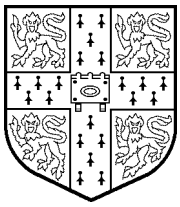
MPhil in Computer Speech and Language Processing

Department of Engineering, Cambridge University

Speaker Adaptation Using Eigenvoices



Robert Westwood

Wolfson College

August 31, 1999

Abstract

The thesis considers a novel technique for adaptation of speaker models, called *eigenvoice decomposition* (ED), based around reducing the dimension of the search space of acoustic models. The technique is compared both practically and theoretically with several other adaptation techniques.

The use of Principal Component Analysis to choose the subspace is discussed and evaluated. One published method of choosing a model within this space, Maximal Likelihood Eigenvoice Decomposition, is presented and compared with a new method, Weighted Projection.

Robust estimation of the subspace was found to be difficult, but when it could be performed, ED was found to give a significant improvement in recognition accuracy with only one adaptation sentence, rather than the five or so required by other methods.

Declaration

I declare that this thesis is substantially my own work. Where reference is made to the works of others the extent to which that work has been used is indicated and duly acknowledged in the text and bibliography.

Acknowledgements

I would like to express my appreciation to those who have assisted me in the preparation of this thesis. Firstly to my supervisor, Mr. Phil Woodland, for his advice; to Mr. Patrick Gosling for his willing computing support; to my fellow M.Phil students for their encouragement and expertise, and for their good humour and toleration of mine. Above all, thanks to him in whom all things have their beginning and their end.

> "I am the Alpha and Omega, the Beginning and the End ... who is and who was and who is to come, the Almighty." *Revelation 1v8, The Bible*

Contents

Chapter 1

Introduction

1.1 Context

There are many applications for speech recognition where the system will be used by a limited range of speakers only; for example a desktop dictation system (voice-operated typewriter), or a voice control system to be used by a specific group of people. However, increasingly it is necessary for a system to be deployed where it will be used by a large number of persons, for example a telephony dialogue system. In such situations it has been normal to use only one, *speaker-independent model* in order to model the speech regardless of the person making the utterance.

The word error rate afforded by using such a system is rather worse than if a *speaker-dependent model* was employed, typically by a factor of 2-3 [14]. However, it is impractical to obtain a sufficient amount of data to train a speaker-dependent model for each speaker, even if all speakers are known *a priori*; hence the interest in *(speaker) adaptation* - transforming a speaker-independent model to a speaker-dependent one using less data than would be required to train the model directly, or alternatively, altering the input data stream to a speaker-independent representation based on a model of a speaker (*speaker normalisation*). If the amount of adaptation data is very small, then the adaptation is referred to as *rapid adaptation*, and is the topic of this thesis.

1.2 Scope

Within this thesis, I first overview the use of Hidden Markov Models [20] for acoustic modelling of speech signals and hence for speech recognition. Of the many methods available for speaker adaptation I review four; two 'classic' methods, *Maximum A-Posterior* re-estimation (MAP) and *Maximal Likelihood Linear Regression* (MLLR), and two others, *Regression Model Prediction* (RMP) and *Cluster Adaptive Training* (CAT) which attempt to incorporate prior knowledge of the relationship between parameters of acoustic models for improved adaptation.

Relationships between parameters are also employed in adaptation by *Eigenvoice Decomposition* (ED) [9], the subject of this thesis, and hence the review of RMP and CAT. The space of acoustic models forms a *vector space* [6]; normally an adapted model may lie anywhere in this space. ED forces the adapted model to lie in a subspace, the *eigenspace*, chosen using training data; the basis vectors for this space are called *eigenvoices*. Kuhn *et al* describe one method for choosing a model within the eigenspace, *Maximal Likelihood Eigenvoice Decomposition* (MLED) [10]. I describe this technique, and also a new technique for performing eigenvoice decomposition, *Weighted Projection*.

Kuhn *et al* tested the eigenvoice method using a letter recognition task. Here, a more demanding task was chosen to test the method's effectiveness: a 1000 word vocabulary continuous speech task using the Resource Management Corpus [18] employing up to 10 sentences of adaptation data, for which transcriptions were available. The results are compared with those obtained for MAP and MLLR on the same task.

Chapter 2

The Use Of HMMs For Acoustic Modelling

For the purposes of this thesis, the acoustic modelling of speech is performed using phone-level *Hidden Markov Models* (HMMs). There is one HMM per phone to be recognised; the acoustic model is formed from a set of such models. I here discuss the structure of such models.

A Hidden Markov Model is a generative model formed from a digraph of nodes, called *states*, one of which is *initial*, one *final*, and the rest *internal*, and arcs, called *transitions*. Every transition has a probability associated with it. In addition, every internal state has an associated *output (probability) distribution*, and are said to be *emitting*. The initial and final states are *non-emitting*, and so do not have an associated output distribution. We will employ the convention that the initial state is state 1, and the final state is state n, if there are n states in the model.

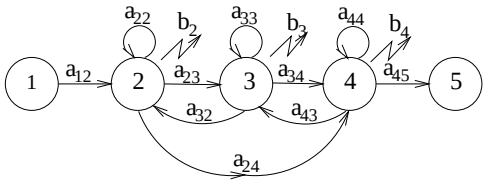


Figure 2.1: Structure Of A HMM

Suppose that at time t we are currently in state i. If we have a continuous probability distribution for the output (so our speech parameterisation is considered to be continuous), then the probability density of making an output of $\mathbf{o}_{(t+1)}$ from state j at time $t+1$ is $a_{ij} \times p(\mathbf{o}_{(t+1)}|\text{state} = j)$, where a_{ij} is the (fixed) probability of making the transition from state i to state j. We will write $p(\mathbf{o}_{(t+1)}|\text{state} = j)$ as $b_j(\mathbf{o}_{(t+1)})$; the value of this function when evaluated on a given $\mathbf{o}_{(t+1)}$ is the likelihood of observing $\mathbf{o}_{(t+1)}$ in state j. The likelihood of a series of observations $\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T$ given a particular path $s_0, s_1, s_2, \ldots, s_T$ through the model is given by multiplying the state and observation likelihoods together

$$p(\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T|s_0, s_1, \ldots, s_T) = \prod_{i=1}^{T-1} a_{s_i s_{i+1}} b_{s_{i+1}}(\mathbf{o}_{s_{i+1}}) \qquad (2.1)$$

Underlying this equation are *independence* assumptions that the transition probabilities and the observation likelihoods are independent over time and of the path

taken to a given state. It should be noted that the states have no physiological or phonetic significance, but merely model the change in parameters over time (in discrete steps), hence *hidden*.

A HMM is a *generative model* as it gives the probability of a series of observations given a path through the model ($\mathcal{M}$), as given by equation 2.1. However, in speech recognition, we have a series of observations, but no state sequence. Our aim is, given a set of HMMs and an observation sequence, to find the model which gives us the maximum likelihood of the observation, regardless of the state sequence. It would be possible to find the probability of the observation sequence by finding all possible paths through a model, and then calculating the likelihood for each path and either summing all probabilities or taking the highest likelihood. This is computationally infeasible due to the exponential number of paths that exist in a typical HMM. However, efficient algorithms exist for computing these values by dynamic programming; the algorithm for the first (total likelihood) is called the *Total Likelihood* algorithm, the second (highest likelihood) is called the *Viterbi* algorithm. As during recognition it is normally the Viterbi algorithm that is employed, we consider this first.

2.1 The Viterbi Algorithm

Let us define $s(t)$ to be the state occupied at time t, and have a series of observations $\mathbf{o}_1 \ldots \mathbf{o}_T =: \mathbf{O}$, and a function Φ as follows:

$$\begin{aligned} \Phi_j(t) &= \text{Likelihood of } s(t) = j \text{ and generating } \mathbf{o}_1 \ldots \mathbf{o}_t \\ &\quad \text{moving along most probable path} \qquad (2.2) \\ &= p(\mathbf{o}_1 \ldots \mathbf{o}_t, s(t) = j \text{moving along most probable path}) \qquad (2.3) \end{aligned}$$

The dynamic programming step is

$$\Phi_j(t) = \max_{1 \le i < N} (\Phi_i(t-1) a_{ij}) b_j(\mathbf{o}_t) \qquad (2.4)$$

with the boundary conditions

$$\begin{aligned} \Phi_1(0) &= 1 \\ \Phi_1(t) &= 0 \quad 0 < t \le T \\ \Phi_i(0) &= 0 \quad 1 < i \le N \end{aligned} \qquad (2.5)$$

After calculating $\Phi_N(T)$, the most likely path can be found by backtracking, if sufficient data is stored along with the values of $\Phi_j(t)$ during the dynamic programming calculations.

On average, the number of transitions into a state of a HMM is equal to the number of transitions leaving a state of a HMM. Using the Viterbi algorithm, only one of the possible paths into a state is propagated on to the output, so the number of paths is constant and equal to the number of states. Hence, the Viterbi algorithm is linear in both the number of states of the HMM and the length of the utterance.

These equations may be interpreted as extending and pruning partial paths through a HMM. For each time t there are $N-1$ paths active, with the likelihood of each given by $\Phi_j(t)$, where j is the state terminating the path. The dynamic programming step at time $t+1$ then extends the partial paths of up to time t, and chooses the best of those coming into each state.

The efficiency of the Viterbi decoding algorithm may be increased during recognition by a technique known as *pruning*. A partial path $s_0, s_1, \ldots, s_\tau$ through one of the models may have a sufficiently low likelihood that it is very unlikely that

the likelihood for any path having that partial path as its initial segment will be sufficiently high for it to be chosen as the best path through the model, and so can be discounted from the *search*. Moreover, all partial paths in a model up to a given time τ may have a sufficiently low likelihood that it is very unlikely that the model will be the one in the set to have the highest likelihood of the path, and so this model may then be discounted from the search. Removing a path or model from the search when it would have actually been selected is called a *search error*.

So far, we have only considered the case of recognition for a single phone. The HMMs for each phone can be concatenated *in parallel*, with the initial and final states for each model joined into a single start initial and a single final state. The Viterbi algorithm will choose the single most likely path through the combined model, indicating the most likely phone for the utterance. Again, to recognise a series of phones, the phones can be concatenated *in series*, collapsing the final state of model and the initial state of the next model to a single state. The Viterbi algorithm will give the likelihood for the most likely path through all models.

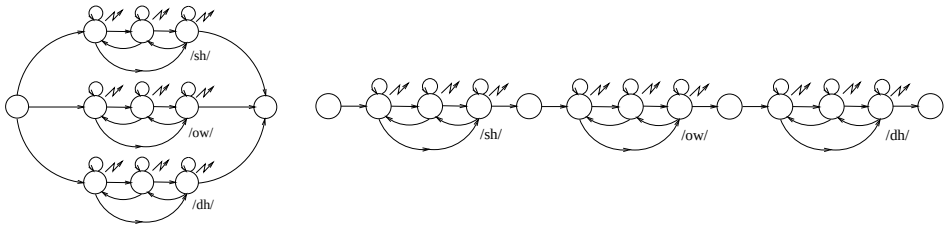


Figure 2.2: Parallel And Serial Connection Of HMMs

2.2 The Total Likelihood Algorithm

The equations for this method are much as per that for the Viterbi algorithm. We define the *forward probability* (actually likelihood) as follows

$$\begin{aligned} \alpha_j(t) &:= \text{Likelihood that s(t)=j and generating } \mathbf{o}_1 \ldots \mathbf{o}_t \\ &= p(\mathbf{o}_1, \ldots, \mathbf{o}_t, s(t) = j) \qquad (2.6) \\ &= \sum_{1 \le i < N} (\alpha i(t-1) a_{ij}) b_j(\mathbf{o}_t) \qquad (2.7) \end{aligned}$$

with the boundary conditions as defined for Φ. As for the Viterbi algorithm, paths merge and diverge at roughly the same rate, and so the Total Likelihood algorithm is also linear in both the number of states of the HMM and the length of the utterance. However, as we require all partial paths to each state for each time, we can not prune within the model as we can when doing Viterbi decoding, which has an impact on efficiency. Also, as there is no concept of one path through a model, but a probabilistic weighted sum of likelihoods over all paths, we can not concatenate the models together as we can with Viterbi decoding, and so the Total Likelihood algorithm is harder to use for continuous speech recognition.

2.3 Training The HMMs

Until now, we have been considering how to perform recognition using HMMs given that we already have the models. Even given the structure of a HMM, there are many parameters that need to be estimated, so we now consider how these parameters may be estimated.

The parameters of a HMM are estimated via the use of an *Expectation - Maximisation* (EM) Algorithm [2]. An EM algorithm consists of two steps, firstly an *Expectation Step* which transforms the observed (*incomplete*) data to a set of *sufficient statistics*, called the *complete* data, via an expectation operator. Given this complete data, the *maximisation step*, which estimates the parameters of the model via maximal likelihood. EM algorithms are generally iterative, as altering the parameters of the model will normally change the expectation of the complete data, so the maximisation step needs to be performed again. Convergence to a maximum of likelihood is guaranteed for a wide range of EM algorithms, although the speed of convergence may be slow [2].

Given a series of observations for a given HMM, we need to know the state sequence that gave rise to the observations. As the state sequence is hidden, it is not known. Thus our series of observation is incomplete, and our transformation to the complete data will be to append the (posterior) probabilities (likelihoods) of being in a particular state at a particular time, given the data. We denote this parameter γ_j where j is a state and t the time; we will refer to the parameter as the *frame alignment*.

Either the Viterbi or Total Likelihood methods of traversing a HMM can be used in the training step. In the case of the Viterbi algorithm, that traces through a single "best path", γ is a $\{0,1\}$-valued function; with Total Likelihood, which works on all paths, γ is a $[0,1]$-valued function. As it is usual to use Total Likelihood in HMM training, we will hereafter assume that method (see section 2.4); the Viterbi case is similar. The frame alignment may be calculated efficiently using the forward probabilities defined earlier, and an analogous quantity, $\beta_j(t)$, the backward probability.

$$\begin{aligned} \gamma_j(t) &= P(s(t)=j|\mathbf{O},\mathcal{M}) \\ &= \frac{\alpha_j(t)\beta_j(t)}{P(\mathbf{O}|\mathcal{M})} \end{aligned} \tag{2.8}$$

Where $\alpha_j(t)$ is as defined in 2.6 and

$$\beta_j(t) = P(\mathbf{o}_{t+1}\mathbf{o}_{t+2}\ldots\mathbf{o}_T|s(t)=j,\mathcal{M})$$

And the dynamic programming step is

$$\beta_j(t) = \left(\sum_{i=1}^{N-1}\beta_i(t+1)a_{ij}\right)b_j(t) \tag{2.9}$$

Given our complete data, we may perform a maximisation step. Re-estimation formulas for transition probabilities and the means & variances of the Gaussian output distributions may be derived from *Baum's auxiliary function* [20]. If the parameter vector of our model (set) $\mathcal{M}$ is $\boldsymbol{\lambda}$, and we wish to estimate the new parameter vector $\hat{\boldsymbol{\lambda}}$ for maximising the likelihood on the complete data (the *ML estimate*, then this is equivalent to maximising

$$Q(\boldsymbol{\lambda},\hat{\boldsymbol{\lambda}}) = \sum_{s\in\mathcal{M}} P(\mathbf{O},s|\boldsymbol{\lambda})\log P(\mathbf{O},s|\hat{\boldsymbol{\lambda}}) \tag{2.10}$$

over $\hat{\boldsymbol{\lambda}}$ as

$$Q(\boldsymbol{\lambda},\hat{\boldsymbol{\lambda}}) \geq Q(\boldsymbol{\lambda},\boldsymbol{\lambda}) \Rightarrow P(\mathbf{O}|\hat{\boldsymbol{\lambda}}) \geq P(\mathbf{O}|\boldsymbol{\lambda}) \tag{2.11}$$

Standard constrained optimisation techniques may be used to derive the re-estimation

formulae. For example, the re-estimation formula for the means is

$$\begin{aligned}\hat{\boldsymbol{\mu}}_j &= \frac{\text{Estimated sum of vectors emitted from state } j}{\text{Estimated number of vectors from state } j} \\ &= \frac{\sum_{t=1}^{T} \gamma_j(t)\mathbf{o}_t}{\sum_{t=1}^{T} \gamma_j(t)} \end{aligned} \tag{2.12}$$

These are normally calculated using *accumulators*. As a file is loaded, running totals for both the numerator and the denominator can be kept, meaning that the data does not need to be kept in memory. The denominator is called the zeroth order accumulator, denoted $A_0(j) := \sum_{t=1}^{T} \gamma_j(t)$, and the numerator the first order accumulator, denoted $A_1(j) := \sum_{t=1}^{T} \gamma_j(t)\mathbf{o}_t$.

2.4 Output Using Gaussian Mixtures

To date, we have assumed that the output is distributed as per a Gaussian distribution with fixed mean and variance. Other distributions, be they continuous or discrete are possible. Using Gaussian distributions is often convenient, and many properties are retained by using a *mixture of Gaussians* distribution.

A Gaussian distribution has a probability density function $p(\mathbf{o}) = \frac{1}{(2\pi)^{d/2}\sqrt{|C|}} \exp(-\frac{1}{2}(\mathbf{o}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{o}-\boldsymbol{\mu}))$, where $\boldsymbol{\mu}$ is the distribution mean, C is the distribution covariance matrix and d is the dimension of the observation vector $\mathbf{o}$. The mixture of Gaussians distribution has a probability density function of the form

$$p(\mathbf{o}) = \sum_{i=1}^{k} c_i \frac{1}{(2\pi)^{d/2}\sqrt{|C_i|}} \exp(-\frac{1}{2}(\mathbf{o}-\boldsymbol{\mu}_i)^T (C_i)^{-1}(\mathbf{o}-\boldsymbol{\mu}_i)) \tag{2.13}$$

with the constraint that the *mixture weights* c_i satisfy $\sum_{i=1}^{k} c_i = 1$; it is the weighted sum of k *mixture components*, each being distributed as per a Gaussian distribution. The advantage of using mixtures is due to the theorem that given an infinite number of Gaussian distributions any (continuous) probability density function can be modelled. Hence, if we have a sufficient number of mixtures, we can accurately model the speech distribution, regardless of the actual underlying distribution.

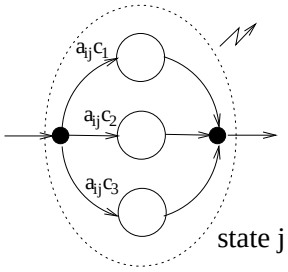


Figure 2.3: Considering Multiple Mixture Components As Separate States

With Baum-Welch (Total-Likelihood) training, as we traverse all paths, an output distribution that is mixture of Gaussians may be regarded as k separate, parallel states, with the mixture weights begin the transition probabilities, as in the diagram. Thus, in contrast to Viterbi training, there is no special training algorithm required.

2.5 Miscellanary

2.5.1 Context Dependency

So far, we have only been considering a single model for each phone. In speech, the properties of each phone vary widely according to the context in which it appears. To model this, we can create a separate model for a phone for each context in which it occurs. If the model is dependent on the phone immediately preceding and succeeding it, then it is called a *triphone*. Adding context dependency in this way vastly increases the number of models for which to estimate the parameters, for example if we have 45 phones, then there are $45 \times 45 \times 45 = 91125$ possible triphones.

2.5.2 Parameter Tying

To decrease the number of parameters, it is possible to *cluster* the models so that parameters are shared or *tied* between distributions. It is possible to tie at many levels, for example whole phone models could be tied, or at distribution level, but for the purposes of this thesis we will assume that it is *states* that are being tied. There are many ways that the tying may be performed; one of the commonest, and the one that will be employed is *tree-clustering*. Initially, all states are tied. A binary decision tree is made, each branch splitting the states into two sets, depending on context; the splits are made to maximise the likelihood of the training data. During training, the parameters for a state are estimated using the lowest node in the tree for which there is sufficient data to perform robust estimation.

Chapter 3

Performing Speaker Adaptation

As discussed in the introduction, we have two methods of adapting for differing speakers: *speaker normalisation*, where we try and map the input data stream to a speaker independent representation, or *model-based adaptation*, where the acoustic model is altered. Although speaker-normalisation may be shown to have beneficial effects, the lack of control at the phone-level means that there is a limit to the amount of inter-speaker variability that may be accommodated. We therefore concentrate on model-based adaptation techniques.

The exact parameters under which an adaptation technique is required to be effective will vary widely according to the application. The following indicate the major parameters that will be defined.

Quantity of Data The effectiveness of a given technique will depend on the amount of data presented to it; ideally we would wish for the technique to converge to the true speaker model given sufficient data, and for it to be effective on small quantities of data.

Supervised *vs.* Unsupervised If a transcription of the adaptation data is known, then the adaptation is said to be supervised, otherwise unsupervised.

Text Dependent *vs.* Text Independent Some techniques may be depend on the adaptation data being constant between each speaker; more generally useful are those which are not dependent are any particular input.

Static *vs.* Dynamic If all adaptation data is presented before the final adapted model is produced, then the technique is said to be a static or *block* adaptation technique, otherwise dynamic or *incremental*

For the purposes of this thesis, the adaptation techniques to be considered are all supervised, text independent, static techniques. As the subject being considered is *Rapid* Speaker Adaptation, the quantity of data to be considered will be reasonably small. With the ML estimation as per section 2.3, if a state is not visited ($\sum_t \gamma_j(t) = 0$), then there were be no data at all to estimate the parameters for the distribution; with a small amount of adaptation data there will be many states where this will hold. If $\sum_t \gamma_j(t) \neq 0$, we will not be able to form a robust estimate of the parameter. The basic problem then becomes how to exploit the data that we do have to form as robust estimates for all parameters. To do this, we can either employ *prior* knowledge of what we would expect the estimates to be, or we can use

the data of more than one state to estimate parameters that are shared; a technique known as data *pooling*. In practice, most techniques employ a combination of both.

To estimate the effectiveness of a technique, we examine the following criteria.

Effect on Recognition Accuracy Given a certain amount of adaptation data for a speaker, how does the recognition rate (word error rate) alter on test data for the same speaker.

Computational Efficiency Given the data can the adapted model by computed quickly and in a small amount of space.

It is also important to note how these criteria change with the amount of adaptation data available, and the number of parameters in the model.

3.1 MAP

Given adaptation data, it is possible to estimate a model for the speaker using the Forward-Backward algorithm as normal. However, many parameters will not be observed sufficiently often to form a robust estimate, if at all. However, the estimated parameters can be smoothed by forming a combination of the estimated parameter and a background prior parameter according to the number of times that parameter has been observed. This is, informally, the idea behind Maximum A-Posteriori estimation.

The name comes from the theory of Bayesian statistics. We are estimating the model parameters such that the likelihood of those parameters given the observation $\mathbf{O}$ is maximised; expressed mathematically using Bayes' rule, using our prior of how likely a model is, independently of the data observed,

$$p(\mathcal{M}|\mathbf{O}) = \frac{p(\mathbf{O}|\mathcal{M})p_p(\mathcal{M})}{p(\mathbf{O})} \tag{3.1}$$

where $p_p(\mathcal{M})$ is the prior probability of the model before observing any data. The derivation of the MAP equations [4] is beyond the scope of this thesis. For illustration, the MAP re-estimation formula for means with Normal output distributions is given by

$$\hat{\mu} = \frac{\tau\mu + \sum_{t=1}^{T}\gamma(t)\mathbf{o}_t}{\tau + \sum_{t=1}^{T}\gamma(t)} \tag{3.2}$$

where τ is a *meta-parameter* which gives the bias between the ML estimate of the mean from the data, and the prior mean. In theory, it would be possible to incorporate knowledge of the expected properties of the models via the prior. However, to derive the re-estimation formula for a general case is intractable, and so only knowledge as to the individual parameters is included, and the speaker independent parameters are used for the prior.

The big advantage of MAP estimation is that it is *convergent* in the following sense: suppose we have a stream of data such that the probability of observing data for every parameter is non-zero; then as the amount of data tends towards infinity, the model estimated by MAP estimation tends towards the true speaker-dependent model. However, the rate at which this convergence occurs is slow, meaning that MAP is not generally used when only small amounts of data are present. It also depends on having sufficient data for every parameter. In the case of models with large numbers of parameters, then some parameters are rarely seen, and so MAP will adapt these parameters poorly, even when a large quantity of data is available.

3.2 RMP

MAP incorporates the prior knowledge of the speaker-independent prior but makes no attempt to use data from states not currently under adaptation to improve the adaptation in states with low occupancy. An extension to MAP which does this is *Regression Model Prediction* (RMP) [1]. The technique uses linear regression to estimate the values of poorly adapted parameters from those that are well adapted, using regression relationships determined from training data. Although the technique can be used with models having a mixture of Gaussians output distribution, for reasons of clear notation we assume a single Gaussian distribution.

There is an off-line training step involved, in finding the linear regression relationships. A set of N speaker dependent models are trained. For each (target) distribution having mean $\boldsymbol{\mu}^{(s)'}$, we can find the K states $s_1, s_2, \ldots, s_K$ that are most correlated with it, and estimate the regression formula

$$\boldsymbol{\mu}^{(s)} \quad = \quad \mathbf{a}_0 + AM \tag{3.3}$$

where $M = (\ldots|\boldsymbol{\mu}^{(s_i)}|\ldots)$. A is here indicated as a full regression matrix, but in [1] it is taken to be diagonal.

Then to perform adaptation, MAP re-estimation is performed, giving the adapted mean vectors $\tilde{\boldsymbol{\mu}}^{(s)}$. For each distribution, the regression formula estimated by equation 3.3 is applied to the MAP adapted means to give another estimate of the mean vector for each distribution, $\check{\boldsymbol{\mu}}^{(s)}$. These two estimates are combined using another MAP formula [1], giving

$$\hat{\boldsymbol{\mu}}^{(s)} \quad = \quad \frac{\check{\boldsymbol{\mu}}^{(s)}(\tilde{s}^{(s)})^2 + \tilde{\boldsymbol{\mu}}^{(s)}(\check{s}^{(s)})^2}{(\check{s}^{(s)})^2 + (\tilde{s}^{(s)})^2} \tag{3.4}$$

where $(\check{s}^{(s)})^2$ is the variance associated with the initial MAP estimate and $(\tilde{s}^{(s)})^2$ is variance for the regression adapted mean. The better an estimate, the lower its variance will be. Thus, if there is sufficient data for the initial MAP estimate of a parameter (giving a low variance) but the regression estimate is poor (and so has a high variance), then the numerator of equation 3.4, and hence the estimate, will be dominated by the first term, and vice-versa.

RMP has the same advantage of MAP, in that it is convergent. The regression of unseen or poorly adapted parameters means that the model as a whole will be closer to the true model on less data. However, individual parameters will converge no more quickly than for MAP. In addition, finding the most correlated parameters for the off-line regression calculation step is a very slow step: Ahadi *et al* estimate that for a 2 mixture word-internal 3-state HMM triphone system, approximately 350 million relationships must be searched. For the on-line efficiency, please see section 4.2.5.

3.3 MLLR

The next technique to be considered is Maximal Likelihood Linear Regression (MLLR) [14]. For a given state (mean), we can estimate the transform required to transform the mean vector $\boldsymbol{\mu}$ to the estimate of the adapted mean vector $\hat{\boldsymbol{\mu}}$, as follows

$$\hat{\boldsymbol{\mu}} \quad = \quad A\boldsymbol{\mu} + \mathbf{b} \tag{3.5}$$

or alternatively

$$\hat{\boldsymbol{\mu}} \quad = \quad W\boldsymbol{\xi} \tag{3.6}$$

where $\boldsymbol{\xi}$ is the extended mean vector $(\boldsymbol{\mu}^T|1)^T$ and W is the transform matrix. The transform matrix is estimated so as to give the maximal likelihood of the adaptation data for that distribution. Having a separate transformation matrix for each distribution does not decrease the number of parameters to be estimated; indeed, it will probably increase it. However, it is possible to estimate a single transformation for several distributions, and pool all data seen for these distributions to estimate the transformation matrix. This enables MLLR to adapt parameters which have not been observed in the adaptation data.

To decide on how data is to be pooled, a *Regression Class Tree* may be formed. A binary decision tree is constructed based upon training data; every node splits into branches depending on the question of acoustic context, for example "Is the current phone a nasal?" or "Is the right phone a liquid?". At every node in the tree, a transform is estimated for all distributions below it in the tree; if there is insufficient data to estimate a transform, then the tree is pruned at that point. For any given distribution, the transform that is the lowest in the tree (with sufficient data to estimate the transform) is employed.

The number of parameters in each transform is dependent on the type of transform employed; it is not necessary to estimate the full $n \times n + 1$ parameters in W. One of the most common forms of reduced matrices is *block diagonal*, as follows

$$W = \begin{pmatrix} W_0 & 0 & \ldots\ldots & \mathbf{b}_0 \\ 0 & W_1 & 0 \ \ldots & \mathbf{b}_1 \\ \ldots\ldots\ldots\ldots\ldots\ldots & & & \\ \ldots\ldots\ldots & 0 & W_n & \mathbf{b}_n \end{pmatrix} \tag{3.7}$$

where the vector of constant offsets, $\mathbf{b} = (\mathbf{b}_o^T|\mathbf{b}_1^T|\ldots|\mathbf{b}_n^T)^T$.

The use of tying transformations together via a regression tree means, that unlike MAP adaptation, all parameters may be adapted. However, sufficient data must be available to robustly estimate the root transformation (which will act as a global transform), before MLLR may be applied. Taking typical speech parameterisation of 39 dimensions, and a block matrix such as W above with $n = 3$, this means that 546 parameters need to be estimated for the transform. The need to estimate these parameters bounds MLLR to applications where there is sufficient data to robustly estimate this number of parameters.

If the number of regression classes is equal to the number of distributions, then MLLR is convergent in the same sense as MAP. However, having so many transforms to estimate results in a very large number of parameters to estimate; more than using MAP or direct ML estimate. Hence, a more restricted number of regression classes are used, reducing the number of parameters to estimate, but losing the property of convergence. To regain such, MLLR may be combined with MAP. When only a finite amount of data is present, MLLR will perform a coarser adaptation on the entire model, then the MAP re-estimation will refine those parameters for which sufficient data exists.

3.4 CAT

The final adaptation technique we consider is *Cluster Adaptive Training* (CAT) [3]. One method of speaker adaptation is *speaker clustering*, whereby, using training data, a set of *cluster models* are produced, where each model represents a cluster of speakers that are, in some sense, 'similar'. Given adaptation data, the canonical model which is estimated to be 'best' for the new adaptation data is selected. CAT extends this procedure, so that instead of picking one model, a linear combination of the canonical models is estimated.

Given a set of N speaker-dependent models estimated from training data, we form a set of K cluster models by the following algorithm

1. Estimate the values of the weight vectors for each speaker in the training data, given the current estimate of the cluster parameters

2. Estimate a new set of cluster models given the weight vector

continuing the iteration until some convergence criterion is satisfied.

Denote the mean of the i^{th} cluster model for mixture component m of state s as $\mathbf{e}_m^{(s)}(i)$. Then the vector of weights $\mathbf{w}$ can be determined by solving the linear equation

$$G_w^{(r)}\mathbf{w} = \mathbf{k}_w^{(r)} \tag{3.8}$$

where

$$G_w^{(r)} = \sum_{s\in\mathcal{M}}\sum_{m\in s}\sum_t \gamma_m^{(s)}(t) {M_m^{(s)}}^T {C_m^{(s)}}^{-1} M_m^{(s)} \tag{3.9}$$

$$\mathbf{k}_w^{(r)} = \sum_{s\in\mathcal{M}}\sum_{m\in s} {M_m^{(s)}}^T C_m^{(s)} \sum_t \gamma_m^{(s)}(t)\mathbf{o}_t \tag{3.10}$$

$$M_m^{(s)}(r) = \left(\mathbf{e}_m^{(s)}(1)|\ldots|\mathbf{e}_m^{(s)}(K)\right) \tag{3.11}$$

It should be noted from the equations that every observation vector is considered in the computation of every weight as the sums run over every mixture component in the system and all time of the observations. One would therefore expect the technique to be particularly suitable for rapid speaker adaptation. In the paper presenting the technique, ([3]), the technique is evaluated using 20 sentences, which is more than we consider here. It should also be noted that the technique is not convergent.

Chapter 4

Eigenvoices

Intuitively, one would expect that the family of speaker models to be of far lower dimensionality than the many thousands employed in even a simple acoustic model. If a more systematic characterisation of speech could be found, it would have an impact on speaker adaptation, as the number of parameters to be estimated would be greatly reduced, and so could be more robustly estimated, while still capturing the essential variation between speakers.

One such attempt at finding a characterisation is that of *eigenvoices*, first published by Juhn, Nguyen *et al* ([10] & [16]) and modelled on earlier work on human face analysis [9]. The eigenvoices form a basis of a subspace of the acoustic model space, and are chosen to account for inter-speaker variability.

Suppose that we wish to adapt a certain set of parameters in a model. These parameters form a *supervector* in the space of acoustic models, the $\mathcal{D}$-space. Previously, using training data, a subspace of the $\mathcal{D}$-space, the $\mathcal{K}$-space, spanned by K eigenvoices, $\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_{K-1}$, has been estimated. Given the adaptation data, we estimate a set of weights, $w_0, w_1, \ldots, w_{K-1}$ to form a model in the $\mathcal{K}$-space given by $\sum w_i \mathbf{e}_i$, which is our adapted model. This process of adaption is called *eigenvoice decomposition* (ED). Kuhn *et al* used an EM-based maximal likelihood algorithm, *Maximal Likelihood Eigenvoice Decomposition* (MLED) to perform the decomposition. This method is discussed shortly, and then a further algorithm, *Weighted Projection*, developed for this thesis, is presented.

4.1 Estimating The Subspace

Before performing ED, it is necessary to have estimated the $\mathcal{K}$-space. We can train a model for each speaker in the training data, and extract a supervector from each. These supervectors are then analysed to choose a basis. There are many techniques which may be used, but to to data *Principal Component Analysis* (PCA) has been employed.

4.1.1 Principal Component Analysis

Suppose that we have n variables $x_0, \ldots, x_{n-1}$. Define the *first principal component*, ξ_1, to be the linear combination of the x_j such that the variance is maximised, and define the i^{th} *principal component*, ξ_i $(i > 1)$, to be the linear combination of the x_j such that the variance is maximised *and* ξ_i is uncorrelated with $\xi_1, \ldots, \xi_{i-1}$. It may be shown [8] that if the i^{th} eigenvector of $\Sigma = (\boldsymbol{\sigma}_0 | \ldots | \boldsymbol{\sigma}_{n-1})$, the covariance matrix of the x_j, $\mathbf{v}_i = \sum_{k=0}^{n-1} \alpha_k^{(i)} \mathbf{c}_k$, then $\xi_i = \sum_{k=0}^{n-1} \alpha_k^{(i)} x_k$, if the eigenvectors are ordered according to the magnitude of the corresponding eigenvalue.

Although we do not have a set of random variables, it is possible to do PCA in a more data-driven manner [5]. From the training data, we have estimated n supervectors of parameters, $\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}$, each of dimension N and can thus estimate the $N \times N$ covariance matrix. Given the covariance matrix, we can find its eigenvectors, $\mathbf{v}_0, \ldots, \mathbf{v}_{N-1}$, where $\mathbf{v}_i = \sum_{k=0}^{n} \alpha_k^{(i)} \mathbf{x}_k$. Our principal components are then $\xi_i = \sum_{k=0}^{n} \alpha_k^{(i)} \mathbf{x}_i$.

Intuitively, what we are doing is rotating the space (the matrix of eigenvectors of Σ being orthogonal) so that the basis vectors are aligned with the directions where there is most variability. Hence, to capture the most variability with just a K-dimensioned space, we take the K principal components, the first being that associated with the largest eigenvalue. Sometimes, the mean of all the observed supervectors is used as the first component, and then the first $K-1$ principal components used for the basis. Although PCA is defined in terms of the covariance matrix, other cross-product matrices may be used, for example the correlation matrix, or XX^T, where $X := (\ldots|\mathbf{x}_i|\ldots)$ [5].

The number of parameter is the model, N, is typically large: anything from a few thousand to over a million. To compute the entire covariance matrix, the size of which is quadratic in the number of parameters, is infeasible for models with more than about 10,000 parameters. However, due to a theorem about *Singular Value Decomposition* (SVD), if we use XX^T as our cross product matrix, then we can perform PCA using much less space. SVD decomposes a matrix into a product of three matrices:

$$\underset{N \times n}{X} = \underset{N \times r}{P} \quad \underset{r \times r}{\Lambda} \quad \underset{r \times n}{Q^T} \tag{4.1}$$

where Λ is a diagonal matrix, and P & Q are unitary matrices. The theorem states that P is the matrix of eigenvectors of XX^T corresponding to the r largest eigenvalues, which are ordered along the diagonal of Λ [5]. As SVD can be performed 'in-place' for the matrices X, P, Λ and Q, the space required is now linear in the number of parameters of the model.

There is often a figure quoted called the *proportion of variability* for a given basis vector. This is the ratio of the eigenvalue for the given eigenvector to the sum of the eigenvalues of the matrix, and is a measure of the proportion of variability in the input accounted for in the given direction [7].

4.2 Eigenvoice Decomposition

4.2.1 Maximal Likelihood Eigenvector Decomposition

Maximal Likelihood Eigenvector Decomposition (MLED) is an application of the *Expectation-Maximisation* (EM) algorithm [2]. Informally, it selects a weight vector $\mathbf{w}$ such that if $E := (\mathbf{e}_0|\mathbf{e}_1|\ldots|\mathbf{e}_{K-1})$, then $E\mathbf{w}$ is the supervector such that when it is recombined to form a model, the likelihood of the observed data is maximised.

The derivation of the equations follows a standard derivation for EM applications, as indicated for Baum-Welch re-estimation in section 2.3; a full derivation is given in [16]. We are aiming to choose a vector of model parameters $\boldsymbol{\lambda}$ such that the likelihood of the observed data $\mathbf{O}$ is maximised. We form our complete data set by calculating the frame alignment (equation 2.8). As per the Baum-Welch derivation we maximise an auxiliary function, Q. In contrast to normal maximal likelihood estimation of model parameters, we will constrain $\boldsymbol{\lambda}$ (and $\hat{\boldsymbol{\lambda}}$) to lie in the eigenspace.

Q is defined by

$$Q(\boldsymbol{\lambda}, \hat{\boldsymbol{\lambda}}) = -\frac{1}{2} P(\mathbf{O}|\mathcal{M}_{\boldsymbol{\lambda}}) \sum_{s \in \mathcal{M}_{\boldsymbol{\lambda}}} \sum_{m \in s} \sum_{t=1}^{T} \Big(\gamma_m^{(s)}(t) \big(n \log(2\pi) + \log |C_m^{(s)}| + h(\mathbf{o}_t, s) \big) \Big) \tag{4.2}$$

where

$$h(\mathbf{o}_t, s) = (\mathbf{o}_t - \hat{\boldsymbol{\mu}}_m^{(s)})^T {C_m^{(s)}}^{-1} (\mathbf{o}_t - \hat{\boldsymbol{\mu}}_m^{(s)})$$

As we are only adapting means, the constraint that $\hat{\boldsymbol{\lambda}}$ lies in the eigenspace is that $\hat{\boldsymbol{\mu}}$ is a linear combination of the eigenvoices, that is

$$\hat{\boldsymbol{\mu}} = \sum_{j=0}^{K-1} w_j \mathbf{e}_j \tag{4.3}$$

Therefore, we need to maximise Q with respect to the w_js. At maxima,

$$\frac{\partial Q}{\partial w_i} = 0 \qquad i = 0, 1, \ldots, E-1 \tag{4.4}$$

and hence

$$\frac{\partial Q}{\partial w_i} = 0 = \sum_{s \in \mathcal{M}_{\boldsymbol{\lambda}}} \sum_{m \in s} \sum_{t=1}^{T} \Big(\frac{\partial}{\partial w_j} \gamma_m^{(s)}(t) h(\mathbf{o}_t, s) \Big) \qquad j = 0, 1, \ldots, E-1 \tag{4.5}$$

as $\frac{\partial w_i}{\partial w_j} = 0$ for $i \neq j$ as the eigenvoices are orthogonal. The computation of the derivative is given in [16]. The equation with the derivative computed is given below. Writing $\mathbf{e}_i$ as $\mathbf{e}(i)$ for clarity,

$$0 = \sum_{s \in \mathcal{M}_{\boldsymbol{\lambda}}} \sum_{m \in s} \sum_{t=1}^{T} \Big(\gamma_m^{(s)}(t) \big(-\mathbf{e}_m^{(s)}(i)^T {C_m^{(s)}}^{-1} \mathbf{o}_t + \sum_{j=0}^{E-1} w_j \mathbf{e}_m^{(s)}(j)^T {C_m^{(s)}}^{-1} \mathbf{e}_m^{(s)}(i) \big) \Big) \qquad i = 0, 1, \ldots, K-1 \tag{4.6}$$

Rearranging gives

$$\sum_{s \in \mathcal{M}_{\boldsymbol{\lambda}}} \sum_{m \in s} \sum_{t=1}^{T} \gamma_m^{(s)}(t) \mathbf{e}_m^{(s)}(i)^T {C_m^{(s)}}^{-1} \mathbf{o}_t = \sum_{s \in \mathcal{M}_{\boldsymbol{\lambda}}} \sum_{m \in s} \sum_{t=1}^{T} \gamma_m^{(s)}(t) \sum_{i=0}^{E-1} w_j \mathbf{e}_m^{(s)}(j)^T {C_m^{(s)}}^{-1} \mathbf{e}_m^{(s)}(i) \qquad i = 0, 1, \ldots, K-1 \tag{4.7}$$

which is a set of K linear equations in K unknowns.

We can rewrite this equation in matrix notation as

$$Q\mathbf{w} = \mathbf{v} \tag{4.8}$$

where

$$Q := (q_{ij})$$

$$q_{ij} := \sum_{s \in \mathcal{M}_{\boldsymbol{\lambda}}} \sum_{m \in s} \sum_{t=1}^{T} \gamma_m^{(s)}(t) \mathbf{e}_m^{(s)}(j)^T {C_m^{(s)}}^{-1} \mathbf{e}_m^{(s)}(i) \tag{4.9}$$

$$\mathbf{v} := \Big(\ldots | \sum_{s \in \mathcal{M}_{\boldsymbol{\lambda}}} \sum_{m \in s} \sum_{t=1}^{T} \gamma_m^{(s)}(t) \mathbf{e}_m^{(s)}(i)^T {C_m^{(s)}}^{-1} \mathbf{o}_t | \ldots \Big)^T \tag{4.10}$$

4.2.2 Projection Techniques

Instead of using Maximal Likelihood to estimate the combination of eigenvoices, it is possible to use *projection* to estimate the vector in $\mathcal{K}$-space. That is, we find the vector $\hat{\boldsymbol{\mu}}$ such that $||\hat{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}}||$ is minimised, where $\tilde{\boldsymbol{\mu}}$ is the ML estimate of the means from the adaptation data; equivalently we find a vector $\hat{\boldsymbol{\mu}}$ such that $\forall \boldsymbol{\mu}' \in \mathcal{K}, ||\hat{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}}|| \leq ||\boldsymbol{\mu}' - \tilde{\boldsymbol{\mu}}||$. For this equation to make sense, we need an estimate for all parameters of the estimated mean vector $\tilde{\boldsymbol{\mu}}$. This is a major obstacle for the use of projection, and the reason that it is discounted by Kuhn *et al* in [10].

Instead of performing the projection in the $\mathcal{D}$-space directly, we can apply a linear transformation such that in the transformed space, where unestimated parameters are given a value 0, and hence have no effect. Moreover, we can select our transform such that parameters that we expect to be robustly estimated have more effect on the choice of $\hat{\boldsymbol{\mu}}$. Thus, our aim is to minimise $||T\hat{\boldsymbol{\mu}} - T\tilde{\boldsymbol{\mu}}||$, where T is the linear transformation.

We assume that the robustness of the estimate is directly related to the amount of data used to estimate it; the quantity of data used is given by the zeroth order accumulator. Therefore, let us define a function $\omega : \mathbb{R} \rightarrow \mathbb{R}$ to map the accumulator to a weight for that parameter to be a *weight function* if the following properties hold:

1. $\omega(0) = 0$
2. ω is (non-strictly) monotonically increasing

We can then define our linear transform T to be the composition of two transforms, the first being a decorrelation of the space, D, which is not necessarily required, but will reduce the weighting in favour of distributions with large variances, followed by a weighting according to the weight function ω, Ω; that is, $T = \Omega D$. D will be a block diagonal matrix, as we do not measure the covariance between distributions; candidates for each block on the diagonal include I, to not decorrelate, $C_m^{(s)^{-1/2}}$ for a component-by-component decorrelation, or $\overline{C}^{-1/2}$, where $\overline{C}$ is the average of the component-by-component covariance matrices, which will provide more robustness in the case of the covariance matrices being badly estimated. Our weighting transformation will be diagonal, but it is probably easier to consider it as being block diagonal, and defined by

$$\begin{pmatrix} \ddots & 0 & 0 \\ 0 & \omega(A_0(s,m))^{1/2} I_d & 0 \\ 0 & 0 & \ddots \end{pmatrix}$$

where d is the dimensionality of the speech vector. We also require that $\forall \mathbf{x} \in \mathcal{K}.T(\mathbf{x}) = 0 \Rightarrow \mathbf{x} = 0$. This is sufficient to ensure that the image of the basis vectors of $\mathcal{K}$ form a basis for $T(\mathcal{K})$. Theorem 1, adapted from a standard linear algebra proof for this thesis, states that given T, an estimate that minimises the square error in the projected space exists.

Theorem 1
Let $E = (\mathbf{e}_0|\mathbf{e}_1|\ldots|\mathbf{e}_{K-1})$. *Then there exists a vector* $\mathbf{w}$ *such that* $(TE)^T(TE)\mathbf{w} = (TE)^T T\tilde{\boldsymbol{\mu}})$. *Moreover,* $E\mathbf{w}$ *satisfies the property* $||T\tilde{\boldsymbol{\mu}} - TE\mathbf{w}|| \leq ||T\tilde{\boldsymbol{\mu}} - T\boldsymbol{\mu}'|| \quad \forall \boldsymbol{\mu}' \in \mathcal{K}$.

Proof.
We first justify our earlier claim that $\{T\mathbf{e}_0, T\mathbf{e}_1, \ldots, T\mathbf{e}_{K-1}\}$ is a basis for $T(\mathcal{K})$.

As T is linear, $\{T\mathbf{e}_0, T\mathbf{e}_1, \ldots, T\mathbf{e}_{K-1}\}$ span $T(\mathcal{K})$. It only remains to show that they are linearly independent. Take $\alpha_0, \alpha_1, \ldots, \alpha_{K-1}$ such that

$$\begin{aligned}
\alpha_0 T\mathbf{e}_0 + \alpha_1 T\mathbf{e}_1 + \ldots + \alpha_{K-1} T\mathbf{e}_{K-1} &= \mathbf{0} \\
T(\alpha_0\mathbf{e}_0 + \alpha_1\mathbf{e}_1 + \ldots + \alpha_{K-1}\mathbf{e}_{K-1}) &= \mathbf{0} \\
\Rightarrow \quad \alpha_0\mathbf{e}_0 + \alpha_1\mathbf{e}_1 + \ldots + \alpha_{K-1}\mathbf{e}_{K-1} &= \mathbf{0} \\
\Rightarrow \alpha_i &= 0 \qquad i = 0, 1, \ldots, K-1
\end{aligned}$$

as $\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_{K-1}$ are linearly independent. Therefore $\{T\mathbf{e}_0, T\mathbf{e}_1, \ldots, T\mathbf{e}_{K-1}\}$ are linearly independent and span $T(\mathcal{K})$, so form a basis for that space, as required.

To show that $\mathbf{w}$ exists, it is sufficient to prove that $(TE)^T(TE)$ is invertible. Let $\mathbf{q}$ be a vector such that $(TE)^T(TE)\mathbf{q} = \mathbf{0}$. Then

$$\begin{aligned}
(TE\mathbf{q})^T(TE\mathbf{q}) &= \mathbf{q}^T(TE)^T(TE)\mathbf{q} \\
&= \mathbf{q}^T\mathbf{0} \\
&= \mathbf{0} \\
\Rightarrow TE\mathbf{q} &= \mathbf{0}
\end{aligned}$$

Suppose $\mathbf{q} = (q_0, q_1, \ldots, q_{K-1})$. Then

$$\begin{aligned}
\mathbf{0} = TE\mathbf{q} &= q_0 T\mathbf{e}_0 + q_1 T\mathbf{e}_1 + \ldots + q_{K-1} T\mathbf{e}_{K-1} \\
\Rightarrow q_i &= 0 \quad i = 0, 1, \ldots, K-1
\end{aligned}$$

as $\{T\mathbf{e}_0, T\mathbf{e}_1, \ldots, T\mathbf{e}_{K-1}\}$ form a basis. Therefore $(TE)^T TE\mathbf{q} = 0 \Rightarrow \mathbf{q} = 0$ and so $(TE)^T(TE)$ is invertible, and consequently $\mathbf{w}$ exists.

Note that $E\mathbf{w}$ is a member of $\mathcal{K}$. Write $\hat{\boldsymbol{\mu}} := E\mathbf{w}$. We need to show that $\|T\tilde{\boldsymbol{\mu}} - T\hat{\boldsymbol{\mu}}\| \leq \|T\tilde{\boldsymbol{\mu}} - T\boldsymbol{\mu}'\| \quad \forall \boldsymbol{\mu}' \in \mathcal{K}$. We first show that $(T\tilde{\boldsymbol{\mu}} - T\hat{\boldsymbol{\mu}})^T T\boldsymbol{\mu}' = 0 \quad \forall \boldsymbol{\mu}' \in \mathcal{K}$. By definition of $\mathbf{w}$,

$$\begin{aligned}
(TE)^T TE\mathbf{w} &= (TE)^T T\tilde{\boldsymbol{\mu}} \\
(TE)^T T\hat{\boldsymbol{\mu}} &= (TE)^T T\tilde{\boldsymbol{\mu}} \\
(TE)^T(T\hat{\boldsymbol{\mu}} - T\tilde{\boldsymbol{\mu}}) &= \mathbf{0} \\
(T\hat{\boldsymbol{\mu}} - T\tilde{\boldsymbol{\mu}})^T(TE) &= \mathbf{0}^T \\
\Rightarrow (T\hat{\boldsymbol{\mu}} - T\tilde{\boldsymbol{\mu}})^T T\mathbf{e}_i &= 0 \qquad i = 0, 1, \ldots, K-1 \\
\Rightarrow (\hat{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}})^T T\boldsymbol{\mu}' &= 0 \qquad \forall \boldsymbol{\mu}' \in \mathcal{K}
\end{aligned}$$

as $\{\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_{K-1}\}$ is a basis for $\mathcal{K}$.

Consider $\boldsymbol{\mu}' \in \mathcal{K}$.

$$\begin{aligned}
\|T\tilde{\boldsymbol{\mu}} - T\boldsymbol{\mu}'\|^2 &= \|(T\tilde{\boldsymbol{\mu}} - T\hat{\boldsymbol{\mu}}) + (T\hat{\boldsymbol{\mu}} - T\boldsymbol{\mu})\|^2 \\
&= (T\tilde{\boldsymbol{\mu}} - T\hat{\boldsymbol{\mu}})^T(T\tilde{\boldsymbol{\mu}} - T\hat{\boldsymbol{\mu}}) + \\
&\qquad 2(T\tilde{\boldsymbol{\mu}} - T\hat{\boldsymbol{\mu}})^T(T\hat{\boldsymbol{\mu}} - T\boldsymbol{\mu}') + \\
&\qquad (T\hat{\boldsymbol{\mu}} - T\boldsymbol{\mu}')^T(T\hat{\boldsymbol{\mu}} - T\boldsymbol{\mu}') \\
&= \|T\tilde{\boldsymbol{\mu}} - T\hat{\boldsymbol{\mu}}\|^2 + \|T\hat{\boldsymbol{\mu}} - T\boldsymbol{\mu}'\|^2 \\
\|T\tilde{\boldsymbol{\mu}} - T\boldsymbol{\mu}'\| &\geq \|T\tilde{\boldsymbol{\mu}} - T\hat{\boldsymbol{\mu}}\| \qquad (4.11)
\end{aligned}$$

Therefore, $\hat{\boldsymbol{\mu}}$ is the vector in $\mathcal{K}$, such that the square error between that and the estimate of the parameter vector $\tilde{\boldsymbol{\mu}}$ *in the space when transformed by* T is the least of all vectors in $\mathcal{K}$.

Theorem 1 gives a matrix equation that we can use to estimate the weight vector $\mathbf{w}$ and hence the re-estimate of the mean vector, $\hat{\boldsymbol{\mu}}$, namely

$$\begin{aligned}
(TE)^T(TE)\mathbf{w} &= (TE)^T T\tilde{\boldsymbol{\mu}} \qquad (4.12) \\
E^T T^T TE\mathbf{w} &= E^T T^T T\tilde{\boldsymbol{\mu}} \qquad (4.13)
\end{aligned}$$

Both the left and right hand sides have a common subexpression of $T^T T$. Considering this expression gives the following derivation, noting that both the identity matrix and a decorrelation matrix are invariant under transpose.

$$\begin{aligned}
T^T T &= D^T \Omega^T \Omega D \\
&= \begin{pmatrix} \ddots & 0 & 0 \\ 0 & D_m^{(s)} & 0 \\ 0 & 0 & \ddots \end{pmatrix} \begin{pmatrix} \ddots & 0 & 0 \\ 0 & \omega(A_0(s,m))^{1/2} I_d & 0 \\ 0 & 0 & \ddots \end{pmatrix} \\
&\quad \begin{pmatrix} \ddots & 0 & 0 \\ 0 & \omega(A_0(s,m))^{1/2} I_d & 0 \\ 0 & 0 & \ddots \end{pmatrix} \begin{pmatrix} \ddots & 0 & 0 \\ 0 & D_m^{(s)} & 0 \\ 0 & 0 & \ddots \end{pmatrix} \\
&= \begin{pmatrix} \ddots & 0 & 0 \\ 0 & D_m^{(s)} \omega(A_0(s,m)) I_d D_m^{(s)} & 0 \\ 0 & 0 & \ddots \end{pmatrix} \\
&= \begin{pmatrix} \ddots & 0 & 0 \\ 0 & \omega(A_0(s,m)) {D_m^{(s)}}^2 & 0 \\ 0 & 0 & \ddots \end{pmatrix} =: B
\end{aligned} \tag{4.14}$$

Then the right hand-side of equation 4.12 can be written

$$\begin{aligned}
\mathbf{v} &:= (TE)^T T \tilde{\boldsymbol{\mu}} = E^T B \tilde{\boldsymbol{\mu}} \\
v_i &= \mathbf{e}(i)^T B \tilde{\boldsymbol{\mu}} \\
&= \sum_{s \in \mathcal{M}} \sum_{m \in s} \mathbf{e}_m^{(s)}(i)^T \omega(A_0(s,m)) D_m(s)^2 \tilde{\boldsymbol{\mu}} \\
&= \sum_{s \in \mathcal{M}} \sum_{m \in s} \omega(A_o(s,m)) \mathbf{e}_m^{(s)}(i)^T D_m(s)^2 \tilde{\boldsymbol{\mu}}_m^{(s)}
\end{aligned} \tag{4.15}$$

And similarly the left hand side can be written as

$$\begin{aligned}
E^T T^T T E \mathbf{w} &= E^T B E \mathbf{w} =: Q\mathbf{w} = (q_{ij})\mathbf{w} \\
q_{ij} &= \mathbf{e}(i)^T B \mathbf{e}(j) \\
&= \sum_{s \in \mathcal{M}} \sum_{m \in s} \omega(A_0(s,m)) \mathbf{e}_m^{(s)}(i)^T D_m(s)^2 \mathbf{e}_m^{(s)}(j)
\end{aligned} \tag{4.16}$$

and the projection amounts to solving the equation

$$Q\mathbf{w} = \mathbf{v} \tag{4.17}$$

Weighted Projection Algorithm

Further manipulation of equation 4.15 is possible in order to convert to a form more suitable for the computation of the weight vector, $\mathbf{w}$. Noting that $\tilde{\boldsymbol{\mu}}_m^{(s)} = \frac{A_1(m,s)}{A_0(m,s)}$, the i^{th} component of $\mathbf{v}$, v_i, is given by

$$v_i = \sum_{s \in \mathcal{M}_\lambda} \sum_{m \in s} \omega(A_0(s,m)) \mathbf{e}_m^{(s)}(i)^T {D_m^{(s)}}^2 \tilde{\boldsymbol{\mu}}_m^{(s)} \tag{4.18}$$

$$= \sum_{s \in \mathcal{M}_\lambda} \sum_{m \in s} \frac{\omega(A_0(m,s))}{A_0(m,s)} \mathbf{e}_m^{(s)} {D_m^{(s)}}^2 A_1(m,s) \tag{4.19}$$

The algorithm can then be described thus

1. Compute the zeroth and first order accumulators for all distributions, $A_0(m,s)$ and $A_1(m,s)$

2. Compute $\mathbf{v}$ using equation 4.19

3. Compute $Q = (q_{ij})$ using equation 4.16

4. Solve the linear equations $Q\mathbf{w} = \mathbf{v}$

5. Compute the new model mean $\hat{\boldsymbol{\mu}} = E\mathbf{w}$

4.2.3 Comparing MLED and Projection Techniques

The Use Of Constraints

ED constrains a model to lie in $\mathcal{K}$-space. When performing projection, we first estimate an unconstrained model, and then enforce the constraint that the model lie in the eigenspace (using a least-squares metric to determine the closest model within the eigenspace to the unconstrained model). By using MLED, we enforce the constraint before performing any model estimation [16]. The re-estimation formula will re-estimate the model subject to the constraint, guaranteeing no reduction in likelihood. The projection technique has no such guarantee.

The difficulty with imposing the constraint is that the true model may not actually lie in the eigenspace, and thus the technique is not convergent. If the amount of data is small, then this is not a problem, as the true model could not be robustly estimated. However, if a suitable amount of data might be available, then the eigenvoice technique will be insufficient. To circumvent the problem, eigenvoice techniques may be combined with MAP [9], or some other technique, as was mentioned for MLLR.

Using An Identity Weight Function

Consider the special case where the weight function used in the projection technique is the identity function, that is $\omega(x) = x$, and the component-by-component covariance matrices are used for decorrelation. Then, from equations 4.15 and 4.16,

$$\begin{aligned} v_i &= \sum_{s\in\mathcal{M}}\sum_{m\in s} A_0(s,m)\mathbf{e}_m^{(s)}(i)^T \left(C_m(s)^{-1/2}\right)^2 \tilde{\boldsymbol{\mu}}_m^{(s)} \\ &= \sum_{s\in\mathcal{M}}\sum_{m\in s}\left(\sum_{t=1}^{T}\gamma_m^{(s)}(t)\right)\mathbf{e}_m^{(s)}(i)^T C_m(s)\frac{\sum_{t=1}^{T}\gamma_m^{(s)}(t)\mathbf{o}_t}{\sum_{t=1}^{T}\gamma_m^{(s)}(t)} \\ &= \sum_{s\in\mathcal{M}}\sum_{m\in s}\sum_{t=1}^{T}\gamma_m^{(s)}(t)\mathbf{e}_m^{(s)}(i)^T C_m^{(s)}\mathbf{o}_t \qquad (4.20) \\ q_{ij} &= \sum_{s\in\mathcal{M}}\sum_{m\in s} A_0(s,m)\mathbf{e}_m^{(s)}(i)^T \left(C_m(s)^{1/2}\right)^2 \mathbf{e}_m^{(s)}(j) \\ &= \sum_{s\in\mathcal{M}}\sum_{m\in s}\sum_{t=1}^{T}\gamma_m^{(s)}(t)\mathbf{e}_m^{(s)}(i)^T C_m(s)\mathbf{e}_m^{(s)}(j) \qquad (4.21) \end{aligned}$$

Comparing these equations with equation 4.9 and 4.10, we see that the equations are the same. Hence, projection with occupancy weighting and component-by-component covariance decorrelation is equivalent to MLED.

4.2.4 Comparing Eigenvoice And Other Adaptation Techniques

Only a small proportion of the $\mathcal{D}$-space consists of models that are humanly realisable. MAP and MLLR do not make use of this *a priori* knowledge, and allow the model they estimate to range over the entire space. In theory MAP could incorporate this information via the prior, but to do so is mathematically intractable. The use of eigenvoices is an attempt, albeit crude, to constrain the search to the space of models that are realisable. The attempt is crude for the following reasons. Firstly, the estimate of the space itself, being a linear closure of a set of vectors, includes many models that unrealisable; moreover the basis vectors themselves may not be realisable, being a linear combination of model parameters from observed data. In addition, the estimate of $\mathcal{K}$-space is made from a finite (and probably small) set of speakers, and so may not reflect the true space of possible models.

RMP and CAT also attempt to incorporate prior knowledge. Of the two, CAT is closer to ED. Indeed, comparison of equations 3.8 to 3.11 and 4.8 to 4.10 reveal that in the adaptation stage CAT and MLED are identical and it is only the choice of basis for the constrained space that is different. CAT assumes that the space of realisable speaker models is in the span of 'typical' cluster speakers. When using PCA, ED attempts to model the *variability* between speakers in an ordered fashion. We can thus pick the dimension of space according to the amount of adaptation data available.

RMP is a very different technique in that by looking across speakers it attempts to find linear relationships between differing parameters, using *training data*. This is in contrast to MLLR, which attempts to find linear relationships to a speaker independent model using *adaptation data*.

When performing speaker adaptation, the more data that is available, the more parameters that we can robustly estimate. When performing rapid speaker adaptation, we have a very small amount of data, and so keeping the number of parameters at a minimum while still allowing sufficient variability to produce a good model is a very important compromise. Also, the unseen parameter problem becomes much worse: not only are most parameters unseen, but very few parameters will have sufficient data for robust estimation of their value. Thus the way that data is pooled is critical. Ideally, every parameter should be influenced by every observation to obtain the most robust estimation possible. The following table summarises the position for each technique.

	Adaptation of unseen parameters	Data polling
MAP	No adaptation	No data pooling
RMP	Adapted as a fixed linear combination of other adapted parameters	Pooled according to fixed regression coefficients
MLLR	Adapted by estimated transform for each class	Data pooled per class
CAT	Adapted globally	All data used for estimating every parameter
Eigenvoice	Adapted globally	All data used for estimating every parameter

4.2.5 Computational Efficiency Of Various Adaptation Techniques

In this section we examine the relative efficiency of each of the adaptation techniques mentioned. Suppose that we have a set of H HMMs each having S states in all and employing M mixture component per state. Let d be the dimension of our observation vector, and T the number of frames of observations. Furthermore, let P be the number of phones observed.

The first stage of all techniques is to compute the zeroth and first order accumulators using the forward-backward algorithm. For each phone, we have SM distributions, and an average of T/P frames of observations, so the time per phone is $O((SM)^2\frac{T}{P})$ [20]; therefore the overall time complexity is $O((SM)^2T)$. We have HSM distributions to store accumulators for, so the storage required is $O(HSMd)$. Hereafter, these figures will be taken as given.

MAP

When using MAP, for each observed distribution, we compute equation 3.2, where the summations are the previously computed accumulators. The time requirement is clearly $O(d)$, giving rise to a time complexity of $O(PSMd)$ and space complexity similarly.

Combining these two phases gives an overall complexity of $O(SM(SMT + Pd))$ in time and $O(HSMd)$ in space.

RMP

After performing MAP, the second stage of RMP is the regression. If we assume that diagonal regression matrices are being used, it will take $O(Kd + d)$ operations per state, so $O(HSM(K + 1)d)$ operations in all. The space required to store the regression matrices is $O(HSM(K + 1)d$. The final combining of regression and MAP parameters will also take $O(HSMd)$. Therefore, overall, RMP has a time complexity of $O(SM(SMT + HKd))$ and a space complexity of $O(HSMKd)$.

MLLR

Suppose that we are using K regression classes to perform out adaptation; suppose for the purposes of computation that each one will apply to $\frac{HSM}{K}$ states. Using the notation of [14] for the intermediate matrices used in computing the adaptation matrix, to compute the g^i_{jk} takes $O(\frac{HSM}{K})$, which gives a time of $O((d+1)^2\frac{HSM}{K})$ for each G^i. To compute each row of W takes d matrix inversions of a matrix of order $d + 1$ (G^i), so each inversion is of order $(d + 1)^3$, $O(d^4)$ in all. So for a *single* regression class, we have a time order of $d^2(d^2 + \frac{HSM}{K})$; all matrices are of size $O(d^2)$, so the space complexity is $O(d^3)$ as we have $d+1$ G^i and 1 W. As we had K regression classes, this gives a time complexity of $O(d^2(Kd^2 + HSM))$ and a space complexity of $O(Kd^3)$.

Updating the means now becomes a trivial matter of multiplying a vector by a matrix. We have HSM means to update, each taking $O(d^2)$, giving $O(HSMd^2)$ in all, with trivial space constraints.

Putting each stage together gives a time complexity of $O(SM(SMT+HKd^2))+$ $O(Kd^4)$ and a space complexity of $O(HSMd^3)$.

CAT

To store the cluster speakers takes $O(KHSMd)$ space. The adaptation techniques involves computing equations 3.8 to 3.11. If $C^{(s)}_m$ is diagonal. then to compute each

source matrix of $G_w^{(r)}$ takes $O((Kd)^2))$, giving $O(HSM(Kd)^2)$ in total. Similarly, $\mathbf{k}_w^{(r)}$, has a time complexity of $O(HSMKd^2)$. As $G_w^{(r)}$ is a $K \times K$ matrix, to solve the linear equation 3.8 is an $O(K^3)$ operation (ignoring Strassen's method [19]). Storing the cluster speaker models takes $O(KHSMd)$ space; the space required for $G_w^{(r)}$ and $\mathbf{k}_w^{(r)}$ is comparatively insignificant. The update time for the means will also be comparatively insignificant.

Combining these results gives a time complexity of $O(SM(SMT + HK^2d^2))$ and a space complexity of $O(KHSMd)$.

Eigenvoice Decomposition

Given the accumulators, we need to compute the matrix Q and the vector $\mathbf{v}$. For each element of Q, from equation 4.16, we see that there is a time complexity of $O(HSMd^2)$, assuming that D^2 has been precomputed, giving a time complexity of $O(K^2HSMd^2)$ overall. Space required is $O(K^2)$, plus the space to store the eigenvoices, which is $O(KHSMd)$. Similarly, from equation 4.15, the time complexity for computing $\mathbf{v}$ is $O(KHSMd^2)$.

The final step is to solve the linear equation 4.17. As the equation is of order K, solving this is an $O(K^3)$ operation. Updating the means is a comparatively insignificant operation.

Adding these together, and removing dominated terms gives an order for the time complexity of $O(SM(SMT + H(Kd)^2)) + O(K^3)$. However, since normally $HSM \gg K$, we can expect the final term to be dominated, and hence discounted. Space complexity is $O(KHSMd)$.

Summary

Method	Space	Time
MAP	$O(HSMd)$	$O(SM(SMT + Pd))$
RMP	$O(KHSMd)$	$O(SM(SMT + HKd))$
MLLR	$O(HSMd^3)$	$O(SM(SMT + HKd^2)) + O(Kd^4)$
CAT	$O(KHSMd)$	$O(SM(SMT + H(Kd)^2))$
Eigenvoice	$O(KHSMd)$	$O(SM(SMT + H(Kd)^2))$

Table 4.1: Time and Space Requirements For Adaptation Techniques

Table 4.1 gives the time and space requirements for each adaptation technique that we have considered. Under both considerations MAP is clearly the most economical. Three other methods, RMP, CAT and eigenvoice, have the same space requirements given the same "order" of adaptation (K). As K will typically be no more than 30, and d 39, $K^2 \ll K_{mllr}d^2$, (K_{mllr} typically 2 or 3) and so MLLR is much more space-intensive. This means that the other techniques are much more suitable for embedded applications, for example, telephony, than MLLR. However, time considerations are often more important than space, given the amount of memory available in many systems. Again, RMP is much less computationally intensive, while CAT and eigenvoice decomposition have the same complexity. It is much harder to compare MLLR, as it has a different structure in its computation. Firstly notice, when a lot of data is being used, the time complexities are dominated by the time taken to perform the forward backward algorithm. We therefore consider the case when SMT is sufficiently small for the other quantities to be significant. The relative performance of ED (& CAT) and MLLR is then determined by which is the larger quantity, $HSM(K-1)$, or d^2. Thus, if the models are very complicated,

so HSM is large, MLLR will be more efficient, on simpler models, Eigenvoice techniques are preferable.

Chapter 5

Experimental Evaluation

5.1 Implementation

The following software was written in order to evaluate the efficacy of MLED and Eigenvoice Projection techniques, using the HTK library [15]. Existing HTK tools were used to evaluate MLLR and MAP techniques.

EVExtractSV	Extracts a supervector of means from a set of HMMs.
EVChoose	Given a list of supervectors, performs PCA and writes a series of supervectors that are the principal components to file.
EVEAdapt	Updates a model file using Eigenvoice Adaptation.

Full source code is supplied in the companion volume to this thesis, which includes a description of the supervector file format in the comments to the `sv.c` file. All the code is my own work, apart from the code for Singular Value Decomposition and LU Decomposition, and their support routines, which is taken from [19].

5.2 Experiments

The corpus used for the experimental evaluation is the speaker-independent portion of the resource management corpus [18]. The training corpus consists of data from 109 speakers, and then their are four test corpora, each of 10 speakers. In all cases, 1 speaker speaks 30 sentences. The four corpora are described by their release date: February 1989, October 1989, February 1991 and September 1992.

Three differing acoustic models were used in the test: single Gaussian monophone and triphones, and 6 mixture component triphones. Each HMM had three internal states.

To compare the relative effectiveness of different adaptation techniques, the first 10 sentences of the 30 for each speaker in the test corpus was withheld with which to perform adaptation, the remaining 20 being used to estimate the accuracy of the model. To act as a prior, a speaker-independent model was estimated by pooling the data for all 109 speakers in the training corpus. Using the HTK Toolkit ([15]), the following adaptation techniques were tested using upto 10 adaptation sentences.

1. No adaptation

2. MAP re-estimation

3. MLLR with full regression matrices

4. MLLR with block diagonal regression matrices, each block being 13×13 in size

5. Both MLLR techniques, each followed by MAP

The effect of choosing different bases for the eigenspace was investigated. A set of 109 speaker-dependent models were generated by taking the speaker-independent model and performing MLLR adaptation with full regression matrices, followed by MAP. Two different bases were chosen: one taking the speaker models, and one performing PCA on XX^T (section 4.1.1). In the case of the monophone models, a basis was also determined by performing PCA on the covariance matrix. Using MLED, the recognition accuracy obtained by using eigenspace dimensions of between 1 and 30 was recorded. In all cases when using ED, 3 iterations of the re-estimation algorithm as employed.

To investigate the effect of differing projection techniques, different choices for the weight function and decorrelation matrix were made. Three differing decorrelation techniques were compared: no decorrelation, component-by-component covariance matrix inverse, and the average of these covariance matrices, inverted. The weight functions were all taken from the same family, defined by

$$\rho_q(x) = \begin{cases} 0 & x \leq 0 \\ x^q & x > 0 \end{cases} \tag{5.1}$$

There are two special cases of this equation, when $q = 0$, which is called *indicator* projection, and when $q = 1$, which is called *occupancy* weighting, and when combined with component-by-component decorrelation is equivalent to MLED (section 4.2.3).

Next, for 2-, 10-, and 30-dimensional eigenspaces, with an appropriate choice of basis, the effect of altering the amount of adaptation data was investigated. Finally, the effect of adding MAP and MLLR adaptation to the ED was investigated.

The following section discuss the results obtained; a results summary is in appendix A, detailed results are supplied in the supplementary volume.

5.3 Choosing An Eigenspace

As described in section 4.1.1, a basis for an eigenspace was chosen using Principal Component Analysis on the matrix XX^T. Figure 5.1 gives the cumulative variability. Since there are at most 109 (number of training data supervectors) positive eigenvalues of the covariance matrix ([7]), the SVD technique will derive all the required eigenvalues to assess the proportions of variability.

Using the single Gaussian monophones, 17 dimensions are required to account for half the variability of the speech signal; with the most complicated models (triphones with 6 mixture components) 19 are required; strangely, with the single Gaussian triphones 24 dimensions are required. This reduction in the variability for the more complicated models is probably due to the difficulty in estimating the speaker-dependent models on the small amounts of adaptation training data available.

As the eigenvalues should be derived from the SVD in order of decreasing magnitude, the space under the cumulative graphs should be convex. Consideration of figure 5.1 shows that this is not the case after approximately 40 eigenvalues. This is indicative of a lack of numerical stability in the algorithm implementation.

We also see the effect of a lack of stability in figures 5.2 to 5.4. Using the single Gaussian models, the word error rate decreases as the dimension of the bases increases, until approximately 15 dimensions. After this point the recognition rate

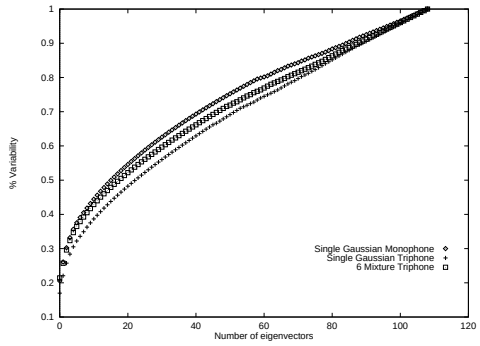


Figure 5.1: Cumulative Variability of First 109 Eigenvalues

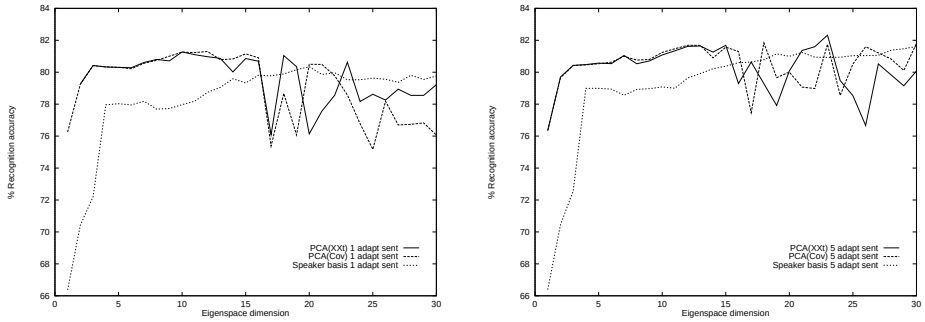


Figure 5.2: Effect Of $\mathcal{K}$-Space Dimensionality On Recognition Accuracy (Monophones)

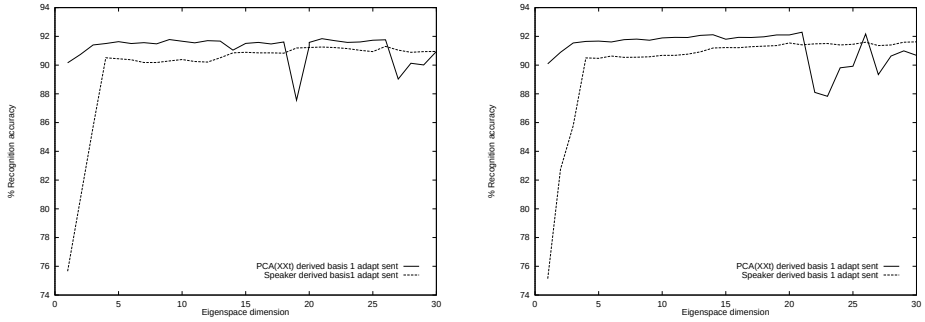


Figure 5.3: Effect Of $\mathcal{K}$-Space Dimensionality On Recognition Accuracy (Single Mixture Triphones)

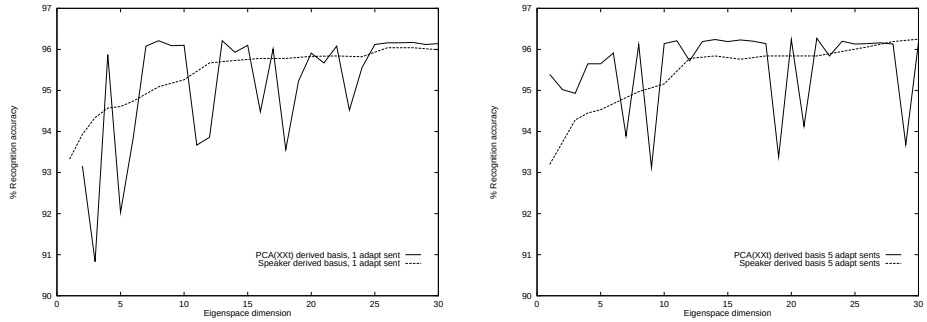


Figure 5.4: Effect Of $\mathcal{K}$-Space Dimensionality On Recognition Accuracy (6 Mixture Triphones)

behaves very unpredictably. As the behaviour is independent of the number of adaptation sentences, it is not due to problems in estimating the ED, and therefore it indicates a lack of robustness in the estimation of the eigenvoices. As additional evidence, although the recognition rate when using a basis of original speaker vectors starts considerably lower than using the PCA-derived basis, indicating that PCA is indeed capturing the inter-speaker variability, as the dimension of the space is increased the recognition rate continues to increase, instead of behaving unpredictably.

However, this is unlikely to be only a problem with numerical stability, as the $PCA(\Sigma)$ derived basis (computed using Matlab) exhibits the same behaviour as the $PCA(XX^T)$ derived one. This indicates that 109 speakers provides insufficient data to robustly estimate more than about 14 dimensions of inter-speaker variability.

The word error rate for the speaker-derived basis of the 6 mixture component triphones also decreases as the dimension of the space increases. On the other hand, the recognition accuracy using an eigenspace with a PCA-derived basis is very unstable with the dimension of the space. This is indicative of very poor estimates of the basis vectors. Performing the PCA on parameter vectors this large is difficult, even with the SVD technique.

Experiments of altering the training data used in obtaining the initial speaker dependent model set are reported in [11]. They found that for the technique to work well, the set of training speakers should be matched as closely as possible with the test speakers, so that the same inter-speaker variability is matched. It was also found that it was better to have half as much data from each speaker than to have data from half the number of speakers. This is indicative of the stability problem above, as reducing the number of speakers will decrease the robustness of the estimate of the dimensions of variability, and so only the few most important eigenvoices will be reliably estimated.

5.4 Choosing A Projection Technique

The definition of the weighted projection in section 4.2.2 was very general. The aim of the following experiments was to determine the effect of differing choices for Ω and D.

The effect of the three differing decorrelation techniques was first investigated: using no decorrelation, component-by-component decorrelation (c) and global decorrelation (g) . Using indicator (Ind) and occupancy (occ) weighting only, the average percentage point difference compared with no correlation for both fixed amounts of adaptation data, averaging across different numbers of dimensions (table 5.1), and

for a fixed dimensionality, averaging across differing amounts of adaptation data (table 5.2), was computed.

Num Sents	Mono - Occ (s)	Mono - Occ (g)	Mono - Ind (s)	Mono - Ind (g)
1	0.42	1.17	0.06	0.00
2	0.17	0.13	0.27	0.00
5	-0.27	-0.63	0.05	0.00
10	0.02	0.21	-0.15	0.00
Num Sents	Tri - Occ (s)	Tri - Occ (g)	Tri - Ind (s)	Tri - Ind (g)
1	0.11	0.71	-2.15	2.51
2	-0.20	0.52	5.01	4.86
5	-0.40	0.31	4.72	4.69
10	0.10	0.46	4.82	4.61

Table 5.1: Effect Of Decorrelation Against Quantitity Of Adaptation Data (% point change on recognition accuracy)

Num Dim	Mono - Occ (s)	Mono - Occ (g)	Mono - Ind (s)	Mono - Ind (g)
1	0.03	-0.12	-0.04	0.00
2	0.24	0.16	0.23	0.00
5	0.02	0.01	0.02	0.00
10	0.06	-0.02	0.01	0.00
20	-1.39	-0.06		
30	1.49	1.27		
Num Dim	Tri - Occ (s)	Tri - Occ (g)	Tri - Ind (s)	Tri - Ind (g)
1	-1.77	-0.53	9.36	9.96
2	-0.27	0.34	2.15	4.14
5	0.34	0.62	0.46	0.76
10	0.95	0.99	0.96	1.12
20	0.42	0.42	0.56	0.32
30	-0.46	0.91		

Table 5.2: Effect Of Decorrelation Against Eigenspace Dimension (% point change in recognition accuracy)

In general, the type of decorrelation chosen, if any, has little effect on the word error rate of the technique. Notice how for monophones, the state-by-state decorrelation mostly outperforms that for the global one, whereas for triphones, the situation is reversed. This is as the covariance matrices are unlikely to be as robustly estimated for many of the distributions in the case of triphones, so averaging them out reduces the problem.

The use of decorrelation has a much higher effect for triphones generally than for monophones. This is particularly true when using indicator-weighted projection with large numbers of adaptation sentences and low dimensionality. This is as there is more scope for certain (possibly badly estimated) parameters dominating results, as these are likely to have large variances in the training models. Decorrelating these removes this bias. Occupancy weighting will also help to remove this bias, as triphones which occur often is test data are likely to also have occurred often in the training data, and so the variances of these parameters will be much less, and hence the need to remove the effect of a large variance is also less.

When investigating the effect of altering ω, the component-by-component decorrelation was always used. This enables us to compare the benefit of the MLED technique of Kuhn, Nguyen *et al* with the weighted projection technique, MLED being occupancy weighted projection with this decorrelation matrix. Graphs of the effect of varying q in equation 5.1 are shown in figures 5.5 to 5.7.

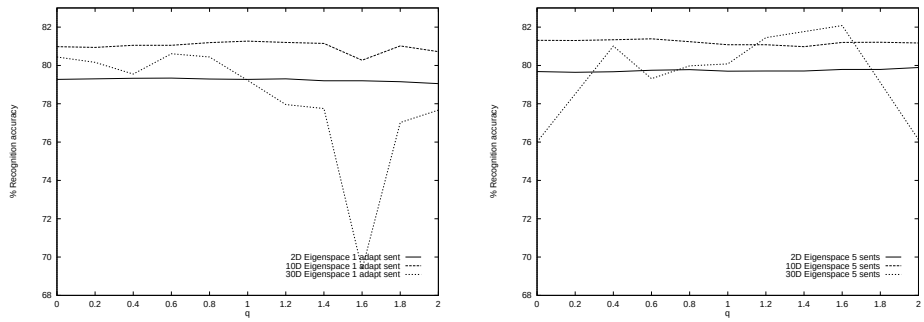


Figure 5.5: Effect of Altering the q value For ρ-Weighted Projection (Single Gaussian Monophone)

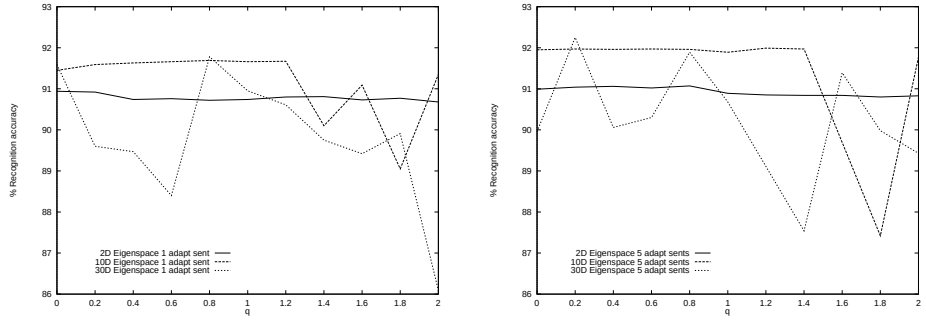


Figure 5.6: Effect of Altering the q Value For ρ-Weighted Projection (Single Gaussian Triphone)

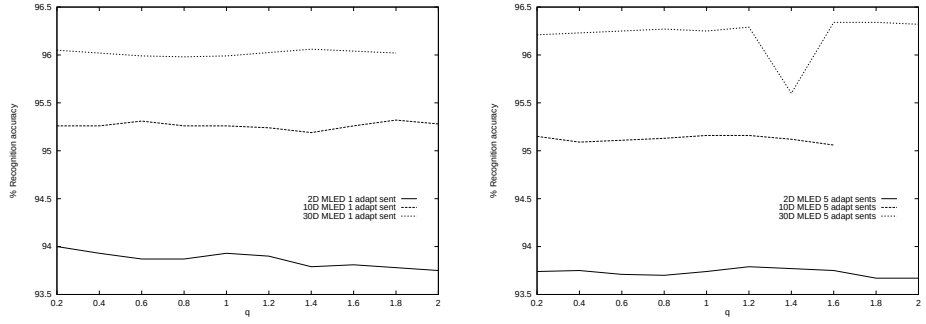


Figure 5.7: Effect of Altering the q Value For ρ-Weighted Projection (Six Gaussian Mixture Triphone)

Given that MLED ($q = 1$) is an EM-derived algorithm, one would expect this to be the most effective technique, and for the recognition accuracy to tail off on either side of this value. This only occurs in the 30-dimensional case when using PCA(XX^T) (hence unrobustly) derived basis, in all other cases around $q = 1$ the choice of q is unimportant, and there is no evidence of a tail off for low or high q.

5.5 Comparing Techniques

Given this analysis of ED, the results can be compared with those for other techniques. We restrict our attention to just three cases of ED: using a low (2), medium (10) and high (30) dimensioned eigenspace, with MLED. The basis vectors are chosen via PCA, except when using a 30-dimensional space or 6 mixture components, when a basis of speaker vectors was used (see section 5.3 for justification).

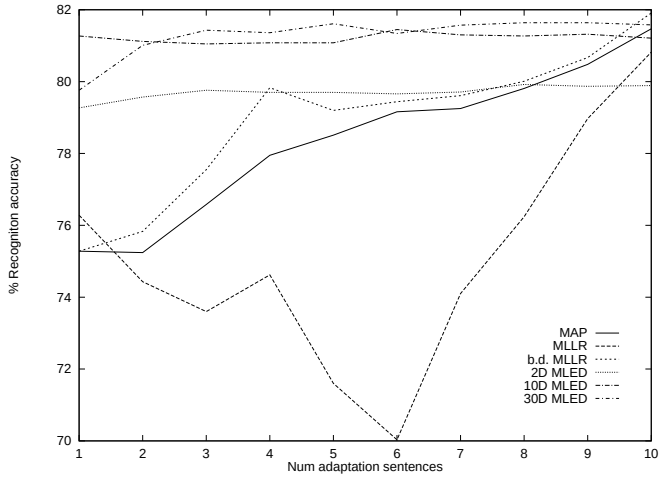


Figure 5.8: Recognition Accuracy Against Number of Adaptation Sentences (Single Gaussian Monophones) (Speaker Independent = 76.28%)

Figures 5.8 to 5.10 show how the recognition accuracy changes with the amount of adaptation data. Firstly note that for the conventional techniques of MAP and MLLR the results on very little data results in a reduction in all but one case, compared with performing no adaptation (the results for which are given in the caption). With a few more sentences of data, then the results start to improve, more rapidly for MLLR than for MAP.

Compared with the change in recognition accuracy with the amount of data for MLLR and MAP, ED gives very little change in recognition accuracy as the amount of data increases; after 2 sentences the recognition accuracy is essentially flat.

With the single Gaussian models, an eigenspace dimension of just 2 is sufficient to obtain a significant increase in recognition accuracy; higher dimensions help further. From figures 5.2 to 5.4, we see that only in the case of the most complicated models does having much more than 10 dimensions give further improvement. In the exception case, then no significant advantage is gained with less than 30 dimensions. However, as all the bases are speaker derived, if a more systematic model of inter-speaker variability were to be employed, a lower dimensional space may be sufficient.

Notice also that the more complicated the model the fewer sentences that are required for MLLR to be as effective as MLED. This is as the number of parameters to be estimated for the MLLR global transform is fixed, whereas with MLED we need to increase the eigenspace dimension to sufficiently model the extra difference

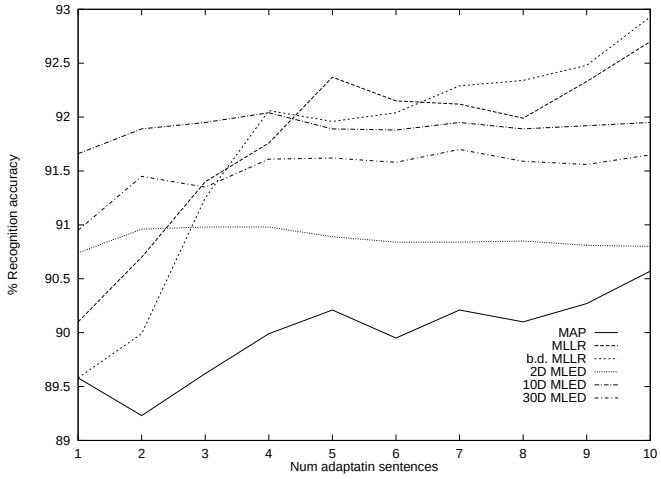


Figure 5.9: Recognition Accuracy Against Number of Adaptation Sentences (Single Gaussian Triphones) (Speaker Independent = 90.10%

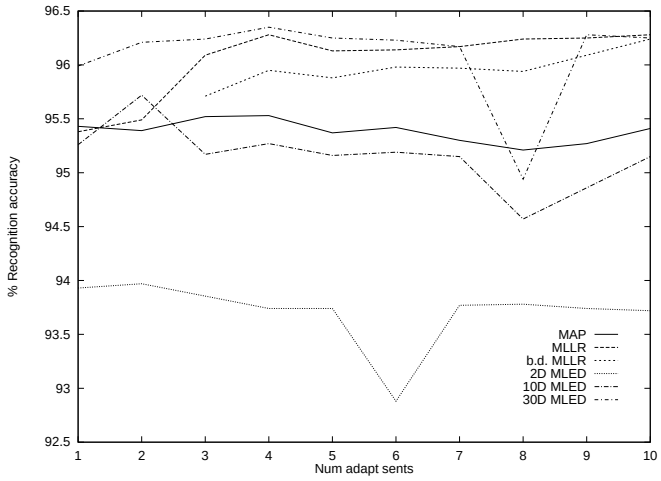


Figure 5.10: Recognition Accuracy Against Number of Adaptation Sentences (Six Gaussian Mixture Triphones) (Speaker Indendent = 95.38%)

between models (section 5.3); MLLR provides a coarse step in the number of adaptation parameters, and so just 1 transform can provide reasonable adaptation, once it has been robustly estimated.

MAP fares well as an adaptation technique for the simplest model only; for the most complicated it hardly affects the recognition accuracy, even with 10 adaptation sentences. This is as the number of model parameters increase the number of parameters with unseen data increases; also as there is less data available per parameter the speaker independent model is more dominant (cf. equation 3.2).

The number of parameters to be estimated, and the amount of data required to estimate each is an important consideration. It was hypothesised that ED was a useful technique was because it had very few parameters to be estimated, compared with other techniques (Table 5.5).

Technique	Monophone	Triphone	6 Mixture Triphone
MAP	5655 [a]	53313	320112
Full MLLR	1 [b] $\times(39 \times 39 + 39) = 1560$ [c]	1560	1560
B.D. MLLR	$1 \times (3 \times (\frac{39}{3} \times \frac{39}{3} + 39) = 546$ [d]	546	546
Eigenvoice	~ 10 [e]	~ 10	~ 10

[a] Number of model parameters

[b] The number of regression matrices is assumed to be 1. The occupancy threshold was set to 700, and so if we assume even data splits the increase to 2 matrices will occur between 3 and 17 adaptation sentences

[c] Num of matrices $\times$ (Dim of speech vector $\times$ Dim of speech vector + Num offset params)

[d] Num of matricies $\times$ Num of blocks $\times$ (Size of block $\times$ Size of block + Num offset params)

[e] Dimension of $\mathcal{K}$-space

Table 5.3: Number of Parameters to Be estimated By Differing Techniques

From the results, we may estimate that we can robustly estimate the ED for 10 dimensions, so we could allow $\frac{1}{10}$ sentences per parameter. For MAP and MLLR, this threshold seems to be at 5 sentences, and so we would allow between $\frac{1}{100}$ and $\frac{1}{300}$ sentences per parameter. Thus it requires more data to robustly estimate the eigenvoice decomposition, so when more data is available, we miss the subtlety of adaptation that many parameters afford. However, MAP requires data for a parameter to adapt it, MLLR requires data from a cluster to adapt a cluster, whereas ED can use any data available to perform the adaptation.

5.6 Combining Adaptation Techniques

As discussed in the previous section, for eigenspaces up to approximately 30 dimensions, 2 adaptation sentences is sufficient to robustly estimate the ED, and then no benefit is accrued from increasing the quantity of adaptation data. We therefore consider whether ED may be combined with another adaptation technique to further improve the models.

MAP is one technique that is often used as a post-processing step in adaptation. There are a couple of reasons for this: firstly, it is a convergent technique, so with sufficient data we obtain the true speaker dependent model; secondly, given the accumulators which in general have already been computed for the first adaptation step, it requires no additional storage and is time-linear in the number of parameters to be updated.

It has been found that MAP was ineffective on the 6 mixture component triphone models, so we only consider employing MAP as a post-processing step on simpler models. For comparison, we also consider using MAP in combination with MLLR.

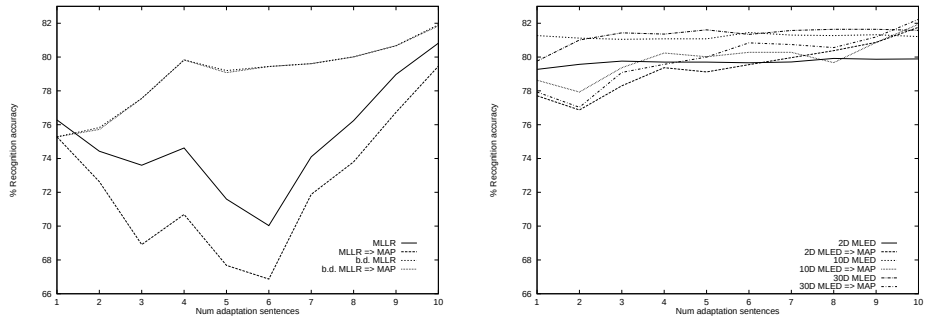


Figure 5.11: How Recognition Accuracy Is Affected By Combining Adaptation With Map (Single Gaussian Monophone)

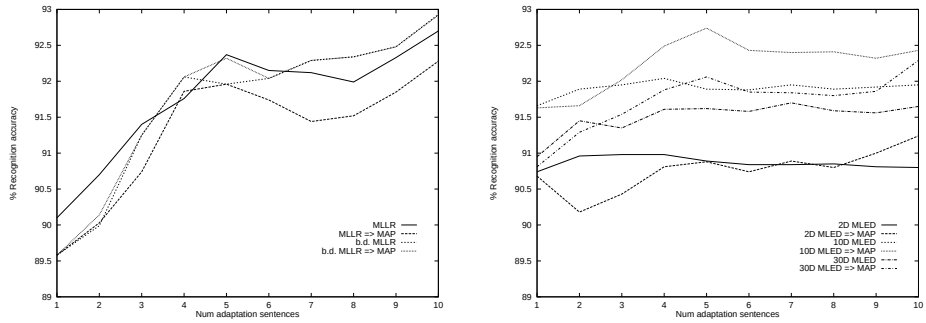


Figure 5.12: How Recognition Accuracy Is Affected By Combining Adaptation With Map (Single Gaussian Triphone)

There is only one case whereby adding MAP gives a significant improvement of more than 1% point to the recognition accuracy, that being 2-dimensional MLED on the monophone models. In other cases, even with 10 adaptation sentences there is very little improvement. One would therefore take it that parameters that MAP can estimate robustly on 10 adaptation sentences have already been adapted to a good approximation of the true value by the first technique; other parameters MAP, by definition, will have little effect upon.

In all cases, adding MAP when there is little data has a negative effect. We have, though, applied MAP in a naïve manner, merely substituting the adapted model for the speaker independent prior in equation 3.2. To apply MAP systematically, we would require an estimate for the variance of the parameter estimate and the maximal likelihood estimate. The adapted estimate of the mean may then be combined with the maximal likelihood estimate in a manner akin to RMP (3.2). In the case of ED, this variance would probably depend not only on the adaptation data, but also the eigenvalues of the original basis definition, which give the inter-speaker variance for each basis vector.

In the experiments, when the amount of adaptation data is very small, the variance of the maximal likelihood estimate is large, and so the combined estimate would be estimate by the first adaptation technique. This will ensure that adding MAP will have only minimal negative impact. It will also improve the effect of adding MAP with more adaptation sentences, as some parameters will still be badly estimated by maximal likelihood, and these parameters ill be little changed when adding MAP in this way.

As MAP proved little effective, we also consider the effect of adding MLLR to ED. The results are shown in figures 5.13 to 5.15.

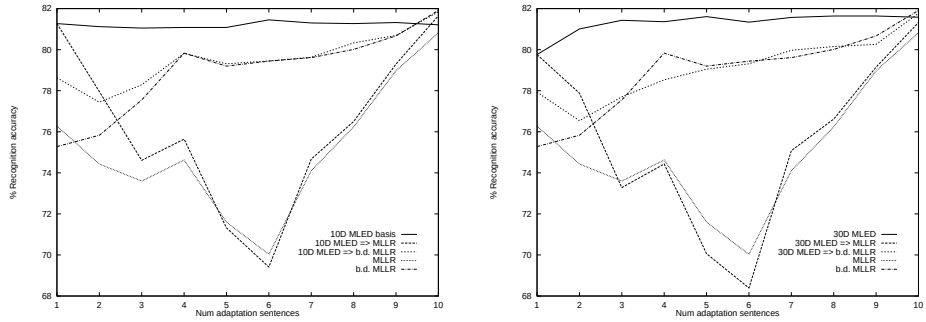


Figure 5.13: How Recognition Accuracy Is Affected By Combining Adaptation With MLLR (Single Gaussian Monophone)

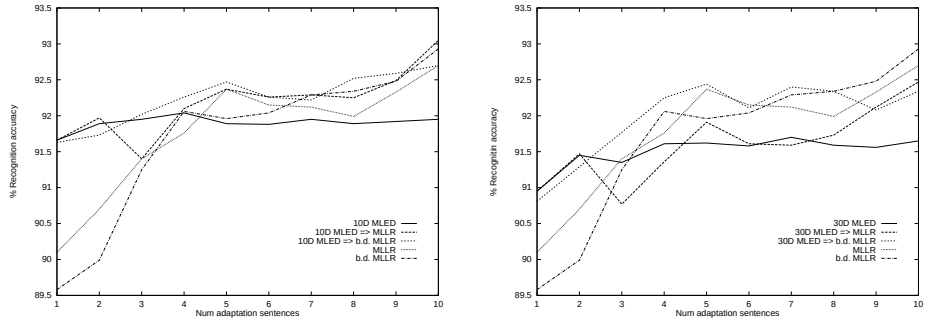


Figure 5.14: How Recognition Accuracy Is Affected By Combining Adaptation With MLLR (Single Gaussian Triphone)

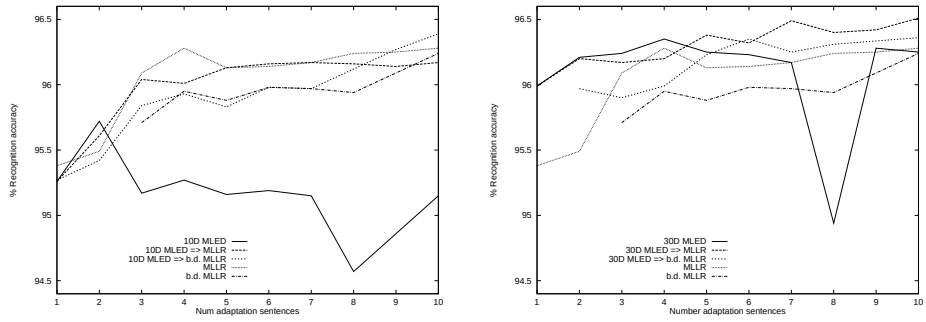


Figure 5.15: How Recognition Accuracy Is Affected By Combining Adaptation With MLLR (6 Mixture Component Triphone)

As with MAP, the effect of adding MLLR with very little adaptation data is negative, although in contrast to MAP initially the effect is less, then it increases, before finally generally having a positive effect. Notice also, that initially MLLR with block diagonal matrices fares worse. We could therefore infer that with very little data MLLR is estimating only a slight transform, which is having little effect, as the amount of data per transform parameter increases, then the transform estimated is greater, but still not robust enough to perform good adaptation.

We also see the same principle as with applying MAP, in that the more complicated the model, the less adaptation data is required before adding the post processing step becomes worthwhile. With the monophone models, even with 10 adaptation sentences it is hardly worth adding a post processing step, whereas with the more complicated 6 mixture component triphones only a few adaptation sentences are required. The reasons for this are the same as discussed in the previous section; there are not enough dimensions in the eigenspace to adequately model the inter-speaker variability between models (compounded by having to use a speaker derived basis rather than a PCA-derived one for reasons of robustness), and so the model estimated by ED is still poor, and as MLLR is only estimating a global transform the complexity of the adaptation is independent of the number of parameters of the model. Justification for the lack of modelling of the variability can be seen in figure 5.15. The 30-dimensional MLED is giving far superior performance to the 10-dimensional case, and combining 10-dimensional MLED with MLLR is very little different to performing MLLR on its own.

In summary, with models of up to 10,000 parameters, with up to 10 adaptation sentences there is very little benefit from combining ED with MAP or MLLR. With models of more than 100,000 parameters, there is very little benefit to be gained by first performing ED unless the eigenspace could be estimated in a more robust fashion. With very small amounts of data, in a sufficiently dimensioned eigenspace, ED on its own is superior to combining it with MAP or MLLR. One would expect the performance of the combined adaptation to increase if the supplementary adaptation technique could be combined more rigorously with the ED, rather than as a separate processing step.

Chapter 6

Conclusions

For the ED technique to be effective, it has been shown that it is important to model the inter-speaker variability well. Employing PCA on the matrix XX^T via SVD with 109 input vectors was found to be sufficient when only a low number of model parameters are required to be adapted, and no worse than applying PCA to the covariance matrix; but for more than about 15 dimensions, PCA was not found to be suitable. When the number of model parameters is large, then this method is not sufficient to estimate the directions of most variability. Hence, extra work is required to find robust ways of estimating the eigenspace is required. Kuhn *et al* suggest independent component analysis [11], others may be possible. Investigation into whether increasing the number of speakers in the training data increases robustness is also required.

The lack of robustness of estimation of the eigenspace is the reason cited for the lack of performance with the most complicated models. Two different methods of performing ED were compared, MLED and Weighted Projection. It was found that for one particular case of Weighted Projection, the method reduced to that of MLED. A family of weighted projection functions based on powers (equation 5.1) were evaluated, and it was found that the exact value used for the parameter of the function in the family was relatively unimportant. This is a useful result, as a patent application has been made for MLED [17], and so the new method of weighted projection may be used instead. There is another patent application made on ED [12] more generally; as these patents have not yet been granted it is unknown precisely what they cover.

ED was combined with MAP and MLLR to try and combine the rapid adaptation to a first model which the technique provides, with a slower adaptation to a truer model using MAP or MLLR. Neither technique provided encouraging results. It was suggested that a more systematic combination of ED and one of the other techniques may provide better adaptation. In particular, to combine ED and MAP effectively, it was suggested that it may be necessary to derive the estimate of the variance of a parameter estimate by ED.

There is more statistical analysis of the technique which may be performed. The rapidity of convergence of the models has not been investigated, although anecdotally the three iterations employed were generally found to be more than sufficient, and for low dimensioned eigenspace, one would have been sufficient. The other parameters that have had no statistical analysis performed on them are the estimate of the weights. It is very difficult to estimate the number of iterations required until convergence, if we do not know how close an estimate needs to be to the true value to have minimal effect on the recognition accuracy.

The weight vector could also be used, conceptually, to characterise a speaker. The eigenvoices are chosen in the directions to maximise the inter-speaker variabil-

ity, which is the criterion used when choosing features for speaker verification or identification, and Kuhn *et al* report that this has been attempted, ([11], patent application [13]). There is no published analysis, though, of the intra- and inter-speaker variability in this weight vector.

In this thesis, only speaker adaptation has been considered. There are other types of adaptation which may be required, in particular *environment adaptation*. Instead of adapting a model for a new speaker, we adapt a model to improve the recognition accuracy under differing conditions. To perform this using eigenvoices, we would require a corpus that contained segregated multi-environment data, to estimate the directions of variability between environments.

Adapting only the distribution means, as we have done in this thesis, is normally sufficient for speaker adaptation. However, changing environment often has a marked impact on the variance as well. Variances are always positive, when adapting variances, MLED would have this constraint built-in to the optimisation, and so a MLED re-estimation formula for variances could be derived. When using weighted projection, it is not immediately apparent how to apply the constraints. It may be possible to derive the least square error formula (theorem 1) with the constraints that the variances are positive built in. Alternatively, if there are only variances in the supervector of parameters being estimated, and the variances of the maximal likelihood estimate of the model before performing the projection will also be positive, in this case the constraint will be enforced automatically. This does imply that the covariance matrices will be diagonal, as adding in covariance values will destroy the positivity required.

ED has been shown to be a promising technique for speaker adaptation, giving a 6.5% increase in recognition accuracy with just a single sentence of adaptation data for the simplest models. An alternative technique to MLED for computing the decomposition has been derived and presented, and shown to be no worse than MLED, which has a patent application outstanding against it. It was found to be difficult to robustly estimate the eigenspace, and combine ED with other adaptation techniques. With sufficiently simple models where the eigenspace could be estimated robustly, ED was shown to be computationally less expensive than MLLR, and gives better recognition accuracy than MAP or MLLR adapted models with less than about 8 sentences of adaptation data.

Bibliography

[1] S. M. Ahadi and P. C. Woodland. Rapid speaker adaptation using model prediction. In *Proceedings ICASSP 95*, 1995.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the *EM* algorithm. In *Journal of the Royal Statistical Society*, volume B, pages 1–38, 1977.

[3] M. J. F. Gales. Cluster adaptive training for speech recognition. In *Proceedings ICSLP 98*, 1998.

[4] Jean-Luc Gauvain and Chin-Hui Lee. Maximum *a Posteriori* estimation for multivariate gaussian mixture observations of markov chains. *IEEE Transactions on Speech and Audio Processing*, 2:291–298, 4 1994.

[5] Paul E. Green and J. Douglas Carroll. *Analyzing Multivariate Data*, chapter 8-9. The Dryden Press, 1978.

[6] Paul R. Halmos. *Finite-Dimensional Vector Spaces*. Springer-Verlag, 1993.

[7] Richard Arnold Johnson and Dean W. Wichern. *Applied Multivariate Statistical Analysis*, chapter 8. Prentice-Hall, 4 edition, 1998.

[8] Maurice G. Kendall and Alan Stuart. *The Advanced Theory Of Statistics*, volume 3 Design and Analysis And Time-Series, chapter 43. Charles Griffin, 1966.

[9] R. Kuhn, P. Nguyen, J.-C. Junqua, and L. Goldwasser. Eigenfaces and eigenvoices: Dimensionality reduction for specialized pattern recognition. In *Proc. MMSP 98*, 1998.

[10] R. Kuhn, P. Nguyen, J.-C. Junqua, L. Goldwasser, N. Niedzielski, S. Fincke, K. Field, and M. Contolini. Eigenvoices for speaker adaptation. In *Proc. ICSLP 98*, 1998.

[11] R. Kuhn, P. Nguyen, J.-C. Junqua, N. Niedzielski, S. Fincke, K. Field, and M. Contolini. Fast speaker adaptation using a priori knowledge. In *Proceedings ICASSP 99*, 1999.

[12] Roland Kuhn, Patrick Nguyen, and Jean-Claude Junqua. Dimensionality reduction for speaker and environment adaptation using eigenvoice techniques, 6 1998. U.S. Patent application.

[13] Roland Kuhn, Patrick Nguyen, Jean-Claude Junqua, and Robert Boman. Speaker verification and speaker identification based on eigenvoices, 6 1998. U.S. Patent application.

[14] C. J. Leggetter and P. C. Woodland. *Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models*, pages 171–185. Academic Press Limited, 1995.

[15] Entropic Limited. *The HTK Book*. Entropic Limited, 1999.

[16] Patrick Nguyen. *Fast Speaker Adaptation*. PhD thesis, Institut Eurécom, June 1998.

[17] Patrick Nguyen, Roland Kuhn, and Jean-Claude Junqua. Maximum-likelihood method for finding an adapted speaker model in eigenvoice space, 4 1998. U.S. Patent application number 09/070,054.

[18] National Instute of Standard and Technology. Darpa resource management continuous speech database. CD-ROM, 1992. NIST Disc 2-3.1 and 2-4.2.

[19] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipies in C*. Cambridge University Press, 2 edition, 1992.

[20] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1 edition, 1992.

Appendix A

Results

A.0.1 Key

The following table gives a reference for the abbreviation of the adaptation techniques used in the following tables.

Abbreviation	Meaning
⇒	' ... followed by ... '
map	MAP.
mllr	MLLR with full regression matrices.
b. d. mllr	MLLR with block diagonal regression matrices.
pind	Weighted projection with indicator weight function and no decorrelation.
pdind	Weighted projection with indicator weight function using $C^{-1/2}$ as the decorrelation matrix.
pgind	Weighted projection with indicator weight function using $\overline{C}^{-1/2}$ as the deocrrelation matrix.
pocc	As *pind*, but using an occupancy weight function.
pdocc	As *pdind*, but using an occupancy weight function. This is MLED.
pgocc	As *pgind*, but using an occupancy weight function.
pdρ_q	As *pdind*, but using ρ_q (equation 5.1) as the weight function.

A.1 Single Gaussian Monophone

A.1.1 Conventional Techniques

Speaker independent recognition accuracy = 76.28%

Method	Eigenspace	Num Adaptation Sentences				
	Dimension	1	2	3	4	5
map		75.28	75.24	76.58	77.95	78.51
mllr		76.28	74.43	73.60	74.62	71.60
b.d. mllr		75.28	75.83	77.55	79.83	79.20
b.d. mllr ⇒ map		75.28	75.72	77.55	79.83	79.08
mllr ⇒ map		75.28	72.63	68.91	70.69	67.68
Method	Eigenspace	Num Adaptation Sentences				
	Dimension	6	7	8	9	10
map		79.16	79.25	79.81	80.48	81.47
mllr		70.03	74.10	76.24	78.97	80.82
b.d. mllr		79.44	79.61	80.01	80.67	81.91
b.d. mllr ⇒ map		79.44	79.61	80.01	80.67	81.84
mllr ⇒ map		66.87	71.88	73.80	76.73	79.46

A.1.2 Eigenvoice Decomposition With PCA(XX^T) Derived Basis

Method	Eigenspace Dimension	Num Adaptation Sentences 1	2	3	4	5
pdocc	1	76.55			76.35	
pdocc	2	79.27	79.57	79.76	79.70	79.70
pdocc	3	80.42				80.43
pdocc	4	80.33				80.48
pdocc	5	80.28			80.57	
pdocc	6	80.28				80.54
pdocc	7	80.61				81.05
pdocc	8	80.80				80.53
pdocc	9	80.71				80.71
pdocc	10	81.27	81.12	81.05	81.08	81.08
pdocc	11	81.10				81.34
pdocc	12	80.97				81.62
pdocc	13	80.87				81.64
pdocc	14	80.02				81.26
pdocc	15	80.86				81.69
pdocc	16	80.69				79.29
pdocc	17	76.09				80.64
pdocc	18	81.04				
pdocc	19	80.35				77.92
pdocc	20	76.14				80.08
pdocc	21	77.57				81.36
pdocc	22	78.56				81.60
pdocc	23	80.63				82.32
pdocc	24	78.17				79.48
pdocc	25	78.62				78.53
pdocc	26	78.26				76.67
pdocc	27	78.94				80.52
pdocc	28	78.55				
pdocc	29	78.55				79.15
pdocc	30	79.22	79.90			80.08
pdocc $\Rightarrow$ map	2	77.71	76.87	78.30	79.37	79.12
pdocc $\Rightarrow$ mllr	2	79.27	77.13			71.63
pdocc $\Rightarrow$ b.d. mllr	2	77.71	76.78			79.20
pdocc $\Rightarrow$ map	10	78.63	77.93	79.37	80.24	80.02
pdocc $\Rightarrow$ mllr	10	81.27	77.96	74.61	75.64	71.32
pdocc $\Rightarrow$ b.d. mllr	10	78.63	77.44	78.29	79.82	79.31
pdocc $\Rightarrow$ mllr	30	79.22				71.71
pdocc $\Rightarrow$ b.d. mllr	30	77.58				79.15
pd$\rho_{0.2}$	2	79.86			79.64	
pd$\rho_{0.2}$	10	80.94	81.15			81.30
pd$\rho_{0.2}$	30	80.16	80.45			
pd$\rho_{0.4}$	2	79.33	79.83			79.67
pd$\rho_{0.4}$	10	81.05	81.15			81.34
pd$\rho_{0.4}$	30	79.55	77.36			81.01
pd$\rho_{0.6}$	2	79.34	79.79			79.75
pd$\rho_{0.6}$	10	81.05	81.15			81.39
pd$\rho_{0.6}$	30	80.61	81.36			79.31
pd$\rho_{0.8}$	2	79.29	79.76			79.78
pd$\rho_{0.8}$	10	81.19	81.17			81.24
pd$\rho_{0.8}$	30	80.44	81.16			79.98
pd$\rho_{1.2}$	2	79.30	79.61			79.71
pd$\rho_{1.2}$	10	81.20	81.43			81.08
pd$\rho_{1.2}$	30	77.96	78.73			81.45

Method	Eigenspace Dimension	Num Adaptation Sentences 1	2	3	4	5
pd$\rho_{1.4}$	2	79.20	79.52			79.71
pd$\rho_{1.4}$	10	81.15	80.11			80.98
pd$\rho_{1.4}$	30	77.75	76.45			
pd$\rho_{1.6}$	2	79.20	79.40			79.79
pd$\rho_{1.6}$	10	80.27	81.02			81.20
pd$\rho_{1.6}$	30	69.35				82.08
pd$\rho_{1.8}$	2	79.15				79.79
pd$\rho_{1.8}$	10	81.02				81.21
pd$\rho_{1.8}$	30	77.02				
pdρ_{2}	2	79.05				79.89
pdρ_{2}	10	80.72				81.17
pdρ_{2}	30	77.67				76.13

Method	Eigenspace Dimension	Num Adaptation Sentences 6	7	8	9	10
pdocc	10	81.45	81.30	81.27	81.32	81.21
pdocc $\Rightarrow$ map	10	80.28	80.28	79.67	80.86	81.94
pdocc $\Rightarrow$ mllr	10	69.41	74.67	76.50	79.30	81.65
pdocc $\Rightarrow$ b.d. mllr	10	79.45	79.63	80.33	80.69	81.84
pdocc	2	79.66	79.71	79.92	79.87	79.89
pdocc $\Rightarrow$ map	2	79.55		80.38	80.87	81.76
pdocc $\Rightarrow$ mllr	2					81.64
pdocc $\Rightarrow$ b.d. mllr	2					81.99
pdocc	30					81.68
pdocc $\Rightarrow$ mllr	30					81.47
pdocc $\Rightarrow$ b.d. mllr	30					81.42

A.1.3 Eigenvoice Decomposition With Speaker Derived Basis

Method	Eigenspace Dimension	Num Adaptation Sentences 1	2	3	4	5
pdocc	1	66.37	66.27			66.39
pdocc	2	70.44	70.22			70.45
pdocc	3	72.20				72.57
pdocc	4	77.96				78.99
pdocc	5	78.03	78.42			78.99
pdocc	6	77.95				78.94
pdocc	7	78.19				78.56
pdocc	8	77.69				78.92
pdocc	9	77.73				78.97
pdocc	10	77.97	78.38			79.08
pdocc	11	78.19				79.00
pdocc	12	78.74				79.64
pdocc	13	79.05				79.92
pdocc	14	79.59				80.22
pdocc	15	79.33				80.38
pdocc	16	79.81				80.60
pdocc	17	79.78				80.64
pdocc	18	79.89				80.78
pdocc	19	80.16				81.15
pdocc	20	80.34	80.56			81.00
pdocc	21	79.86				81.24
pdocc	22	79.97				80.94

Method	Eigenspace Dimension	Num Adaptation Sentences 1	2	3	4	5
pdocc	23	79.53				80.93
pdocc	24	79.53				80.94
pdocc	25	79.63				81.04
pdocc	26	79.56				81.04
pdocc	27	79.37				81.06
pdocc	28	79.81				81.38
pdocc	29	79.55				81.47
pdocc	30	79.76	81.01	81.43	81.36	81.61
pdocc ⇒ map	30	77.93	77.03	79.09	79.57	80.00
pdocc ⇒ mllr	30	79.76	77.88	73.28	74.43	70.06
pdocc ⇒ b.d. mllr	30	77.93	76.54	77.70	78.53	79.05

Method	Eigenspace Dimension	Num Adaptation Sentences 6	7	8	9	10
pdocc	30	81.34	81.57	81.64	81.64	81.58
pdocc ⇒ map	30	80.84	80.74	80.56	81.19	82.23
pdocc ⇒ mllr	30	68.39	75.08	76.62	79.15	81.32
pdocc ⇒ b.d. mllr	30	79.31	79.97	80.15	80.26	81.77

A.1.4 Eigenvoice Decomposition With PCA(Σ) Derived Basis

Method	Eigenspace Dimension	Num Adaptation Sentences 1	2	3	4	5
pdocc	1	76.26	76.55			76.35
pdocc	2	79.26	79.51			79.66
pdocc	3	80.41				80.42
pdocc	4	80.34				80.46
pdocc	5	80.31	80.34			80.53
pdocc	6	80.22				80.63
pdocc	7	80.56				81.00
pdocc	8	80.75				80.76
pdocc	9	80.80				
pdocc	10	81.26	81.12			81.24
pdocc	11	81.23				81.47
pdocc	12	81.30				81.69
pdocc	13	80.79				81.68
pdocc	14	80.83				80.90
pdocc	15	81.15				81.57
pdocc	16	80.91				81.30
pdocc	17	75.38				77.47
pdocc	18	78.67				81.84
pdocc	19	76.07				79.66
pdocc	20	80.50	79.20			80.01
pdocc	21	80.49				79.07
pdocc	22	79.78				78.98
pdocc	23	78.58				81.75
pdocc	24	76.78				78.55
pdocc	25	75.17				80.52
pdocc	26	78.18				81.60
pdocc	27	76.70				81.21
pdocc	28	76.74				80.83
pdocc	29	76.83				80.11
pdocc	30	76.06	78.06			81.80

A.2 Single Gaussian Triphone

A.2.1 Conventional Techniques

Speaker independent recognition accuracy = 90.10%

Method	Eigenspace Dimension	Num Adaptation Sentences 1	2	3	4	5
map		89.58	89.23	89.62	89.99	90.21
mllr		90.10	90.70	91.40	91.76	92.37
b.d. mllr		89.58	89.99	91.25	92.06	91.96
b.d. mllr ⇒ map		89.58	90.14	91.25	92.06	92.32
mllr ⇒ map		89.58	90.03	90.74	91.86	91.96

Method	Eigenspace Dimension	Num Adaptation Sentences 6	7	8	9	10
map		89.95	90.21	90.10	90.27	90.57
mllr		92.15	92.12	91.99	92.33	92.70
b.d. mllr		92.04	92.29	92.34	92.48	92.93
b.d. mllr ⇒ map		92.04	92.29	92.34	92.48	92.92
mllr ⇒ map		91.74	91.44	91.52	91.85	92.28

A.2.2 Eigenvoice Decomposition With PCA(XX^T) Derived Basis

Method	Eigenspace Dimension	Num Adaptation Sentences 1	2	3	4	5
pdocc	1	90.16	90.17			90.09
pdocc	2	90.74	90.96	90.98	90.98	90.89
pdocc	3	91.40				91.54
pdocc	4	91.50				91.65
pdocc	5	91.63	91.55			91.67
pdocc	6	91.50				91.61
pdocc	7	91.56				91.77
pdocc	8	91.48				91.81
pdocc	9	91.78				91.73
pdocc	10	91.66	91.89	91.95	92.04	91.89
pdocc	11	91.55				91.93
pdocc	12	91.70				91.92
pdocc	13	91.67				92.07
pdocc	14	91.04				92.11
pdocc	15	91.51				91.80
pdocc	16	91.58				91.93
pdocc	17	91.47				91.92
pdocc	18	91.61				91.97
pdocc	19	87.56				92.10
pdocc	20	91.33	91.86			92.10
pdocc	21	91.84				92.29
pdocc	22	91.70				88.10
pdocc	23	91.58				87.83
pdocc	24	91.61				89.81
pdocc	25	91.73				89.92
pdocc	26	91.76				92.17
pdocc	27	89.03				89.34
pdocc	28	90.13				90.63
pdocc	29	90.01				90.99

Method	Eigenspace	Num Adaptation Sentences				
	Dimension	1	2	3	4	5
pdocc	30	90.95	89.72			90.68
pdocc ⇒ map	2	90.68	90.18	90.43	90.81	90.88
pdocc ⇒ map	10	91.63	91.66	92.02	92.49	92.74
pdocc ⇒ mllr	10	91.66	91.97	91.40	92.10	92.37
pdocc ⇒ b.d. mllr	10	91.63	91.73	92.02	92.26	92.47
pd$\rho_{0.2}$	2	90.92				91.04
pd$\rho_{0.2}$	10	91.59				91.97
pd$\rho_{0.2}$	30	89.60				92.25
pd$\rho_{0.4}$	2	90.74				91.06
pd$\rho_{0.4}$	10	91.63				91.96
pd$\rho_{0.4}$	30	89.47				90.06
pd$\rho_{0.6}$	2	90.76				91.02
pd$\rho_{0.6}$	10	91.66				91.97
pd$\rho_{0.6}$	30	88.40				90.30
pd$\rho_{0.8}$	2	90.72				91.07
pd$\rho_{0.8}$	10	91.69				91.96
pd$\rho_{0.8}$	30	91.78				91.89
pd$\rho_{1.2}$	2	90.80				90.85
pd$\rho_{1.2}$	10	91.67				91.99
pd$\rho_{1.2}$	30	90.61				
pd$\rho_{1.4}$	2	90.81				90.84
pd$\rho_{1.4}$	10	90.10				91.97
pd$\rho_{1.4}$	30	89.75				87.54
pd$\rho_{1.6}$	2	90.73				90.84
pd$\rho_{1.6}$	10	91.09				
pd$\rho_{1.6}$	30	89.42				91.39
pd$\rho_{1.8}$	2	90.77				90.80
pd$\rho_{1.8}$	10	89.05				87.42
pd$\rho_{1.8}$	30	89.91				89.98
pdρ_{2}	2	90.68				90.83
pdρ_{2}	10	91.35				91.77
pdρ_{2}	30	86.09				89.43

Method	Eigenspace	Num Adaptation Sentences				
	Dimension	6	7	8	9	10
pdocc	2	90.84	90.84	90.85	90.81	90.80
pdocc	10	91.88	91.95	91.89	91.92	91.95
pdocc	30					92.17
pdocc ⇒ map	2	90.74	90.89	90.80	91.00	91.24
pdocc ⇒ map	10	92.43	92.40	92.41	92.32	92.43
pdocc ⇒ mllr	10	92.26	92.29	92.25	92.49	93.05
pdocc ⇒ b.d. mllr	10	92.26	92.22	92.52	92.59	92.70

A.2.3 Eigenvoice Decomposition With Speaker Derived Basis

Method	Eigenspace	Num Adaptation Sentences				
	Dimension	1	2	3	4	5
pdocc	1	75.66	74.95			75.14
pdocc	2	82.70			82.75	
pdocc	3	85.70				85.78
pdocc	4	90.51				90.50
pdocc	5	90.66			90.47	
pdocc	6	90.37				90.63

Method	Eigenspace Dimension	Num Adaptation Sentences 1	2	3	4	5
pdocc	7	90.18				90.54
pdocc	8	90.18				90.55
pdocc	9	90.29				90.58
pdocc	10	90.39	90.78	90.68		90.68
pdocc	11	90.25				90.68
pdocc	12	90.21				90.76
pdocc	13	90.51				90.92
pdocc	14	90.85				91.19
pdocc	15	90.89				91.22
pdocc	16	90.85				91.21
pdocc	17	90.85				91.28
pdocc	18	90.83				91.32
pdocc	19	91.19				91.36
pdocc	20	91.43			91.54	
pdocc	21	91.26				91.41
pdocc	22	91.22				91.48
pdocc	23	91.15				91.50
pdocc	24	91.03				91.40
pdocc	25	90.94				91.45
pdocc	26	91.30				91.59
pdocc	27	91.04				91.36
pdocc	28	90.89				91.40
pdocc	29	90.94				91.59
pdocc	30	90.95	91.45	91.35	91.61	91.62
pdocc ⇒ map	30	90.81	91.29	91.54	91.88	92.06
pdocc ⇒ mllr	30	90.95	91.47	90.77	91.36	91.91
pdocc ⇒ b.d. mllr	30	90.81	91.29	91.77	92.25	92.44

A.3 6 Mixture Component Triphone

A.3.1 Conventional Techniques

Speaker independent recognition accuracy = 95.38%

Method	Eigenspace Dimension	Num Adaptation Sentences 1	2	3	4	5
map		95.43	95.39	95.52	95.53	95.37
mllr		95.38	95.49	96.09	96.28	96.13
b.d. mllr		95.71	95.95	95.88		
b.d. mllr ⇒ map		95.43	95.50	95.71	95.95	95.72
mllr ⇒ map		95.43	95.45	96.01	96.12	95.86

Method	Eigenspace Dimension	Num Adaptation Sentences 6	7	8	9	10
map		95.42	95.30	95.21	95.27	95.41
mllr		96.14	96.17	96.24	96.25	96.28
b.d. mllr		95.98	95.97	95.94		96.24
b.d. mllr ⇒ map		95.98	95.97	95.94	96.12	96.24
mllr ⇒ map		95.94	96.08	96.31	95.99	96.25

A.3.2 Eigenvoice Decomposition With PCA(XX^T) Derived Basis

Method	Eigenspace	Num Adaptation Sentences				
	Dimension	1	2	3	4	5
pdocc	1	95.47			95.39	
pdocc	2	93.16	92.30			95.02
pdocc	3	90.83				94.93
pdocc	4	95.87				95.65
pdocc	5	92.03	92.99			95.65
pdocc	6	93.85				95.91
pdocc	7	96.08				93.89
pdocc	8	96.21				96.12
pdocc	9	96.09				93.12
pdocc	10	96.10				96.14
pdocc	11	93.67				96.21
pdocc	12	93.86				95.72
pdocc	13	96.21				96.19
pdocc	14	95.93				96.24
pdocc	15	96.10				96.19
pdocc	16	94.48				96.23
pdocc	17	96.01				96.20
pdocc	18	93.56				96.14
pdocc	19	95.23				93.38
pdocc	20	95.91				96.24
pdocc	21	95.67				94.13
pdocc	22	96.08				96.27
pdocc	23	94.52				95.84
pdocc	24	95.56				96.20
pdocc	25	96.12				96.13
pdocc	26	96.16				96.14
pdocc	27	96.16				96.16
pdocc	28	96.17				96.13
pdocc	29	96.12				93.68
pdocc	30	96.14				96.17

A.3.3 Eigenvoice Decomposition With Speaker Derived Basis

Method	Eigenspace	Num Adaptation Sentences				
	Dimension	1	2	3	4	5
pdocc	1	93.33	93.33			93.20
pdocc	2	93.93	93.97		93.74	93.74
pdocc	3	94.34				94.28
pdocc	4	94.57				94.45
pdocc	5	94.61	94.64			94.53
pdocc	6	94.74				94.68
pdocc	8	95.09				94.97
pdocc	10	95.26	95.72	95.17	95.27	95.16
pdocc	12	95.67				95.78
pdocc	14	95.73				95.84
pdocc	16	95.78				95.76
pdocc	18	95.78				95.84
pdocc	20	95.83	95.91			
pdocc	22	95.84				95.84
pdocc	24	95.82				95.95
pdocc	26	96.04				96.06
pdocc	28	96.04				96.19
pdocc	30	95.99	96.21	96.24	96.35	96.25
pdocc $\Rightarrow$ map	2	94.04	93.82	94.12	94.03	94.00

Method	Eigenspace	Num Adaptation Sentences				
	Dimension	1	2	3	4	5
pdocc ⇒ map	10	95.27	95.32	95.37	95.54	95.50
pdocc ⇒ mllr	10	95.26	95.61	96.04	96.01	96.13
pdocc ⇒ b.d. mllr	10	95.27	95.42	95.84	95.93	95.83
pdocc ⇒ map	30	95.95	96.17		96.32	95.84
pdocc ⇒ mllr	30	95.99	96.20	96.17	96.20	96.38
pdocc ⇒ b.d. mllr	30	95.97	95.90	95.99	96.23	
pd$\rho_{0.2}$	2	94.00				93.74
pd$\rho_{0.2}$	10	95.26				95.15
pd$\rho_{0.2}$	30	96.05				96.21
pd$\rho_{0.4}$	2	93.93				93.75
pd$\rho_{0.4}$	10	95.26				95.09
pd$\rho_{0.4}$	30					
pd$\rho_{0.6}$	2	93.87				93.71
pd$\rho_{0.6}$	10	95.31				95.11
pd$\rho_{0.6}$	30	95.99				96.25
pd$\rho_{0.8}$	2	93.87				93.70
pd$\rho_{0.8}$	10	95.26				95.13
pd$\rho_{0.8}$	30	95.98				96.27
pd$\rho_{1.2}$	2	93.90				93.79
pd$\rho_{1.2}$	10	95.24				95.16
pd$\rho_{1.2}$	30	96.29				
pd$\rho_{1.4}$	2	93.79				93.77
pd$\rho_{1.4}$	10	95.19				95.12
pd$\rho_{1.4}$	30	96.06				95.60
pd$\rho_{1.6}$	2	93.81				93.75
pd$\rho_{1.6}$	10	95.26				95.06
pd$\rho_{1.6}$	30	96.34				
pd$\rho_{1.8}$	2	93.78				93.67
pd$\rho_{1.8}$	10	95.32				
pd$\rho_{1.8}$	30	96.02				96.34
pdρ_{2}	2	93.75				93.67
pdρ_{2}	10	95.28				
pdρ_{2}	30	96.32				

Method	Eigenspace	Num Adaptation Sentences				
	Dimension	6	7	8	9	10
pdocc	2	92.88	93.77	93.78	93.74	93.72
pdocc	10	95.19	95.15	94.57		95.15
pdocc	30	96.23	96.17	94.94	96.28	96.25
pdocc ⇒ map	2	93.71	93.63	93.67	93.46	93.72
pdocc ⇒ map	10	95.23	95.37	95.41		95.42
pdocc ⇒ mllr	10	96.16	96.17	96.16	96.14	96.17
pdocc ⇒ b.d. mllr	10	95.98	95.97	96.12	96.27	96.39
pdocc ⇒ map	30	96.00	96.16	95.91	95.87	96.23
pdocc ⇒ mllr	30	96.32	96.49	96.40	96.42	96.51
pdocc ⇒ b.d. mllr	30	96.35	96.25	96.31		96.36