
&= -\frac{1}{2} n \left[1 + \ln(2\pi)\right] \sum_{t=1}^{T} \gamma_e(t) - \frac{1}{2} \ln(|C_e|) \sum_{t=1}^{T} \gamma_e(t) \\
&\quad + \sum_{d \in D} \frac{1}{2} \left\{ n \left[1 + \ln(2\pi)\right] \right\} \sum_{t=1}^{T} \gamma_d(t) + \sum_{d \in D} \frac{1}{2} \ln(|C_d|) \sum_{t=1}^{T} \gamma_d(t)
\end{aligned} \tag{B.74}$$

remembering that $\gamma_e(t) = \sum_{d \in D} \gamma_d(t)$

$$\begin{aligned}
\delta\mathcal{L} &= -\sum_{d \in D} \frac{1}{2} n \left[1 + \ln(2\pi)\right] \sum_{t=1}^{T} \gamma_d(t) - \frac{1}{2} \ln(|C_e|) \sum_{t=1}^{T} \gamma_e(t) \\
&\quad + \sum_{d \in D} \frac{1}{2} \left\{n \left[1 + \ln(2\pi)\right]\right\} \sum_{t=1}^{T} \gamma_d(t) + \sum_{d \in D} \frac{1}{2} \ln(|C_d|) \sum_{t=1}^{T} \gamma_d(t) \\
&= -\frac{1}{2} \ln(|C_e|) \sum_{t=1}^{T} \gamma_e(t) + \frac{1}{2} \sum_{d \in D} \ln(|C_d|) \sum_{t=1}^{T} \gamma_d(t). \qquad \text{(B.75)}
\end{aligned}$$

Similarly, the change in likelihood when a single distribution e is split into a set of distributions D may be expressed by

$$\delta\mathcal{L} = -\frac{1}{2} \sum_{d \in D} \ln(|C_d|) \sum_{t=1}^{T} \gamma_d(t) + \frac{1}{2} \ln(|C_e|) \sum_{t=1}^{T} \gamma_e(t). \qquad \text{(B.76)}$$

Appendix C

Reducing the Memory Requirements of the CBKLT

As with the computation of the eigenvoices using the Karhunen-Loève transform, the memory requirements will be massive should the large covariance or correlation matrix of the supermean vectors be used. A similar procedure to Section 5.4.3 can be used to reduce the memory requirements.

Finding a data matrix for the intra-class dispersion matrix S_w is the key to applying the same memory requirement reducing technique. If a data matrix Z_i is defined for each class i as follows:

$$Z_i = \left[[\mathbf{y}_1^{(i)} - \mu_i] \quad [\mathbf{y}_2^{(i)} - \mu_i)] \quad \dots \quad [\mathbf{y}_{n_i}^{(i)} - \mu_i] \right], \quad i = 1 \dots c, \tag{C.1}$$

then a data matrix for all the classes Z is given by

$$Z = \left(\frac{1}{n_T} \right)^{1/2} \left[Z_1 \quad Z_2 \quad \dots \quad Z_c \right]. \tag{C.2}$$

The intra-class dispersion matrix S_w can now be expressed in terms of Z

$$S_w = ZZ'. \tag{C.3}$$

To obtain Equation C.3, Equation 7.23 was substituted for the class probability P_i in the intra-class dispersion matrix.

As in Section 5.4.3 the right singular matrix of Z can now be used to determine the left singular matrix of Z — the left singular matrix of Z being the eigenvectors of the

intra-class dispersion matrix S_w. Only r of these eigenvectors will not describe the null-space, and only these r eigenvectors are needed to transform the feature (in our case the speaker) space. For r the following holds:

$$r \leq \min(m, n_T) \tag{C.4}$$

where m is the number of elements in a supervector. Normally, the dimension of a supervector will be much greater than the number of speakers available for creating the transform, so that r will be much smaller than m. The small amount of non-zero eigenvalues means that the $m \times r$ transform B_+ can be computed without ever having to determine the massive $m \times m$ intra-class dispersion matrix S_w.

Now the memory requirements for the second part of the transform must be considered. As the inter-class dispersion matrix is also a massive $m \times m$ matrix, computing and storing it should be avoided. The equation for $\hat{S}_b$ in expanded form is restated here for convenience:

$$\hat{S}_b = \sum_{i=1}^{c} P_i B'_+(\mu_i - \mu_T)(\mu_i - \mu_T)' B_+ \tag{C.5}$$

$$= \sum_{i=1}^{c} \frac{n_i}{n_T}(\hat{\mu}_i - \hat{\mu_T})(\hat{\mu}_i - \hat{\mu_T})' \tag{C.6}$$

where $\hat{\mu}_i$ is the mean of the features of class i in the transformed space and $\hat{\mu}_i$ is the mean of all features in the transformed space. After the first transformation, the features are r-dimensional, and so these transformed mean vectors are also only r instead of m-dimensional. We wish to determine the data matrix Q from which $\hat{S}_b$ could have been computed. From Equation C.5, Q can be extracted and is given by

$$Q = \left[[(\tfrac{n_1}{n_T})^{1/2} B'_+(\mu_1 - \mu_T)] \quad [(\tfrac{n_2}{n_T})^{1/2} B'_+(\mu_2 - \mu_T)] \quad \ldots \quad [(\tfrac{n_c}{n_T})^{1/2} B'_+(\mu_c - \mu_T)] \right] \tag{C.7}$$

so that

$$\hat{S}_b = QQ'. \tag{C.8}$$

The computation of Q can be accomplished in two ways. We could apply B'_+ and transform the feature vectors into the new r-dimensional space and compute the class mean vectors and total mean vector from the transformed feature vectors. The second option involves first computing the required mean vectors and then applying the B'_+ transform

on them. In order to choose one of the methods, the computational and memory requirements must be compared. For the experiments on the ISOLET corpus, there were 120 supervectors, the dimension of each supervector was 2862 and there were 2 classes.

Only the final results of the detailed analysis will be given:

- Method 1: (41 213 040 multiplication or division operations) + (41 212 680 addition or subtraction operations)
- Method 2: (689 742 multiplication or division operations) + (1 373 520 addition or subtraction operations)

It is clear that the second method is computationally far superior. The memory requirements were also much less than those required for the first method, as the first method needs to store the transformed feature vectors. The dominant cause for the first method's larger computational requirements, is that more B'_+ transformations are used. Each of the 120 supervector features require 2862 multiplications and 2861 additions to be transformed.

Q is thus computed using the second method. It should be noted that μ_T need not be computed using all the supervector features, as the class means can be used. As Equation C.9 demonstrates, fewer addition operations are necessary when computing μ_T using the previously calculated class mean vectors:

$$\mu_T = \frac{1}{n_T}\sum_{i=1}^{c}\sum_{j=1}^{n_i}\mathbf{y}_j^{(i)} = \sum_{i=1}^{c}\frac{n_i}{n_T}\frac{1}{n_i}\sum_{j=1}^{n_i}\mathbf{y}_j^{(i)} = \sum_{i=1}^{c}P_i\frac{1}{n_i}\sum_{j=1}^{n_i}\mathbf{y}_j^{(i)} = \sum_{i=1}^{c}P_i\mu_i \tag{C.9}$$

V_{++} can now be computed by using the same principle as in Section 5.4.3. First the right singular matrix and singular values of Q are computed by determining the eigenvectors and eigenvalues of $Q'Q$. The left singular matrix of Q is then computed using the right singular matrix and the singular values of Q. V_{++} consists of the dominant eigenvectors of $\hat{S}_b$, and these eigenvectors correspond to the first columns of the left singular matrix of Q.

Appendix D

Supplementary Mathematical Derivations for Weighted Projection

This section covers some of the basic linear algebra proofs that are used in the formulation of the weighted projection adaptation method introduced by Westwood [51]. The proofs may be found in greater detail in Strang [48].

D.1 Projection onto a Subspace

Suppose that the columns of matrix A are n linearly independent vectors $\mathbf{a_1}, \ldots, \mathbf{a_n}$ that span the n-dimensional subspace R^n. We wish to find the point in this subspace that is closest to vector $\mathbf{b}$ in the m-dimensional space R^m $(m \geq n)$. In other words, the distance between the linear combination $x_1\mathbf{a}_1 + \ldots + x_n\mathbf{a}_n$ and $\mathbf{b}$ is to be minimised.

This problem is solved in three steps:

- Find the best $\mathbf{x}$ value, denoted by $\hat{\mathbf{x}}$.
- Determine the projection $\mathbf{p} = A\hat{\mathbf{x}}$.
- Find the projection matrix P, such that $\mathbf{p} = P\mathbf{b}$.

The error vector $\mathbf{e}$ between $\mathbf{b}$ and its projection on the subspace is given by

$$\mathbf{e} = \mathbf{b} - A\hat{\mathbf{x}}. \tag{D.1}$$

The length of $\mathbf{e}$ will be the shortest when it is perpendicular to all the vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n$

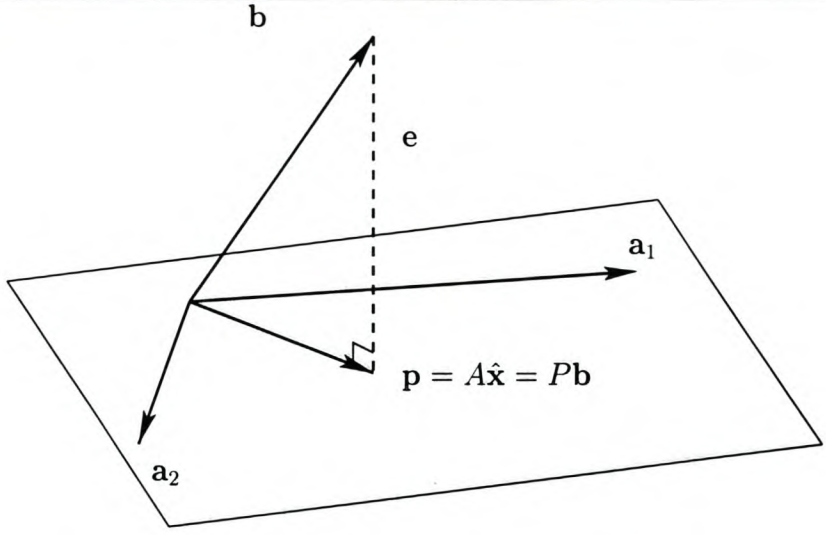

Figure D.1: Projection onto a subspace. $\mathbf{b}$ is projected onto A. The projection $\mathbf{p}$ is the nearest point to $\mathbf{b}$ in the column space of A, and the error $\mathbf{e}$ is perpendicular to A (i.e. in the nullspace of A').

(refer to the geometry depicted in Figure D.1). Thus

$$\begin{aligned} \mathbf{a}_1'(\mathbf{b} - A\hat{\mathbf{x}}) &= 0 \\ &\vdots \\ \mathbf{a}_n'(\mathbf{b} - A\hat{\mathbf{x}}) &= 0 \end{aligned} \quad \text{or} \quad \begin{bmatrix} \mathbf{a}_1' \\ \vdots \\ \mathbf{a}_n' \end{bmatrix} \left[\mathbf{b} - A\hat{\mathbf{x}}\right] = \begin{bmatrix} \mathbf{0} \end{bmatrix} \quad \text{or} \quad A'(\mathbf{b} - A\hat{\mathbf{x}}) = \mathbf{0}. \tag{D.2}$$

Solving for $\hat{\mathbf{x}}$, we find

$$\hat{\mathbf{x}} = (A'A)^{-1}A'\mathbf{b}. \tag{D.3}$$

Now that $\hat{\mathbf{x}}$ has been determined, the projection $\mathbf{p}$ is

$$\mathbf{p} = A\hat{\mathbf{x}} = A(A'A)^{-1}A'\mathbf{b}. \tag{D.4}$$

and, from the above the projection matrix P is

$$P = A(A'A)^{-1}A' \tag{D.5}$$

The above derivations hold only when $A'A$ is invertible. It will now be proven that $A'A$ is invertible if and only if the columns of A are linearly independent.

Proof: If the nullspace of a matrix contains only the zero vector, then the columns of the matrix are linearly independent, which for a square matrix implies invertibility. First it will be shown that $A'A$ has the same nullspace as A:

Suppose that A is any matrix and that vector $\mathbf{x}$ is in its nullspace. Then

$$A\mathbf{x} = \mathbf{0}. \tag{D.6}$$

Premultiplying by A'

$$A'A\mathbf{x} = \mathbf{0}. \tag{D.7}$$

Thus $\mathbf{x}$ is also in the nullspace of $A'A$. Now consider the nullspace of $A'A$. If $\mathbf{x}$ is a vector in the nullspace of $A'A$, then

$$A'A\mathbf{x} = \mathbf{0}. \tag{D.8}$$

Multiplying by $\mathbf{x}'$ (note that we cannot multiply by the inverse of A', as this does not exist in general)

$$\mathbf{x}'A'A\mathbf{x} = \mathbf{x}'\mathbf{0} \quad \text{or} \quad (A\mathbf{x})'(A\mathbf{x}) = 0 \quad \text{or} \quad \|A\mathbf{x}\|^2 = 0. \tag{D.9}$$

Thus vector $A\mathbf{x}$ has zero length and must be the zero vector $A\mathbf{x} = \mathbf{0}$.

Therefore every vector in one nullspace also exists in the other nullspace, and the nullspaces of the two matrices A and $A'A$ are identical. If A has dependent columns, $A'A$ has dependent columns, and if A has independent columns, so does $A'A$.

To summarise: When columns of A are linearly independent, $A'A$ is square, symmetric and invertible.

Appendix E

Additional Results for Experiments on ISOLET

E.1 MAP-Reestimation Results

Table E.1 contains all results for the set of experiments testing the effect of different τ-values on the MAP-reestimation of SG systems. Table E.2 contains all results for the set of experiments testing the effect of different τ-values on the MAP-reestimation of GMM systems.

E.2 The Phonemes of the Dialect-Independent Transcriptions for TIMIT

The following phonemes remain in the dialect-independent transcriptions: ae, ao, ay, d, dh, dx, eh, epi, er, g, hh, ih, iy, k, l, m, n, ow, oy, r, s, sh, t, uw, w, y.

Adaptation method	SG PLP	SG LPCC
SI	16.15%	16.18%
MAP, $\tau = 5$, $O_{\text{min}} = 8$	15.61%	15.32%
MAP, $\tau = 10$, $O_{\text{min}} = 8$	15.66%	15.25%
MAP, $\tau = 20$, $O_{\text{min}} = 8$	15.81%	14.98%
MAP, $\tau = 30$, $O_{\text{min}} = 8$	13.57%	11.55%
MAP, $\tau = 40$, $O_{\text{min}} = 8$	15.78%	14.96%
MAP, $\tau = 60$, $O_{\text{min}} = 8$	15.63%	14.81%
MAP, $\tau = 100$, $O_{\text{min}} = 8$	11.91%	10.54%
MAP, $\tau = 5$, $O_{\text{min}} = 4$	15.33%	13.45%
MAP, $\tau = 10$, $O_{\text{min}} = 4$	14.74%	12.95%
MAP, $\tau = 20$, $O_{\text{min}} = 4$	13.98%	12.22%
MAP, $\tau = 30$, $O_{\text{min}} = 4$	13.61%	11.83%
MAP, $\tau = 40$, $O_{\text{min}} = 4$	13.76%	11.93%
MAP, $\tau = 60$, $O_{\text{min}} = 4$	13.49%	12.02%
MAP, $\tau = 100$, $O_{\text{min}} = 4$	13.46%	12.69%
MAP, $\tau = 5$, $O_{\text{min}} = 1$	14.67%	12.68%
MAP, $\tau = 10$, $O_{\text{min}} = 1$	13.57%	11.55%
MAP, $\tau = 20$, $O_{\text{min}} = 1$	12.76%	10.47%
MAP, $\tau = 30$, $O_{\text{min}} = 1$	12.52%	10.23%
MAP, $\tau = 40$, $O_{\text{min}} = 1$	11.98%	10.50%
MAP, $\tau = 60$, $O_{\text{min}} = 1$	11.91%	10.54%
MAP, $\tau = 100$, $O_{\text{min}} = 1$	12.55%	11.35 %

Table E.1: A comparison of letter error rates for SG systems after MAP-reestimation.

Adaptation method	GMM PLP	GMM LPCC
SI	11.76%	13.22%
MAP, $\tau = 5$, $O_{\min} = 8$	12.08%	13.22%
MAP, $\tau = 10$, $O_{\min} = 8$	12.10%	13.22%
MAP, $\tau = 20$, $O_{\min} = 8$	12.15%	13.16%
MAP, $\tau = 30$, $O_{\min} = 8$	10.29%	8.33%
MAP, $\tau = 40$, $O_{\min} = 8$	12.20%	13.23%
MAP, $\tau = 60$, $O_{\min} = 8$	12.27%	13.18%
MAP, $\tau = 100$, $O_{\min} = 8$	10.39%	9.85%
MAP, $\tau = 5$, $O_{\min} = 4$	11.23%	11.92%
MAP, $\tau = 10$, $O_{\min} = 4$	11.13%	11.95%
MAP, $\tau = 20$, $O_{\min} = 4$	11.01%	11.99%
MAP, $\tau = 30$, $O_{\min} = 4$	11.18%	12.17%
MAP, $\tau = 40$, $O_{\min} = 4$	11.35%	12.34%
MAP, $\tau = 60$, $O_{\min} = 4$	11.77%	12.41%
MAP, $\tau = 100$, $O_{\min} = 4$	11.75%	12.34 %
MAP, $\tau = 5$, $O_{\min} = 1$	10.54%	8.87%
MAP, $\tau = 10$, $O_{\min} = 1$	10.29%	8.83%
MAP, $\tau = 20$, $O_{\min} = 1$	9.77%	8.94%
MAP, $\tau = 30$, $O_{\min} = 1$	9.80%	9.16%
MAP, $\tau = 40$, $O_{\min} = 1$	9.82%	9.43%
MAP, $\tau = 60$, $O_{\min} = 1$	10.39%	9.85%
MAP, $\tau = 100$, $O_{\min} = 1$	10.74%	10.83%

Table E.2: A comparison of letter error rates for GMM systems after MAP-reestimation.