Discriminative Methods in HMM-based Speech Recognition

Valtcho Valtchev

St. John's College

March 1995

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

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where stated. It has not been submitted in whole or part for a degree at any other University. The length of this thesis including footnotes and appendices is approximately 55,000 words.

Summary

Conventional speech recognition systems require information from two knowledge sources - a family of *acoustic models* and a *language model*. The acoustic models incorporate knowledge extracted from the speech waveform and they are commonly based on hidden Markov models (HMMs). HMMs have been used successfully for speech recognition for many years, however, in many respects the assumptions behind the HMM framework are poor. The following issues can be considered

- HMMs are usually trained according to the Maximum Likelihood estimation (MLE) procedure whose optimality, in the sense of providing the lowest possible error rate, is based on assumptions which are never satisfied in practice. Discriminative training techniques remove the need for these assumptions and attempt to optimise an information-theoretic criterion which is related to the performance of the recogniser. Unfortunately, discriminative methods require substantially more computation than MLE and many previous implementations of such techniques have been based on the somewhat unreliable steepest-descent procedure.
- In the HMM framework, the acoustic observations are assumed to be independent of each other, hence, speech dynamics cannot be modelled directly. Such information is typically provided in "canned" form by extending the feature vectors to accommodate differential components which reflect the changes in the standard coefficients. Although this approach results in improved recognition performance, it entails an increased number of model parameters and consequently longer training and recognition times. Another problem with differential coefficients is the assumption that the parameter slope/curvature are the only useful features.

In this dissertation we describe an implementation of the Maximum Mutual Information estimation (MMIE) discriminative training algorithm, where an approximate second-order optimisation scheme is employed to update the parameters of the HMMs. This algorithm is shown to provide improved recognition performance, achieved after a small number of iterations. A modification of the MMIE algorithm is also discussed whereby a different weighting is given to each utterance in the training set based on the mutual information measure.

The problem of providing compact and informative feature vectors is introduced and discussed in terms of feature selection and feature extraction algorithms. In this respect, the Minimal Mutual Information Change (MMIC) feature selection algorithm is proposed where the change in the mutual information criterion is used to rank-order the components of the feature vector.

Feature extraction techniques are investigated through the introduction of adaptive input transformations in the conventional HMM framework. Transformations of different topologies are initialised to perform meaningful parameter transformations. Subsequently, during training, the parameters of the transformations are optimised simultaneously with the HMM parameters according to a discriminative objective criterion.

The discriminative training algorithms are evaluated on a British English E-set task, an American alphabet recognition task, and a large continuous phone recognition task (TIMIT).

Keywords: speech recognition, hidden Markov models, discriminative training, feature extraction, feature selection, adaptive input transformations.

Acknowledgements

Thanks are due, first and foremost, to my supervisor Prof. Steve Young for his invaluable guidance, enormous support and encouragement. His blend of constructive criticism, enthusiasm and dedication to the cause have been particularly helpful in stimulating my research.

I acknowledge the Benefactors' scholarship from St. Johns College which enabled me to undertake this research in the period October 1990 - September 1993. I would also like to thank the Engineering Department and St. John's college who provided funding for attending several conferences.

Thanks to Steve Young and Phil Woodland for providing the outstanding HTK toolkit. A special thanks also goes to Sadik Kapadia for the many helpful discussions on discriminative training and other topics on speech recognition. Furthermore, I would like to acknowledge him for the provision of the `HNRest` re-estimation tool and associated library modules which, together with HTK, have saved me a substantial amount of time and effort in getting my experiments to work and at the same provided a flexible platform to extend and test the new ideas on.

Many thanks to my present bosses Steve and Phil for allowing me to have the extra time to finish off this dissertation. I would also to like to express my gratitude to Richard Prager, Patrick Gosling, Tony Robinson, Carl Seymour, Andrew Gee and everybody else involved in maintaining the excellent computing facilities which were crucial for carrying out my research.

I would like to thank all colleagues in the Speech Vision and Robotics (SVR) research group for their enjoyable company. In particular, many thanks to Chris Leggetter, Mark Gales, Phil Woodland, Julian Odell and Tony Robinson for the helpful suggestions on various topics relating to my work.

Finally, thanks for proofreading various parts of the dissertation are due to Kate Knill, Chris Leggetter, Mark Gales and Dan Kershaw.

Contents

List of Figures

List of Tables

Notation

Commonly used symbols and notation

$\boldsymbol{x}$	a vector (lower case bold)
$\boldsymbol{x}'$	the vector transpose of $\boldsymbol{x}$
$\hat{\boldsymbol{x}}$	an estimate of $\boldsymbol{x}$
$\boldsymbol{A}$	a matrix (upper case bold)
$\boldsymbol{A}'$	the matrix transpose of $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	the inverse of matrix $\boldsymbol{A}$
$\lvert\boldsymbol{A}\rvert$	the determinant of matrix $\boldsymbol{A}$
$tr(\boldsymbol{A})$	the trace of matrix $\boldsymbol{A}$
$\boldsymbol{\mu}$	the mean vector of a Gaussian distribution
$\boldsymbol{W}$	the covariance matrix of a Gaussian distribution
λ	the parameter set of the acoustic/language model

Acronyms

HMM	Hidden Markov Model
CMD-HMM	Continuous Mixture Density HMM
MLE	Maximum Likelihood Estimation
CMLE	Conditional Maximum Likelihood Estimation
MMIE	Maximum Mutual Information Estimation
SMMIE	Sigmoid Maximum Mutual Information Estimation
MMIC	Minimal Mutual Information Change
DCT	Discrete Cosine Transform
LDA	Linear Discriminant Analysis
PCA	Principal Components Analysis
MFCC	Mel Frequency Cepstral Coefficient
SI	Speaker Independent
SD	Speaker Dependent
LVCSR	Large Vocabulary Continuous Speech Recognition

Chapter 1

Introduction

Speech is the most natural form of human communication. The goal of automatic speech recognition (ASR) is to develop systems that are capable of transcribing natural speech. Speech recognition devices find widespread use in many applications including information retrieval, data entry and general man-machine communication thus aiding productivity and convenience. Research in ASR originated as early as 1950's and nearly forty years later speech recognition is still considered unsolved. ASR is a very difficult task due to the large problem dimension and generally ambiguous nature of speech. Although there are no complete solutions, the performance of ASR systems has improved dramatically in the last few years[1]. Nevertheless, the performance of the resulting systems is still not comparable to that achieved by human beings. Furthermore, there is a variety of small-vocabulary speech recognition tasks where high recognition accuracy is the only requirement to warrant their immediate appearance in our every-day lives.

The performance of current speech recognition devices is entirely dependent on the ability of the system to discriminate between the different sounds of speech. Most speech recognition systems make use of two major knowledge sources: a family of *acoustic models* and a *language model*. The acoustic models are commonly based on hidden Markov models. The focus of this thesis, as suggested by its title, is on enhancing the discriminative ability of these acoustic models with the aim of improving the overall recognition performance of the system.

1.1 Speech recognition systems

The structure of a conventional speech recognition system is shown in figure 1.1. The acoustic preprocessor transforms the speech waveform into a sequence of acoustic observations used during classification. Using knowledge from the acoustic and language models in con-

[1] When this research began in 1990, the DARPA Resource Management task (RM) with a vocabulary of $1,000$ words was considered a difficult large vocabulary continuous speech recognition (LVCSR) task. Most recently [110], the HTK LVCSR system using a vocabulary in excess of $65,000$ words achieved a lower error rate in the November 1994 ARPA Wall Street Journal (WSJ) evaluation than was achieved by the best system in the first RM evaluation.

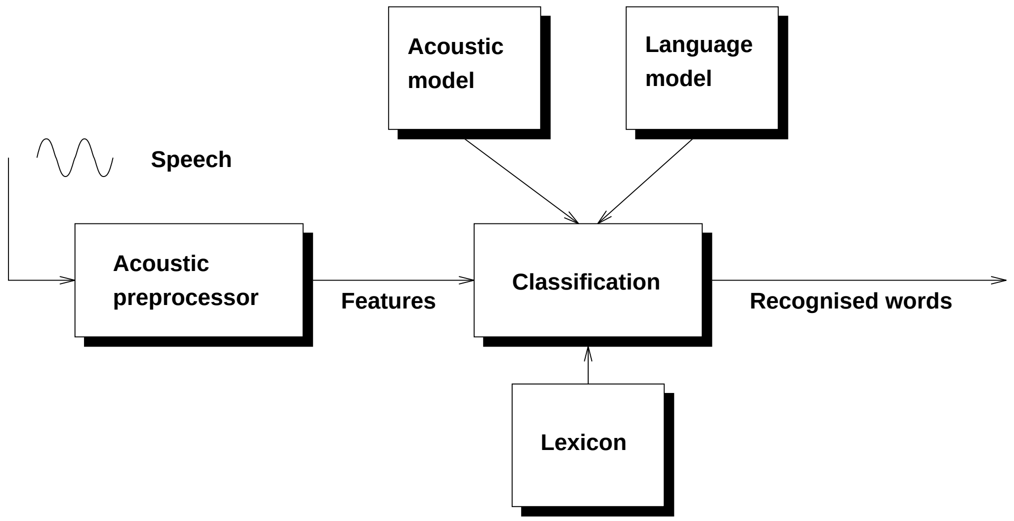

Figure 1.1: Block structure of a conventional speech recogniser

junction with the language constraints provided by the lexicon of the task, the classification procedure outputs the most likely word sequence hypothesised for the input sequence of observations.

The usability of a speech recognition device is dependent on its speed and performance. Acceptable recognition performance is often achieved by limiting the size of the lexicon, constraining the way the user speaks and limiting the number of speakers the system is able to recognise. Even constrained systems do not offer perfect recognition and much of the research effort is still concentrated on improving the recognition performance of systems with small but useful vocabularies such as digits and spoken letters. Speech recognition systems vary in their speed and performance and their most important characteristics can be classified as follows:

- *Isolated or continuous speech.* This property refers to the mode of speech that the system is designed to handle. Continuous speech recognition (CSR) is much more difficult than isolated word recognition (IWR) due to the absence of word boundary information, increased co-articulation and confusability. Another recently introduced task is "keyword" spotting which appears to be as difficult as CSR. The aim is to detect a relatively small number of "keywords" in continuous speech and at the same time ignore all other words and non-speech sounds.

- *Vocabulary size.* The vocabulary size is the most important factor affecting the speed and performance of an ASR system. A larger vocabulary is likely to contain more acoustically confusable words. Increasing the number of words will typically require more training data and longer recognition time. In general, the performance of an ASR system is significantly affected by the confusability in the vocabulary and this can be a problem in even small vocabulary systems. A typical example is the recognition of spoken letters from the English alphabet where the members of the E-set {"B", "C", "D", "E", "G", "P", "T", "V"} are the major source of errors.

- *Speaker dependence/independence.* Speaker dependent (SD) systems are trained to recognise speech from a single speaker. Speaker independent systems are capable of recognising speech from any new speaker. Generally, SD systems achieve better recognition performance than SI systems because of the limited variability in the speech signal coming from the same speaker. Speaker independent systems require complex acoustic models for modelling between-speaker variability with adverse effects on speed and performance. An elegant solution to the problem is offered by adapting a generic SI system on-the-fly to the speech characteristics of the current speaker.

Speaker independent large vocabulary continuous speech recognition (LVCSR) systems are available at present as prototypes in a number of research laboratories. Examples include BYBLOS from BBN [64], SPHINX [66] and SPHINX-2 [2] from Carnegie-Mellon University, the HTK system [110] and the ABBOT system [48] from Cambridge University.

1.2 The acoustic modelling problem

1.2.1 Parameter estimation

Speech recognition requires dealing with uncertainty and handling variability from a variety different sources such as speaker, context, environment, etc. Current speech recognition systems employ statistical methods that are capable of learning acoustic-phonetic knowledge from samples of speech. The most common statistical approach is based on hidden Markov models. This approach is popular since it provides a tractable framework for modelling the variation in the acoustic signal as a sequence of events of variable duration. At the same time the models are simple enough so that their parameters can be automatically deduced from samples of speech. The quality of the acoustic model used in a speech recognition system is the major factor in determining the system's ability to distinguish between the various acoustic classes chosen as the basic units of recognition. Furthermore, an inadequate acoustic model will limit the potential gains from the introduction of other knowledge sources such as language and prosody.

In the HMM framework, the parameters of the acoustic model are estimated from large samples of acoustic data. Ideally, the parameter estimation procedure should yield an HMM set which minimises the probability of error during recognition. This can be achieved by maximising the probability of each utterance given the corresponding class model and at the same time minimising the probability of the utterance given all other models in the system. Conventional parameter estimation techniques attempt to satisfy the former requirement, the latter requirement is simply ignored. One such technique is Maximum Likelihood estimation, whereby the parameters of the models are re-estimated from in-class data only. A major contributing factor to the success of HMM-based speech recognition is the powerful Baum-Welch training algorithm for Maximum Likelihood estimation (MLE) training. One problem with MLE is that it has no obvious relationship with the objective of minimising the recognition error rate. Furthermore, the rationale behind the use of MLE training relies on the assumption that the underlying models are the "true" models of speech. However, it is well known that from a speech production point of view, HMMs are a notoriously poor model of speech.

Discriminative training techniques remove the need for the assumption that the underlying models are correct and attempt to improve recognition performance by adjusting the model parameters themselves. Maximum Mutual Information estimation (MMIE) is a discriminative training technique proposed by Bahl et al. [6] as an alternative to MLE. The approach seems plausible since it directly maximises the quantities which are used during recognition. Unfortunately, there is no equivalent of the Baum-Welch algorithm for MMIE and the standard training procedure is often based on the somewhat unreliable gradient search techniques. Several researchers have reported improvements in recognition rate using the MMIE approach. The most prominent studies are the work published by Brown [22] on an American English E-set recognition task and most recently, the successful application of MMIE training to continuous digit recognition investigated by Normandin [84]. In general, discriminative training techniques are much more difficult to implement than conventional MLE training. At the same time, they require substantially more computation than MLE training since in order to achieve discrimination they consider all acoustic models at the same time. Thus, it is often perceived that the marginal gains in performance provided by discriminative training do not justify the introduction of gradient-based optimisation techniques into otherwise robust, simple and efficient re-estimation algorithms.

1.2.2 The acoustic preprocessor

The acoustic preprocessor is a device which converts the analogue audio signal into a sequence of observations which can be modelled by the HMM's. The extracted feature vectors should contain as much information as possible about the linguistic content of the acoustic signal whilst being reasonably compact and free of redundant detail. The quality of the preprocessing has a considerable impact on the overall performance of the speech recognition system. Compact feature vectors will contribute to faster training and recognition times whilst well conveyed linguistic detail will aid discrimination. Conventional preprocessors perform simple types of filtering and data compression based on Fourier analysis and linear predictive coding (LPC). Feature extraction is then performed through a parameter transformation into the "cepstral" domain using the discrete cosine transform (DCT).

The most significant recent development in terms of feature sets is the introduction of dynamic parameters [45]. Performance improvements from using derivative parameters as part of the feature vectors demonstrate a major limitation of HMMs. Although, the information contained in the differential parameters is directly available from the basic feature set, it has to be provided explicitly for the HMMs to make use of it. This limitation arises as a consequence of the so-called *observation independence* assumption in the HMM framework, whereby each observation vector is modelled independently of all past and future vectors. However, enhancing existing feature sets through the addition of new feature derivatives entails more model parameters and increased processing time. Hence, it is clearly desirable to rank-order the elements of the feature sets and then select a compact subset which yields minimal decrease in the overall performance of the system. This is usually achieved by using feature selection and feature extraction algorithms. One problem with these schemes is that they are not directly related to the objective of minimising the overall error rate. Furthermore, once derived, the linear transformations used to perform feature extraction remain unchanged throughout the training process.

1.3 This study

In this thesis we present an implementation of the MMIE algorithm for continuous mixture density hidden Markov models (CMD-HMMs). The optimisation procedure uses an approximate second order method (*QuickProp*) which only requires first derivative information. An extension of the MMIE algorithm is also presented, whereby the parameter derivatives from each individual training utterance are adjusted according to a non-linear weighting function which is designed to give higher weighting to utterances near the decision boundary. In an attempt to provide compact and informative feature sets, the Minimal Mutual Information Change (MMIC) feature selection algorithm is proposed where the mutual information measure is used to rank-order the components in the feature set. Finally, the conventional HMM framework is enhanced to incorporate a variety of adaptive input transformations which transform the observation vectors before they enter the output distributions of the HMMs. During training, the parameters of the transformations are optimised together with the HMM parameters according to a discriminative objective function.

In this study, the above techniques are investigated in the context of SI isolated word recognition and SI continuous phone recognition. These tasks have been chosen for the following reasons

1. Recognition experiments are performed on standard databases which have been studied by other researchers in the speech recognition community with numerous results available for comparison.

2. The tasks are simple enough to guarantee a reasonable development and evaluation cycle whilst, and at the same time, being representative of the typical problems which the discussed methods attempt to confront.

Appendix A describes the three databases used in this study, together with previous results and details about the pre-processing.

1.4 Outline of this thesis

Chapter 2 provides a concise introduction to the theory and application of hidden Markov models to automatic speech recognition. In order to provide an easier understanding of the presented experimental work, several issues related to their implementation are also discussed. **Chapter 3** reviews the various objective criteria which can be used to optimise the parameters of the acoustic models. The rationale behind two such techniques, MLE and MMIE, are described in detail. This chapter also presents a modification to the MMIE objective function which gives different weighting to different utterances in the training set based on the mutual information criterion. **Chapter 4** describes the implementation of discriminative training in the context of CMD-HMMs. **Chapter 5** presents the experimental evaluation of discriminative training on three speech databases. **Chapter 6** introduces the background theory of feature selection and feature extraction algorithms. In **Chapter 7** a feature selection algorithm based on the mutual information measure is proposed and evaluated on the TIMIT continuous phone recognition task. **Chapter 8** introduces the theory and application of adaptive input transformations in the CMD-HMM framework. A variety of input transformations are derived and evaluated on the TIMIT continuous phone recognition task. **Chapter 9** concludes this study by discussing the effectiveness of the evaluated discriminative methods. Suggestions for future research in discriminative training are also given.

Chapter 2

Basic Concepts

Conventional speech recognition systems employ statistical models to model various sources of information. In particular, recognising speech requires the integration of information from two models - the *acoustic model* and the *language model*. The acoustic model incorporates knowledge extracted from the speech waveform and its design has a considerable impact on the performance of a speech recognition system. The power of representation in an acoustic model lies in its structure and parameters. Acoustic models are commonly based on hidden Markov models. This chapter introduces the theory of hidden Markov models as applied to the speech recognition problem. Although, the theory is well documented elsewhere [91, 117] the major steps in the derivation of the parameter estimation formulae are presented here to serve as a reference for later chapters in this thesis.

2.1 Introduction

The performance of a speech recognition device depends on the system's ability to reduce uncertainty about the identity of a spoken word using information from the acoustic signal and past word sequences.

The speech recognition problem can be viewed as a problem in communication theory [104]. A spoken string of words of known identity w is viewed as passing through an acoustic channel model which produces a sequence of acoustic observation symbols a. The term "word" is used to denote the basic unit of speech such as sentence, phrase, word, phoneme. An acoustic observation a is a sequence feature vectors extracted from the acoustic signal generated by the speaker while uttering w. The joint probability of words w and acoustics a is

$$P(w, a) = P(a|w)P(w) = P(w|a)P(a)$$

The language model component, $P(w)$, provides information about the word sequence in w. The conditional distribution $P(a|w)$ of acoustics given words describes the acoustic channel model and the conditional distribution $P(w|a)$ defines a probabilistic decoder. For a known sequence of observations, the marginal distribution $P(a)$ is assumed to be constant since it does not depend on the models.

The above definition of the speech recognition problem translates into the following practical considerations:

- *Acoustic model structure* - The acoustic model is a probabilistic function which models the phonological and acoustic-phonetic variation in the speech signal. It is extremely difficult for a human expert to devise an accurate and complete acoustic model due to partial knowledge and inability to express such knowledge in an algorithmic form. For this reason an acoustic model is defined as a family of parametric distributions with parameter vector λ. The chosen family of distributions should be based on true assumptions about speech and have a relatively small number of free parameters. The value of λ identifies a unique acoustic model from the family and is usually estimated from a large sample of speech data.

- *Parameter estimation* - The ultimate goal in parameter estimation is to find a parameter vector λ which produces a decoder with the lowest possible recognition error rate. This is done by optimising some objective function $\mathcal{F}(\lambda)$ which relates to the decoder's performance. The objective function should be such that when $\mathcal{F}(\hat{\lambda}) > \mathcal{F}(\lambda)$ then $\hat{\lambda}$ will produce a better decoder than λ. Once $\mathcal{F}(\lambda)$ has been chosen, the second problem is to find the parameter set λ which maximises it. Complex acoustic models typically employ several thousand parameters which makes it very unlikely that a globally optimal λ will be found. This means that even with a good function, it is possible to obtain unsatisfactory results if the estimation procedure converges to a bad local maximum.

- *Probabilistic decoder* - A speech decoder is a device which attempts to find the identity of a word from its acoustic representation. Using the above notation this can be defined as a transformation $a \Rightarrow \hat{w}$. If the chosen identity $\hat{w}$ is different from the actual identity of the spoken word w then there is a decoding error. The probability of making an error is the most important factor in choosing the decoder. The optimal decoder with regard to minimising the probability of error is the maximum *a posteriori* (MAP) decoder, where w is chosen such that

$$\hat{w} = \arg\max_{w} P(w|a) = \arg\max_{w} P(a|w)\frac{P(w)}{P(a)} \tag{2.1}$$

2.2 Hidden Markov models

Hidden Markov models (HMMs) constitute the most successful approach developed so far for modelling the statistical variations of speech. Since their introduction in the 1970's [8, 56] HMMs have been tested on a wide selection of speech recognition tasks. The HMM approach provides a framework which includes an automatic supervised training algorithm with mathematically proven convergence (the *Baum-Welch algorithm*), and an efficient decoding scheme for use in recognition (the *Viterbi algorithm*). The models have the ability to generalise from a large amount of training data by making structural assumptions and adjusting model parameters so as to optimise a meaningful objective function. The HMM

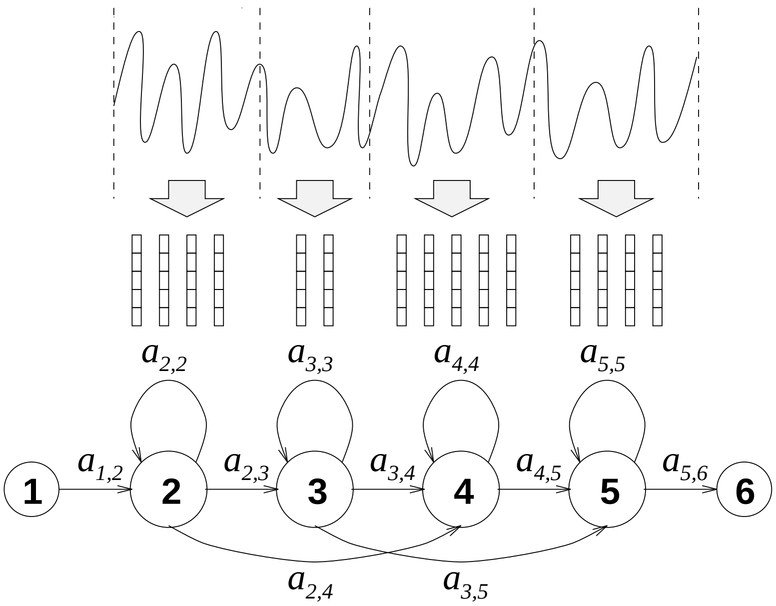

Figure 2.1: A hidden Markov model with 4 emitting states

theory is well documented in the literature [91] and only a short description is included in this chapter for completeness. Although the basic HMM concepts have remained unchanged, different variations of the basic model have been used by different research sites.

2.2.1 Definition

Speech production is a non-stationary process, however, within the HMM framework it is assumed that over a short period of time its acoustical realisations do not vary significantly from sample to sample. A Markov model is a finite N-state machine which changes state according to a probabilistic transition function. Thus, the input utterance is modelled as a sequence of discrete stationary states, with instantaneous transitions between them. Transition probabilities are non-negative and the sum of all transitions that leave a state is unity. Furthermore, transitions from one state to another satisfy the following constraint

$$P(\theta_t|\theta_1^{t-1}) = P(\theta_t|\theta_{t-1}) \tag{2.2}$$

where θ_t is the state at time t and the θ_1^{t-1} denotes the sequence $\{\theta_1, \theta_2, \ldots, \theta_{t-1}\}$. The probability that the Markov chain is in state θ_t at time t depends only on the state θ_{t-1} and is conditionally independent of the past acoustic vector sequence. The transition function is described by a $N \times N$ matrix of discrete probabilities $a_{i,j}$ where

$$a_{i,j} \triangleq P(\theta_t = j|\theta_{t-1} = i), \tag{2.3}$$

The state sequence θ can be viewed as a sequence of random variables. In a hidden Markov model, there is a sequence of random variables, $\boldsymbol{o}_1^T = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_T\}$, which is a probabilistic function of the underlying stationary Markov chain θ. Then, θ is a hidden Markov model for the sequence of random variables, $\boldsymbol{o}_1^T$, provided that for each t, $\boldsymbol{o}_t$ is defined over continuous space, and

$$b(\boldsymbol{o}_t|\boldsymbol{o}_1^{t-1}, \theta_1^t) = b(\boldsymbol{o}_t|\theta_t) = b_{\theta_t}(\boldsymbol{o}_t) \tag{2.4}$$

where $b_j(\boldsymbol{o})$ is the probability density (p.d.) of $\boldsymbol{o}$ in state j. The probability in equation 2.4 is assumed to be independent of t. This property is commonly referred to as the *observation-independence* assumption.

The sequence $\boldsymbol{o}_1^T$ is the observed output of the hidden Markov model. The state sequence θ is not observed, it is hidden. The usual definition of a hidden Markov model also includes an initial state probability vector $\boldsymbol{\pi}_i$ which describes the likelihood of occupying each state in the model prior to generating any observable output [91]. This is somewhat inconvenient to handle when many HMMs are linked together. Instead, the basic HMM structure can be extended to include an initial non-emitting state θ_0 and a final non-emitting state θ_{T+1}. Thus, initial state probabilities are implicitly incorporated into the state transition vector of θ_0.

2.2.2 Output distributions

Although the structure of the acoustic models has been determined nothing has been said so far about the properties of the state output distributions $b_j(\boldsymbol{o}_t)$. In general, the state output distributions can be either discrete or continuous however, parameter tying can lead to the so-called semi-continuous output distributions [50], which can be considered as a mixture of the two [13].

2.2.2.1 Discrete input features

In this case $b_j(\boldsymbol{o}_t)$ is described by a V-dimensional vector of scalars. The system has to incorporate a Vector Quantiser (VQ) which will assign one of V unique labels to each observation vector. The important quality of such a representation is that discrete output distributions do not assume any specific form of the density functions although modelling inaccuracies are inherent in the VQ. This is implicitly dealt with in the choice of distance metric for the clustering procedure in the VQ (e.g. the Euclidean distance measure as used in the k-means clustering algorithm). Most early speech recognition systems [56, 92] utilised such densities. In an attempt to reduce the quantisation distortion for large observation vectors following the introduction of differential parameters, Lee [66] utilised multiple independent code-books in the VQ. All components were assumed independent and their probabilities were simply multiplied to give the probability of the compound vector.

2.2.2.2 Continuous input features

In this case, $b_j(\boldsymbol{o}_t)$ is a general parametric distribution of a predetermined form. The most commonly used distribution is the continuous Gaussian density function defined as

$$\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}, \boldsymbol{W}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{W}|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{o}-\boldsymbol{\mu})'\boldsymbol{W}^{-1}(\boldsymbol{o}-\boldsymbol{\mu})}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{W}$ are the mean vector and the covariance matrix respectively of the distribution and D is the dimensionality of the observation vectors. To reduce the number of free parameters it is often assumed that the components of the feature vector are uncorrelated i.e. the off-diagonal elements in the covariance matrix are set to zero. Unfortunately, the "true" parameter vector distributions will often have complex shapes, and in such cases a single Gaussian density with diagonal covariance matrix may prove inadequate. This is especially true in speaker independent systems trained on both male and female data. The modelling inadequacies of the Gaussian model will depend to a large extent on the speaker normalisation capabilities of the preprocessor and the structure of the Markov models [22]. In order to obtain more accurate approximations, it is common to use mixtures of Gaussian densities

$$b_j(\boldsymbol{o}_t) = \sum_{m=1}^{M} c_{j,m}\mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_{j,m}, \boldsymbol{W}_{j,m}) = \sum_{m=1}^{M} c_{j,m} b_{j,m}(\boldsymbol{o}_t)$$

where M is the number of mixture components and $c_{j,m}$ is the mixture weight for the m^{th} mixture component in state j. Note that the mixture distributions can be implemented by having several states in parallel with mixture weights as state transitions and single mixture components associated with corresponding states. Other distribution forms have also been proposed and used. These include the Gaussian autoregressive mixture density [60], the Richter mixture density [96] and the Laplace mixture density [81]. Although it can be proved that a mixture of Gaussians can model any kind of distribution it is not clear how detailed each model should be. However, an existing mixture density can be "up-mixed" as more training data becomes available by cloning the dominant mixture component (see section 2.4.2). Hence, in practice, mixture Gaussian densities provide great flexibility.

The choice between discrete and continuous distributions depends on the following factors:

- continuous distributions provide more accurate modelling and the structure of the model can be dynamically altered to achieve the optimal match between model complexity and available amount of training data;
- the performance of discrete models depends to a large extent on the design of the vector quantiser;
- for the discrete models, the probability calculation can be replaced by a table lookup, whilst the continuous densities will require a significant amount of computation.

Since accuracy is the major concern here, continuous mixture density HMMs (CMD-HMMs) will be used exclusively throughout this thesis.

2.2.3 Probability calculation

Now that the structure of the acoustic models has been determined we can proceed to define the various probabilistic quantities necessary for the parameter estimation and evaluation of the models. In a speech recognition task, the input utterance is transformed into the sequence of T observation vectors $\boldsymbol{o}_1^T = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_T\}$. Given an acoustic model with parameter set λ, the probability that this model generates the observed sequence can be expressed as

$$\begin{aligned} P_\lambda(\boldsymbol{o}_1^T) &= \sum_{\theta \in \Theta} P_\lambda(\theta) P_\lambda(\boldsymbol{o}_1^T | \theta) \\ &= \sum_{\theta \in \Theta} \prod_{t=1}^{T+1} a_{\theta_{t-1},\theta_t} \prod_{t=1}^{T} b_{\theta_t}(\boldsymbol{o}_t) \end{aligned} \tag{2.5}$$

where Θ is the set of all possible state sequences in the model, θ is a particular state sequence and θ_t is the state occupied at time t. It is often observed that the quantity $P_\lambda(\boldsymbol{o}_1^T)$ is dominated by the largest term in the summation. Consequently, in many cases it is more convenient to compute

$$P_\lambda(\boldsymbol{o}_1^T) = \max_\theta \left\{ \prod_{t=1}^{T+1} a_{\theta_{t-1},\theta_t} \prod_{t=1}^{T} b_{\theta_t}(\boldsymbol{o}_t) \right\} \tag{2.6}$$

In practice, expressions 2.5 and 2.6 cannot be used directly since the number of different state sequences through the model grows exponentially with the duration of the utterance. Following Baum [12], we can define a set of forward/backward probabilities for a hidden Markov model with non-emitting initial and final states. The forward probabilities $\alpha_j(t)$ are defined as follows

$$\alpha_j(t) = P_\lambda(\boldsymbol{o}_1^t, \theta_t = j)$$

where $\alpha_j(t)$ is the joint probability that the model with parameters λ generates the output sequence $\{\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_t\}$ through a state sequence which ends in state j. The forward probabilities can be generated in an incremental fashion using the following rules[1]

1. Initialisation

$$\alpha_j(0) = \begin{cases} 1 & \text{for } j = 1 \\ 0 & \text{for } 1 < j < N \end{cases}$$

2. Recursion (for $1 \leq t \leq T$)

$$\begin{aligned} \alpha_1(t) &= 0 \\ \alpha_j(t) &= \left[\sum_{i=1}^{N-1} \alpha_i(t-1) a_{i,j} \right] b_j(\boldsymbol{o}_t) \qquad \text{for } 1 < j < N \end{aligned} \tag{2.7}$$

3. Termination

$$\alpha_N(T) = \sum_{i=2}^{N-1} \alpha_i(T) a_{i,N}$$

[1] As mentioned earlier, states 1 and N are non-emitting and transitions $a_{1,j}$ are equivalent to π_j as discussed in [91].

Similarly, the backward probabilities $\beta_i(t)$ are defined as

$$\beta_i(t) = P_\lambda(\boldsymbol{o}_{t+1}^T | \theta_t = i)$$

The quantity $\beta_i(t)$ is the probability that the model will generate the sequence $\boldsymbol{o}_{t+1}^T$ starting with a transition from state i. The backward probabilities are computed recursively starting from the end of the observation sequence according to

1. Initialisation
$$\beta_i(T) = \begin{cases} 1 & \text{for } i = N \\ a_{i,N} & \text{for } 1 < i < N \end{cases}$$

2. Recursion
$$\beta_i(t) = \sum_{j=2}^{N-1} a_{i,j} b_j(\boldsymbol{o}_{t+1}) \beta_j(t+1) \qquad \text{for} \qquad \begin{array}{c} 1 < i < N \\ 1 \leq t < T \end{array}$$

3. Termination
$$\beta_1(0) = \sum_{j=2}^{N-1} a_{1,j} b_j(\boldsymbol{o}_1) \beta_j(1)$$

Using the forward/backward probabilities we can define several important quantities used in the parameter estimation procedure. The overall probability of an utterance can be efficiently computed as

$$P_\lambda(\boldsymbol{o}_1^T) = \alpha_N(T) = \beta_1(0)$$

The joint probability of observing the sequence $\boldsymbol{o}_1^T$ and occupying state j at time t is given by

$$P_\lambda(\boldsymbol{o}_1^T, \theta_t = j) = \alpha_j(t) \beta_j(t)$$

Similarly, the probability of observing the sequence $\boldsymbol{o}_1^T$ and taking transition $a_{i,j}$ at time t is given by

$$P_\lambda(\boldsymbol{o}_1^T, \theta_{t-1} = i, \theta_t = j) = \alpha_i(t-1) a_{i,j} b_j(\boldsymbol{o}_t) \beta_j(t)$$

Finally, the probability of $\boldsymbol{o}_1^T$ whilst generating the observation vector $\boldsymbol{o}_t$ using mixture component m at state j is given by

$$P_\lambda(\boldsymbol{o}_1^T, \theta_t = j, \psi_t = m) = \sum_{i=1}^{N} \alpha_i(t-1) a_{i,j} c_{j,m} b_{j,m}(\boldsymbol{o}_t) \beta_j(t)$$

2.3 Maximum Likelihood estimation

The most common HMM parameter estimation technique is Maximum Likelihood estimation (MLE). Its best quality is the existence of a re-estimation formula $f(.)$ such that if $\hat{\lambda} = f(\lambda)$ then $\mathcal{F}(\hat{\lambda}) \geq \mathcal{F}(\lambda)$ with equality only when $\hat{\lambda}$ is a local maximum of $\mathcal{F}(\hat{\lambda})$. The existence of this re-estimation formula is the main reason for the introduction of HMMs in speech recognition and it is largely responsible for their success and popularity.

2.3.1 Objective function

In Maximum Likelihood estimation a parameter vector $\hat{\lambda}$ is derived so that

$$P_{\hat{\lambda}}(w,a) = \max_{\lambda} P_{\lambda}(w,a)$$

where $P_{\lambda}(w,a)$ is the probability that the model with parameter set λ in the family of distributions will generate the sample (w,a). By factoring the left hand side in the above equation we obtain

$$P_{\lambda}(w,a) = P_{\lambda}(a|w)P_{\lambda}(w) \tag{2.8}$$

The above shows that the acoustic model parameters and the language model parameters can be estimated separately by choosing the language parameters to maximise $P_{\lambda}(w)$ and by choosing the acoustic model parameters to maximise $P_{\lambda}(a|w)$. This suggests that MLE tries to increase the *a posteriori* probability of the training data given the model corresponding to the data. The models from other classes do not participate in the parameter re-estimation. Consequently, it is not obvious how the MLE objective function relates to the objective of reducing the error rate. In the following chapter, the rationale behind MLE and its limitations will be discussed in greater detail.

2.3.2 The Baum-Welch algorithm

The mathematical foundations of the Baum-Welch (BW) algorithm for MLE were established by Baum in [12]. The paper presented an iterative method for monotonically increasing the value of an arbitrary homogeneous polynomial $\mathcal{P}(X)$ with non-negative coefficients of degree d in variables x_{ij}, $i = 1,\dots,p$, $j = 1,\dots,q_i$, defined over a stochastic domain, $\mathcal{D} : x_{ij} \geq 0$, $\sum_{j=1}^{q_i} x_{ij} = 1$, through a series of transformations performed on $\{x_{ij}\}$. The transformation is defined as

$$T(x_{ij}) = \frac{x_{ij}\frac{\partial}{\partial x_{ij}}\mathcal{P}(X)}{\sum_{j=1}^{q_i} x_{ij}\frac{\partial}{\partial x_{ij}}\mathcal{P}(X)} \tag{2.9}$$

and is often referred to as a *growth* transformation of $\mathcal{P}(X)$. In a preceding paper Baum and Eagon [11] described a special case of the re-estimation procedure for probabilistic functions of Markov chains with discrete observations. The proof that the re-estimates provide an increase in the value of the likelihood was rather intricate. Later, in [12] and [10] the method was generalised to functions of Markov chains with continuously distributed observations. The proof relied on the assumption that the output distributions of the Markov chain are strictly log-concave. More recently, Liporace [72] and Juang [58] presented an analysis which extended the algorithm to accommodate a larger class of distributions and mixture distributions.

For HMMs with discrete output distributions, transition and observation parameters are both updated according to expression 2.9. The following section outlines the derivations of the ML re-estimation formulae for the parameters of continuous mixture density HMMs.

2.3.3 Parameter re-estimation

For continuous mixture distributions, we can express the likelihood of an input utterance in terms of the individual mixture components:

$$\begin{aligned} P_\lambda(\boldsymbol{o}_1^T) &= \sum_{\theta\in\Theta} P_\lambda(\theta)P_\lambda(\boldsymbol{o}_1^T|\theta) \\ &= \sum_{\theta\in\Theta} \prod_{t=1}^{T+1} a_{\theta_{t-1},\theta_t} \prod_{t=1}^{T} \sum_{m=1}^{M} c_{\theta_t,m}\mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_{\theta_t,m}, \boldsymbol{W}_{\theta_t,m}) \end{aligned} \tag{2.10}$$

For a fixed state sequence θ we can define a possible component sequence ψ. Then equation 2.10 can be re-written as

$$\begin{aligned} P_\lambda(\boldsymbol{o}_1^T) &= \sum_{\theta\in\Theta} \sum_{\psi\in\Psi_\theta} P_\lambda(\theta)P_\lambda(\psi)P_\lambda(\boldsymbol{o}_1^T|\theta,\psi) \\ &= \sum_{\theta\in\Theta} \sum_{\psi\in\Psi_\theta} \prod_{t=1}^{T+1} a_{\theta_{t-1},\theta_t} \prod_{t=1}^{T} c_{\theta_t,\psi_t} \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_{\theta_t,\psi_t}, \boldsymbol{W}_{\theta_t,\psi_t}) \end{aligned} \tag{2.11}$$

where Ψ_θ is the set of all possible mixture component sequences given a fixed state sequence θ. In order to re-estimate the parameters of the HMM, an auxiliary function $\mathcal{Q}(\lambda,\hat{\lambda})$ is introduced

$$\mathcal{Q}(\lambda,\hat{\lambda}) = \sum_{\theta\in\Theta} \sum_{\psi\in\Psi_\theta} P_\lambda(\boldsymbol{o}_1^T,\theta,\psi) \log P_{\hat{\lambda}}(\boldsymbol{o}_1^T,\theta,\psi) \tag{2.12}$$

The usefulness of $\mathcal{Q}(\lambda,\hat{\lambda})$ comes from the fact that $\mathcal{Q}(\lambda,\hat{\lambda}) \geq \mathcal{Q}(\lambda,\lambda)$ implies that $P_{\hat{\lambda}}(\boldsymbol{o}_1^T) \geq P_\lambda(\boldsymbol{o}_1^T)$. A proof of this inequality can be found in [12]. The log factor in equation 2.12 can be rewritten as

$$\log P_{\hat{\lambda}}(\boldsymbol{o}_1^T,\theta,\psi) = \sum_{t=1}^{T+1} \log \hat{a}_{\theta_{t-1},\theta_t} + \sum_{t=1}^{T} \log \hat{c}_{\theta_t,\psi_t} + \sum_{t=1}^{T} \log \mathcal{N}(\boldsymbol{o}_t; \hat{\boldsymbol{\mu}}_{\theta_t,\psi_t}, \hat{\boldsymbol{W}}_{\theta_t,\psi_t}) \tag{2.13}$$

and combining 2.12 and 2.13 gives

$$\begin{aligned} \mathcal{Q}(\lambda,\hat{\lambda}) &= \sum_{\theta\in\Theta} \sum_{\psi\in\Psi_\theta} P_\lambda(\boldsymbol{o}_1^T,\theta,\psi) \sum_{t=1}^{T+1} \log \hat{a}_{\theta_{t-1},\theta_t} \\ &+ \sum_{\theta\in\Theta} \sum_{\psi\in\Psi_\theta} P_\lambda(\boldsymbol{o}_1^T,\theta,\psi) \sum_{t=1}^{T} \log \hat{c}_{\theta_t,\psi_t} \\ &+ \sum_{\theta\in\Theta} \sum_{\psi\in\Psi_\theta} P_\lambda(\boldsymbol{o}_1^T,\theta,\psi) \sum_{t=1}^{T} \log \mathcal{N}(\boldsymbol{o}_t; \hat{\boldsymbol{\mu}}_{\theta_t,\psi_t}, \hat{\boldsymbol{W}}_{\theta_t,\psi_t}) \end{aligned} \tag{2.14}$$

Using this separability, maximisation of the likelihood function can be accomplished by maximising each component independently.

2.3.3.1 Mixture component parameters

Taking the partial derivative of equation 2.14 with respect to $\hat{\boldsymbol{\mu}}_{j,m}$ gives

$$\frac{\partial}{\partial \hat{\boldsymbol{\mu}}_{j,m}} \mathcal{Q}(\lambda, \hat{\lambda}) = \sum_{t=1}^{T} P_\lambda(\boldsymbol{o}_1^T, \theta_t = j, \psi_t = m) \frac{\partial}{\partial \hat{\boldsymbol{\mu}}_{j,m}} (\log b_{j,m}(\boldsymbol{o}_t)) \tag{2.15}$$

where $b_{j,m}(o_t) = \mathcal{N}(\boldsymbol{o}_t; \hat{\boldsymbol{\mu}}_{j,m}, \hat{\boldsymbol{W}}_{j,m})$ is mixture component m at state j. Equating the above equation to zero and solving for $\hat{\boldsymbol{\mu}}_{j,m}$ (see equation B.14) yields

$$\begin{aligned} \hat{\boldsymbol{\mu}}_{j,m} &= \frac{\sum_{t=1}^{T} P_\lambda(\theta_t = j, \psi_t = m)\boldsymbol{o}_t}{\sum_{t=1}^{T} P_\lambda(\theta_t = j, \psi_t = m)} \\ &= \frac{\sum_{t=1}^{T} \sum_{i=1}^{N-1} \alpha_i(t-1) a_{i,j} c_{j,m} b_{j,m}(\boldsymbol{o}_t) \beta_j(t) \boldsymbol{o}_t}{\sum_{t=1}^{T} \sum_{i=1}^{N-1} \alpha_i(t-1) a_{i,j} c_{j,m} b_{j,m}(\boldsymbol{o}_t) \beta_j(t)} \end{aligned} \tag{2.16}$$

similarly for the covariance matrices

$$\frac{\partial}{\partial \hat{\boldsymbol{W}}_{j,m}} \mathcal{Q}(\lambda, \hat{\lambda}) = \sum_{t=1}^{T} P_\lambda(\boldsymbol{o}_1^T, \theta_t = j, \psi_t = m) \frac{\partial}{\partial \hat{\boldsymbol{W}}_{j,m}} (\log b_{j,m}(\boldsymbol{o}_t)) \tag{2.17}$$

Equating to zero and solving for $\hat{\boldsymbol{W}}_{j,m}$ (analogous to equation B.17) yields

$$\begin{aligned} \hat{\boldsymbol{W}}_{j,m} &= \frac{\sum_{t=1}^{T} P_\lambda(\theta_t = j, \psi_t = m)(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_{j,m})(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_{j,m})'}{\sum_{t=1}^{T} P_\lambda(\theta_t = j, \psi_t = m)} \\ &= \frac{\sum_{t=1}^{T} \sum_{i=1}^{N-1} \alpha_i(t-1) a_{i,j} c_{j,m} b_{j,m}(\boldsymbol{o}_t) \beta_j(t) (\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_{j,m})(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_{j,m})'}{\sum_{t=1}^{T} \sum_{i=1}^{N-1} \alpha_i(t-1) a_{i,j} c_{j,m} b_{j,m}(\boldsymbol{o}_t) \beta_j(t)} \end{aligned} \tag{2.18}$$

2.3.3.2 Transition probabilities and mixture weights

The transition probabilities $a_{i,j}$ and mixture weights $c_{j,m}$ are non-negative and share the *sum-to-one* constraint. For the transition probabilities, the corresponding term in expression 2.14 can be rewritten as

$$\begin{aligned} &\sum_{\theta \in \Theta} \sum_{\psi \in \Psi_\theta} P_\lambda(\boldsymbol{o}_1^T, \theta, \psi) \sum_{t=1}^{T+1} \log \hat{a}_{\theta_{t-1}, \theta_t} = \\ &\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{t=1}^{T} \sum_{\psi \in \Psi_\theta} P_\lambda(\boldsymbol{o}_1^T, \theta_{t-1} = i, \theta_t = j, \psi) \log \hat{a}_{i,j} \end{aligned} \tag{2.19}$$

Considering transitions out of state i only, the above expression takes the form

$$\sum_{i=1}^{N} z_j \log y_j \tag{2.20}$$

This function attains a global maximum subject to the *sum-to-one* constraints at

$$y_j = \frac{z_j}{\sum_{i=1}^{N} z_i} \qquad \text{for } j = 1, 2, \ldots, N \tag{2.21}$$

A proof of the above is available in [71]. Hence, the re-estimation formula for $\hat{a}_{i,j}$

$$\begin{aligned}\hat{a}_{i,j} &= \frac{\sum_{t=2}^{T} P_\lambda(\boldsymbol{o}_1^T, \theta_{t-1} = i, \theta_t = j)}{\sum_{t=1}^{T} P_\lambda(\boldsymbol{o}_1^T, \theta_t = i)} \\ &= \frac{\sum_{t=2}^{T} \alpha_i(t-1) a_{i,j} b_j(\boldsymbol{o}_t) \beta_j(t)}{\sum_{t=1}^{T} \alpha_i(t)\beta_i(t)}\end{aligned} \tag{2.22}$$

where $1 < i < N$ and $1 < j < N$. The transitions from the non-emitting entry state and the transitions from the emitting states to the final non-emitting state are treated as special cases and their re-estimation formulae can be found in [117]. Following similar derivations, the re-estimation formulae for the mixture component weights $\hat{c}_{j,m}$ is given by

$$\begin{aligned}\hat{c}_{j,m} &= \frac{\sum_{t=1}^{T} P_\lambda(\boldsymbol{o}_1^T, \theta_t = j, \psi_t = m)}{\sum_{t=1}^{T} P_\lambda(\boldsymbol{o}_1^T, \theta_t = i)} \\ &= \frac{\sum_{t=1}^{T} \sum_{i=1}^{N-1} \alpha_i(t-1) a_{i,j} c_{j,m} b_{j,m}(o_t) \beta_j(t)}{\sum_{t=1}^{T} \alpha_i(t)\beta_i(t)}\end{aligned} \tag{2.23}$$

The re-estimation formulae given above can be interpreted as an implementation of the Expectation Maximisation (EM) algorithm [31] in which the expectation step is the calculation of the auxiliary function and the maximisation step is the maximisation of $\mathcal{Q}(\lambda, \hat{\lambda})$ over λ which provides $\hat{\lambda}$.

Finally, the fact that the Baum-Welch algorithm is mathematically guaranteed not to degrade the likelihood function is very pleasing. Most importantly, there is plenty of experimental evidence which shows that good parameter estimates are usually obtained after a small number of iterations.

2.4 HMM Structure alterations

Throughout the experimental work presented in this thesis, the HMM structure parameters will be manipulated using the following two mechanisms.

2.4.1 Parameter tying

Sharing of parameters between multiple HMMs is a powerful technique for achieving better generalisations and coping with limited amounts of training data. The idea of sharing parameters is not new. Indeed, in continuous speech recognition the same phone model is used for each occurrence of the corresponding phoneme in the training utterances. Tying all covariance matrices across all models produces a *Grand Variance/Covariance* model set [89] and tying the means across all mixture components can be used to obtain Richter style distributions [96]. The above listed cases can be considered as special instances of a more general tying mechanism where each HMM parameter set can be shared. Such general tying framework was described by Young [112] with the aim of manipulating and modifying existing HMM sets. In general

- Tying aids robust parameter estimation when data is limited since the model structure can be adjusted to match the available amount of training data. Furthermore, this opens the way to an automatic data-driven approach for HMM construction.
- Careful implementations of such tying schemes will result in a dramatic reduction in storage requirements and caching of computation. This is particularly true for large speaker independent continuous speech recognition systems, where the use of complex context dependent acoustic models is mandatory.

More formally it can be shown that tying does not alter the form of the parameter re-estimation formulae and the convergence properties of the Baum-Welch algorithm. This follows from the expanded form of the auxiliary function (equation 2.14), where tying of any parameter can be regarded as partitioning the summations which does not alter the re-estimation formulae [13].

2.4.2 Mixture component incrementing

Mixture component incrementing provides an iterative mechanism for building a multiple mixture component system from a single Gaussian system. This is accomplished in stages, by incrementing all state distributions by one or two mixture components at each stage. An output distribution of M mixture components is converted to an $M+1$ component mixture distribution by cloning the mixture component with the largest weight and then perturbing the mean vectors of the two identical distributions by adding/subtracting 0.2 standard deviations respectively. The new system is then trained using a few iterations of the Baum-Welch algorithm.

Traditionally, mixture density HMMs are built by using the segmental k-means procedure to initialise the required number of mixture components and then retraining the models using the Baum-Welch algorithm. However, this approach requires one to decide on the number of mixture components prior to building and assessing the performance of the desired system. The former approach has been shown to produce similar results to the k-means clustering method [116]. At the same time, it has the advantage that the number of mixture components can be continuously increased to obtain any desired balance between performance and model complexity.

2.5 Whole-word and sub-word modelling

Hidden Markov models can be used to model speech at several linguistic levels e.g. phones, syllables, words etc. Lee [66] defines a good unit model as the one which satisfies the following two criteria:

- *consistency* - different examples of the same unit have similar acoustic realisations.
- *trainability* - there is a sufficient number of examples for each unit to guarantee robust estimation of model parameters.

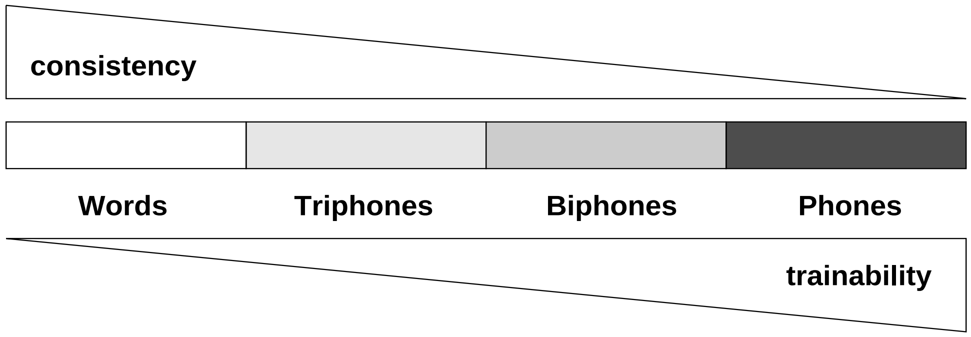

Figure 2.2: Typical levels of acoustic modelling

Consistent acoustic models will improve discrimination and overall recognition performance. Trainability will guarantee generalisation and better use of model parameters (see figure 2.2).

Words are the most natural units of speech and usually form the output of most speech recognition systems. Word models are consistent by providing an implicit modelling of within-word coarticulations and word-dependent context dependencies. They are the obvious choice for small vocabulary systems and keyword spotting applications. However, using word models in LVCSR introduces several problems. A large number of word models will require more training data with an adverse effect on the ability to add new words to the dictionary.

Sharing model parameters is the obvious solution to improving trainability. English has a relatively small number of phonemes (40) which can be used to construct every word in the language. Due to their small number, these can be easily trained from a relatively small corpora of data. However, phonemes are a very abstract linguistic units and their actual acoustic realisations are extremely variable. Contextual effects in current speech recognition systems are usually modelled by using context dependent phonetic models. The most common such unit is the triphone which is a particular instance of a phone occurring in a specific neighbouring context. Triphone models are powerful because they are much more specific and thus able to capture fine phoneme variations. However, triphone models suffer from severe undertraining problems. For a typical set of 40 monophones, there are 40^3 potential triphones with only a few of them actually occurring in the training script. The problem of reliable estimation can be overcome by applying smoothing techniques [66], and selective parameter tying based on HMM state clustering [112, 116]. Recent results in [114], have demonstrated the use of a phonetically-driven tree-based clustering method. Using this approach, acoustic models are assembled on-the-fly from a library of states using binary decision trees and context information.

In all phone recognition experiments described later in this thesis, we shall make exclusive use of context independent HMMs. This choice was made for the following reasons: 1) monophone HMM systems are far easier to train and manipulate; 2) as it will be seen later, discriminative training methods require significantly more computation than conventional

MLE training. Since the required computation is also proportional to the number of models, the use of context independent models was necessary to make the experimental work tractable.

2.6 Viterbi decoding and search strategies

The decoding of an unknown utterance proceeds according to the MAP rule discussed in section 2.1. This involves computing the likelihood of the unknown observation sequence given each acoustic model and choosing the one with the highest likelihood. In general it is possible to use the forward probability calculation to compute the overall likelihood $P_\lambda(\boldsymbol{o}_1^T)$, and to identify the utterance based on these quantities. However, in practice we are more interested in the most likely state sequence which generated the sequence $\boldsymbol{o}_1^T$. Furthermore, in many cases the decision of choosing w as in expression 2.1 is implicitly incorporated in the model by combining several models in parallel with common initial and final states and in such cases the maximum likelihood path is an essential outcome of the recognition. The Viterbi algorithm [107] is a general dynamic programming technique used to find the most likely path in a trellis of nodes. The likelihood of the path is computed according to

$$\gamma_j(t) = \max_i \{\gamma_i(t-1)a_{i,j}\}\, b_j(\boldsymbol{o}_t)$$

where for a given model $\gamma_j(t)$ is the maximum likelihood of observing $\boldsymbol{o}_1^t$ and being in state j at time t. Indeed the above expression is very similar to 2.7 with the summation replaced by a maximum.

Continuous speech recognition is normally performed as a time-synchronous Viterbi search in a state space. The search produces the most likely word sequence by matching each frame from the unknown utterance to a network of HMM instances. The network is compiled to reflect the grammar of the language and may explicitly incorporate language model probabilities. In this case, the search itself is the computationally most expensive part of the recognition system due to the huge number of possible paths one has to consider. This is a result of the vocabulary size and inherent acoustic ambiguities. In order to limit the search space it is customary to threshold the scores generated by the acoustic models. Multi-pass recognition systems are another way of making the recognition task more manageable. A typical example is a two-pass system where the first pass generates a list of the N most probable word sequences (N-best list) using simplified acoustic models [3, 103]. The second pass re-scores the list using detailed acoustic models and a language model. A fundamental problem with multi-pass systems is that search errors introduced in early passes are impossible to rectify thus resulting in degraded performance. With this in mind, Odell et al. [86, 106] have developed a single-pass decoding algorithm capable of using complex acoustic models and long-span language models. A version of this decoder will be used in the experimental chapters of this thesis to perform continuous phone recognition with a fourgram language model.

2.7 Language models

The language model is a natural component in the information-theoretic formulation of the speech recognition problem. A language model assigns a probability value to every word sequence w. In speech recognition this value is interpreted as the *a priori* probability that the speaker will utter the word sequence w. These probabilities will direct the search during recognition amongst the various partial hypotheses. Well designed language models result in improved recognition accuracy and reduced search complexity. The constraining power of a language model is usually measured by its *perplexity* which can be interpreted as the average number of words that have to be hypothesised at any decision point [82]. Simple language models are implemented as network representations of finite state grammars [88]. Such models are rather restrictive and their application is limited to small vocabulary tasks with a well defined language structure.

The introduction of *word-pair* grammars [66] marked the start of the transition towards more general language models for LVCSR. The most popular language models at present are the bigram and trigram language models. These models are estimated from a text corpus during the training phase.

Any type of language model faces the problem that the amount of training data is limited and always too sparse to observe the typical events often enough. The most common approach to dealing with unseen events is the general discounting back-off method proposed by Katz [62]. In this approach, counts of frequent events (n-grams) are discounted and the accumulated probability mass is redistributed amongst the unseen events according to a more general distribution, e.g. the $(n-1)$-gram. Two common discounting methods are: 1) Turing-Good discounting [62] and 2) Absolute discounting [82]. In the continuous phone recognition experiments presented in the thesis, the latter approach will be used to construct a phone-level fourgram back-off language model.

2.8 The HTK toolkit and extensions

HTK is a hidden Markov model toolkit originally designed by Young [113] at Cambridge University Engineering Department. The toolkit contains a set of tools and associated library modules for building and testing HMM based speech recognizers. The toolkit is primarily intended for building sub-word based continuous speech recognition systems, however, it can also be used for isolated or whole-word based systems. The toolkit was designed to support continuous density hidden Markov models with any number of states and mixture components. It also implements a general parameter tying mechanism which allows the creation of complex model topologies to suit a variety of speech recognition applications. The models are stored as text which allows for the easy creation, manipulation and inspection of model sets. The following lists the very basic set of tools provided by HTK for building and evaluating speech recognition systems

- `HInit` - this tool is used to compute the initial set of parameters for an HMM from labelled data using the k-means clustering algorithm followed by repeated Viterbi segmentation of the training utterances.

- `HRest` - implements Maximum Likelihood re-estimation of the parameters of individual HMMs using labelled training utterances.

- `HERest` - implements *embedded* Maximum Likelihood training of a family of HMMs. The tool uses only sequence information from the label file corresponding to the training utterance to derive a new set of parameters.

- `HVite` - this is a continuous speech Viterbi recogniser which matches unknown utterances against a set of HMMs using finite state grammar constraints.

- `HResults` - this tool performs Dynamic Programming (DP) alignment between the output of the recogniser and the true transcription to measure the accuracy of the recognition.

The experimental work described in this thesis also made extensive use of a generic implementation of the embedded Baum-Welch algorithm (`HNRest`), in which each training utterance can be aligned against a pre-specified network of HMMs. As it will be seen later, such an alignment is essential to carry out discriminative optimisation of HMM parameters. Any tools written to evaluate the methods discussed in this thesis were built as extensions of existing HTK tools and library modules.

Chapter 3

Discriminative Training

In HMM-based speech recognition, the purpose of training is to find the HMM parameter set which results in the lowest possible recognition error rate for the chosen decoder. The most common HMM parameter estimation technique is Maximum Likelihood estimation (MLE) which was briefly described in chapter 2. Its most obvious quality is the existence of a re-estimation algorithm which, in practice, requires very few iterations to obtain the desired results. One problem with MLE training is that it has no direct relationship with the aim of minimising the recognition error rate. Discriminative training techniques remove the need for the assumption that the underlying models are correct and attempt to improve recognition performance by adjusting the model parameters themselves.

In this chapter we examine the HMM parameter re-estimation problem from an information theoretic point of view. Two principle methods are then discussed in detail - MLE and Maximum Mutual Information estimation (MMIE). Their theoretical formulation is compared to other previous and current approaches to enhancing the discriminative abilities of the HMM framework. Finally, a modification to the MMIE objective function is discussed which can result in better generalisation under certain conditions.

3.1 Introduction

Current speech recognition technology is based on the *information-theoretic* formulation of the speech recognition problem. The following two sections provide a concise introduction to this theory.

3.1.1 Entropy and mutual information

Let X, Y be random variables with realisations x, y. The measure of uncertainty in X is the average number of bits [1] necessary to specify the outcome of X when using an optimal encoding scheme. This measure is given by

$$H(X) = -\sum_{x} P(x) \log P(x) \tag{3.1}$$

[1] The units are bits if the log in equation 3.1 is taken to base 2.

and is known as the entropy of X. A measure of the average amount of uncertainty about X, given knowledge of Y is the conditional entropy of X given Y defined as

$$H(X|Y) = -\sum_{x,y} P(x,y) \log P(x|y) = -E\left[\log P(x|y)\right] \tag{3.2}$$

The amount of information in Y as to the identity of X is given by

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= \sum_{x,y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)} \end{aligned} \tag{3.3}$$

Since $I(X;Y) = I(Y;X)$, $I(X;Y)$ is known as the *average mutual information* between X and Y.

In the information theoretic formulation of the speech recognition problem, a spoken sequence of words w is viewed as passing through a probabilistic acoustic channel which produces a sequence of acoustic observations a, which are in turn passed to the probabilistic decoder. Let W, A denote random variables with corresponding values $w \in \mathcal{V}$, $a \in \mathcal{A}$, where $\mathcal{V}$ is the set of possible word sequences and $\mathcal{A}$ is the inventory of acoustic realisations. Using the above notation the message w is encoded into a. The uncertainty about the identity of a word sequence w given a sequence of acoustic information a is the conditional entropy of W given A

$$H(W|A) = H(W) - I(W;A) \tag{3.4}$$

Hence, $H(W|A)$ can be interpreted as the shortest sequence of bits which are necessary to specify W, given knowledge of A. In practice $P(w,y)$ is not known and we have to consider a family of parametric distributions $P_\lambda(w,y)$ where λ is the parameter set of the distribution model. The conditional entropy of words given acoustics is

$$\begin{aligned} H_\lambda(W|A) &= -\sum_{w,a} P(w,a) \log P_\lambda(w|a) \\ &= -\sum_{w,a} P(w,a) \log \frac{P_\lambda(w|a)}{P(w|a)} - \sum_{w,a} P(w,a) \log P(w|a) \\ &= -\sum_{w,a} P(w,a) \log \frac{P_\lambda(w|a)}{P(w|a)} + H(W|A) \\ &\geq -\sum_{w,a} P(w,a) \left[\frac{P_\lambda(w|a)}{P(w|a)} - 1\right] + H(W|A) \\ &\geq H(W|A) \end{aligned} \tag{3.5}$$

The above inequality follows from the fact that $\log(x) \leq x - 1$, with equality for $x = 1$. In the above equation the equality only holds when $P_\lambda(w|a) = P(w|a)$. Consequently, in minimising $H_\lambda(W|A)$ we attempt to make $P_\lambda(w|a)$ as similar as possible to $P(w|a)$.

3.1.2 Probabilistic decoder

A speech decoder is a device which estimates the identity of w using information from its acoustic representation a. The decoder can be expressed as a mapping $\hat{w} = f(a)$ where $f(.)$

is the decoding function. If the chosen identity $\hat{w}$ is different from the actual identity of the spoken utterance then there is a decoding error. The probability of making an error is the most important factor in choosing the decoder. This quantity is defined as

$$\vartheta(f) = 1 - \sum_{a} P(f(a), a)$$

where the term on the right defines the probability of correct classification. The optimal decoder with regard to minimising the probability of error is the maximum *a posteriori* (MAP) decoder, where $\hat{w}$ is chosen such that

$$\hat{w} = \arg\max_{w} P(w|a) \tag{3.6}$$

In practice the true probability distributions are not known, hence the decoder will make the decision about the identity of the utterance by using the chosen parametric model. Hence, the model $P_\lambda(w|a)$ should be chosen so that $P_\lambda(w|a)$ is large when the true $P(w, a)$ is large. Inspection of equation 3.5 reveals that such a model tends to make $H_\lambda(W|A)$ small. Consequently, in reverse, minimising $H_\lambda(W|A)$ will tend to minimise the decoder's probability of error. By Bayes' rule, equation 3.6 can be reformulated as

$$\hat{w} = \arg\max_{w} P_\lambda(w|a) = \arg\max_{w} \frac{P_\lambda(a|w)P_\lambda(w)}{P(a)}$$

This shows that our model of $P(w|a)$ can be decomposed into two components, the acoustic model $P_\lambda(a|w)$, and the language model $P_\lambda(w)$.

3.1.3 Acoustic and language models

The acoustic model $P_\lambda(a|w)$ computes the probability that the speaker will generate the acoustic observation sequence a, knowing that he or she will utter the word sequence w. The language model $P_\lambda(w)$ provides the prior probability that the word sequence w will be spoken. Analogous to equation 3.4, the uncertainty $H(W|A)$ calculated using our parametric model is

$$H_\lambda(W|A) = H_\lambda(W) - I_\lambda(W; A)$$

where, analogous to equation 3.1

$$H_\lambda(w) = -\sum_{w} P(w) \log P_\lambda(w)$$

and, analogous to equation 3.3

$$I_\lambda(W; A) = \sum_{w,a} P(w, a) \log \frac{P_\lambda(w, a)}{P_\lambda(w)P_\lambda(a)}$$

It is most common to minimise $H_\lambda(W|A)$ by first finding a language model which minimises $H_\lambda(w)$. Once the language model has been determined, we can choose an acoustic model which maximises $I_\lambda(W; A)$ given $P_\lambda(w)$. The lower bound of $H_\lambda(w)$ is $H(w)$ and this value can be achieved only when $P(w)$ is known. In the rare circumstances where word sequences are generated according to an artificial grammar, the lower bound is reached simply by using the probabilities from the grammar. In large vocabulary systems designed to cope with natural speech this is not possible, and the language model parameters are estimated from large samples of text. The work presented in this thesis is primarily concerned with improving the quality of the acoustic model. However, the continuous phone recognition experiments presented in chapter 5 will make use of a bigram and a fourgram language models.

As in the language modelling task, when maximising $I_\lambda(W;A)$ the true probabilities are not known and one has to consider a family of parametric distributions. The parameters of the acoustic model are then estimated from labelled training utterances e.g. samples of (W, A). In maximising $I_\lambda(W;A)$, the goal of acoustic modelling is to extract as much information as possible from the acoustic signal about the identity of the corresponding word sequence. In chapter 2 we described an acoustic modelling approach based on hidden Markov models. The structure of these models is based on certain assumptions about speech and, although these assumptions are not strictly true, the HMM framework provides a tractable scheme for modelling the peculiarities of the acoustic phenomenon. An HMM has two principle types of parameters - transition probabilities and parametric output distributions. The time-varying nature of speech is modelled by the transition parameters. With a large number of states very fine phonetic detail can be modelled. The output distributions model the acoustic signal. Mixture Gaussian probability densities have the power to model any probability density function when estimated from sufficient training data. Finally, the structure of the model can be finely tuned to achieve good generalisation even when a limited amount of training data is available.

In the remaining sections of this chapter we shall examine different ways of estimating the parameter vector λ. Without any loss of generality, the parameter estimation techniques will refer to the portion of λ corresponding to the acoustic model, although, in certain cases, the language model probabilities will be used in the parameter estimation procedure. Most speech recognition systems utilise HMMs to model the acoustic signal at subword or word level. In this case, the probability $P_\lambda(a|w)$ is computed using the HMM corresponding to w. In order to make the re-estimation explicit, we shall assume that the training data consists of R independent samples (a_r, w_r) for $r = 1, \ldots, R$, and the HMM corresponding to word w_r will be denoted by $\mathcal{M}_{w_r}$[2]. Two principal methods which relate to the entropy criterion (section 3.1.1) will be discussed in more detail: *Maximum Likelihood estimation* (MLE) and *Maximum Mutual Information estimation* (MMIE).

3.2 Maximum Likelihood estimation (MLE)

In the statistical framework, speech recognition can be performed optimally if the probability of any word in the recogniser's vocabulary (*the language model*) and the probability distribution of the acoustic signal representation (*the acoustic model*) corresponding to that

[2] In the case of sub-word models, $\mathcal{M}_{w_r}$ is the compound model constructed by concatenating several sub-word HMMs according to the pronunciation of w_r.

word are known. In this case, the MAP decoder (section 3.1.2), which chooses from all possible words in the vocabulary the word that gives the highest conditional probability given the acoustic signal, produces the optimal speech recognition device in the sense of minimising the probability of error. An estimator $g(w, a)$ is a function of samples of random variables W and A which yields a new parameter vector $\hat{\lambda}$. In standard ML estimation the parameter vector $\hat{\lambda}$ is derived such that

$$\hat{\lambda}_{ML} = g_{ML}(w, a) = \arg\max_{\lambda} P_{\lambda}(w, a) = \arg\max_{\lambda} \{P_{\lambda}(a|w)P_{\lambda}(w)\}$$

Explicitly accounting for each individual training utterance, the above can be reformulated as

$$\hat{\lambda}_{ML} = \arg\max_{\lambda} \prod_{r=1}^{R} \{P_{\lambda}(a_r|\mathcal{M}_{w_r})P_{\lambda}(w_r)\}$$

In other words, we choose the parameter set $\hat{\lambda}$ which maximises the joint probability of words w and acoustics a. Assuming model correctness, the performance of the recogniser will get closer to the optimal performance as the estimator $g(w, a)$ gets closer to the true parameter set $\tilde{\lambda}$. The estimator $g(w, a)$ is a function of samples of random variables. Hence, the estimator itself is a random variable with a distribution related to the distributions of the sample components. It can be shown that if 1) the sample (w, a) is a sample from the assumed family of distributions, 2) the family of distributions is well behaved, and 3) the sample (w, a) is large enough, then, the maximum likelihood estimator $g_{ML}(w, a)$ has a Gaussian distribution with mean $\tilde{\lambda}$ and a variance based on $1/nB^2_{(w,a)}$, where n is the size of the sample and $B_{(w,a)}$ is the *Fisher Information* [78]. Furthermore, it follows from the law of large numbers that the maximum likelihood estimator is consistent e.g. $\lim_{n\to\infty} g_{ML}(w, a) = \tilde{\lambda}$. Nádas [78] has shown that when the three previously mentioned conditions are true no other consistent estimator has lower variance. Hence, no other estimator can provide a closer estimate to the true parameter set $\tilde{\lambda}$ than the maximum likelihood estimator. Finally, if the performance of the system does not get worse as the estimated parameter set moves closer to the true parameter set, then a system trained using MLE will perform as well as any other system trained using a different form of estimation.

The argument in favour of MLE presented in the above relies on the assumption that the family of parametric models used to compute $P(a|w)$ contains the true distribution of the source. This is simply not true, because speech production is a complex mechanical process whose accurate modelling, even if possible, will require a large number of parameters. Almost certainly, such a model will be under-restrictive, thus requiring a vast amount of data in order to obtain reliable estimates of its parameters. A similar argument applies to the distribution $P(w)$. The true language model is only available if the spoken utterances conform to an artificial grammar. Due to its complexity and variation, the true grammar of the English language cannot be expressed in an algorithmic form, hence, we can only estimate $P(w)$ using a parametric model trained from samples of text. Finally, training data is always limited. Our acoustic models were specifically chosen to be HMMs since the HMM allows its structure to be altered and adapted to the available amount of training data.

From the above, it follows that the optimality of MLE is based on assumptions which are not valid in practice. However, it is still possible to minimise the uncertainty of finding the correct identity of a spoken utterance given its acoustic representation by adjusting the model parameters so as to minimise this quantity directly.

3.3 Maximum Mutual Information estimation (MMIE)

A few years ago, Maximum Mutual Information estimation (MMIE) was proposed by Bahl et al. [6] as an alternative to MLE. MMIE attempts to find the HMM parameter set which maximises the mutual information between the models and the training data. This is equivalent to maximising the *a posteriori* probability that each utterance in the training data was generated by the corresponding model. MMIE is very similar to the Conditional Maximum Likelihood estimation (CMLE) procedure discussed in [78, 80]. In many cases, optimising HMM parameters using MMIE and CMLE will produce identical estimates, however, the subtle differences between MMIE and CMLE are worth pointing out. The Conditional Maximum Likelihood estimator $g_{CML}(w,a)$ is defined as

$$\hat{\lambda}_{CML} = g_{CML}(w,a) = \arg\max_{\lambda} E\left[\log P_\lambda(w|a)\right]$$

However, analogous to equation 3.2, we have

$$H_\lambda(W|A) = -\sum_{w,a} P(w,a)\log P_\lambda(w|a) = -E\left[\log P_\lambda(w|a)\right] \tag{3.7}$$

Hence, $g_{CML}(w,a)$ will choose a parameter set $\hat{\lambda}$ which minimises $H_\lambda(W|A)$. Since the true distribution $P(w,a)$ is unknown, the value of $E\left[\log P_\lambda(w|a)\right]$ can be estimated by assuming that the sample (w,a) is representative and replacing the expectation in equation 3.7 by the sample average. The CML objective function is thus given by

$$f_{CML}(\lambda) = E\left[\log P_\lambda(w|a)\right] = \frac{1}{R}\sum_{r=1}^{R}\log P_\lambda(w_r|a_r) \tag{3.8}$$

We can use Bayes' rule to reformulate expression 3.8 in terms of our parametric models

$$\begin{aligned} f_{CML}(\lambda) &= \frac{1}{R}\sum_{r=1}^{R}\log\frac{P_\lambda(a_r|\mathcal{M}_{w_r})P_\lambda(w_r)}{P(a_r)} \qquad (3.9)\\ &= \frac{1}{R}\sum_{r=1}^{R}\left\{\log P_\lambda(a_r|\mathcal{M}_{w_r})P_\lambda(w_r) - \log P(a_r)\right\} \end{aligned}$$

Next, we consider the Maximum Mutual Information (MMI) estimator as discussed in [6, 22]. This estimator is defined as

$$\hat{\lambda}_{MMI} = g_{MMI}(w,a) = \arg\max_{\lambda} I_\lambda(W;A)$$

Assuming that the language model $P_\lambda(w)$ is given, the MMI objective function is defined as

$$f_{MMI}(\lambda) = I_\lambda(W;A) = H_\lambda(W) - E\left[\log P_\lambda(w|a)\right] \tag{3.10}$$

and using sample averages instead of expectations

$$
\begin{aligned}
f_{MMI}(\lambda) &= -\frac{1}{R}\sum_{r=1}^{R}\log P_\lambda(w_r) + \frac{1}{R}\sum_{r=1}^{R}\log\frac{P_\lambda(a_r|\mathcal{M}_{w_r})P_\lambda(w_r)}{P(a_r)} \\
&= \frac{1}{R}\sum_{r=1}^{R}\Big\{\log P_\lambda(a_r|\mathcal{M}_{w_r}) - \log P(a_r)\Big\} \qquad (3.11)
\end{aligned}
$$

The first/second term on the right in equation 3.11 will be referred to as the "numerator"/"denominator" likelihoods respectively of the MMIE objective function. Let us now compare the CML objective function (equation 3.8) and the MMI objective function (equation 3.10). If the re-estimation procedure does not change the parameters of the language model $P_\lambda(w)$ then the first term on the right in equation 3.10 is a constant, hence, maximising $I_\lambda(W;A)$ with respect to λ is equivalent to minimising $H_\lambda(W|A)$. In virtually all MMI-related experimental work published in the literature, MMI estimation is in fact equivalent to CML estimation. The estimation of the language model parameters is carried out separately, prior to estimating the parameters of the acoustic model. Minimisation of $H_\lambda(W)$ is achieved by choosing a language model which yields the lowest perplexity on the training data. However, in continuous speech recognition tasks using the language model whilst computing $P_\lambda(a_r|\mathcal{M}_{w_r})$ may allow for a more efficient pruning. Unless explicitly required by the context, for the remaining parts of this thesis the term MMIE will be loosely used to refer to both MMIE and CMLE.

The MMI objective function involves two components. Choosing λ to maximise the first term in the bracketed expression in equation 3.11 is equivalent to finding the ML estimate of λ. The second term constitutes the difference between MMIE and MLE. The second term can be expanded in terms of the acoustic model and the language model as follows

$$P_\lambda(a_r) = \sum_{\hat{w}} P_\lambda(a_r|\mathcal{M}_{\hat{w}})P_\lambda(\hat{w}) \qquad (3.12)$$

In an isolated speech recognition task the above expression is relatively easy to compute since the summation is taken over all words in the vocabulary. In a continuous speech recognition task, the summation involves all possible word sequences. Even for small vocabulary tasks ($\ll$ 100 words) this expression can be prohibitively expensive to compute and store. In general, $P_\lambda(a_r)$ can be computed as the likelihood $P_\lambda(a_r|\mathcal{M}_{rec})$ where $\mathcal{M}_{rec}$ is a composite HMM built from the individual models $\mathcal{M}_{w_r}$ according to the grammar of the task. In many cases, $\mathcal{M}_{rec}$ is equivalent to the model used in the Viterbi recognition. Using the recognition model to compute the denominator of the MMI objective function has the effect of propagating the grammar constraints into the training process. In general, the more restrictive the grammar is the less computation will be required to evaluate $P_\lambda(a_r)$. However, incorporating higher level grammar constraints will require more sophisticated search techniques. An example of $\mathcal{M}_{rec}$ is the looped phonetic model used in continuous phone recognition where bigram language constraints are enforced on the loop transitions [76]. In the cases where it is not possible to design a suitable $\mathcal{M}_{rec}$ of a reasonable size, then expression 3.12 is usually approximated [6, 26]. Since $P_\lambda(a_r|\mathcal{M}_{rec})$ is computed as the sum of all possible paths through the model, an obvious approximation is to take the N largest terms in the summation which can be determined using the N-best algorithm [103].

The effect of $P_\lambda(a_r)$ is made obvious by computing the derivative of the MMI objective function with respect to the parameter set λ.

$$\frac{\partial}{\partial\lambda} f_{MMI}(\lambda) = \frac{1}{R}\sum_{r=1}^{R}\left\{\frac{1}{P_\lambda(a_r|\mathcal{M}_{w_r})}\frac{\partial}{\partial\lambda}P_\lambda(a_r|\mathcal{M}_{w_r}) - \frac{1}{P_\lambda(a_r|\mathcal{M}_{rec})}\frac{\partial}{\partial\lambda}P_\lambda(a_r|\mathcal{M}_{rec})\right\} \tag{3.13}$$

Assuming that the recognition model $\mathcal{M}_{rec}$ includes $\mathcal{M}_{w_r}$ for $r = 1, \ldots, R$ then for a given r, $P_\lambda(a_r|\mathcal{M}_{rec})$ can be decomposed into

$$P_\lambda(a_r|\mathcal{M}_{rec}) = P_\lambda(a_r|\mathcal{M}_{rec-r}) + P_\lambda(a_r|\mathcal{M}_{w_r})P_\lambda(w_r) \tag{3.14}$$

and combining equations 3.13 and 3.14 gives

$$\begin{aligned}\frac{\partial}{\partial\lambda} f_{MMI}(\lambda) &= \frac{1}{R}\sum_{r=1}^{R}\frac{\partial}{\partial\lambda}P_\lambda(a_r|\mathcal{M}_{w_r})\left\{\frac{1}{P_\lambda(a_r|\mathcal{M}_{w_r})} - \frac{P_\lambda(w_r)}{P_\lambda(a_r|\mathcal{M}_{rec})}\right\} \\ &- \frac{1}{R}\sum_{r=1}^{R}\frac{1}{P_\lambda(a_r|\mathcal{M}_{rec})}\frac{\partial}{\partial\lambda}P_\lambda(a_r|\mathcal{M}_{rec-r}) \end{aligned} \tag{3.15}$$

The first term in the MMIE derivative is in the same direction and proportional to the MLE derivative. The effect of the second term is to subtract a component in the direction of each incorrect path constituted when computing the likelihood of a_r given $\mathcal{M}_{rec-r}$. It is important to realise that $\mathcal{M}_{rec-r}$ implicitly incorporates the language model, hence, paths corresponding to more probable word sequences will make a greater contribution to the MMIE derivative. This is intuitively correct, since we would like to correct errors in frequently occurring word sequences. Equation 3.15 also shows the fundamental difference between MMIE and MLE. In MLE, the model parameters are adjusted to maximise the probability of generating the data given the corresponding transcriptions. In MMIE, the parameter set λ is chosen to improve discrimination between each correct word sequence and every possible word sequence. This is also equivalent to maximising the *a posteriori* probability that the training data was generated by the corresponding model. If the likelihood $P_\lambda(a_r|\mathcal{M}_{rec})$ is represented by a single path in $\mathcal{M}_{rec}$ and this path corresponds to the correct transcription of a_r, the contribution of a_r to the MMIE derivative will vanish. This corresponds to perfect recognition of a_r.

Nádas [78] has carried out a theoretical comparison between MLE and CMLE. He has shown that if the true distributions are known, the CML estimator is consistent and the asymptotic variance of the ML estimator cannot exceed the asymptotic variance of the CML estimator. However with a limited amount of training data, the CML estimator will have greater variance than the corresponding ML estimator.

3.4 Previous MMIE results

There have been a number of comparisons between MLE and MMIE over the past few years. The IBM speech recognition group was the first to report results with MMIE. In their case, MMIE reduced the error rate by 18% on a 2000 word speaker dependent isolated word recognition system. Shortly after, Brown [22] reported improvements from MMIE for isolated recognition on the E-set task. However, in the case of discrete output distributions MMIE actually degraded the performance which was explained by the fact that discrete distributions do not make any assumptions about the shape of the "true" distributions.

Merialdo [76] successfully applied MMIE to speaker-dependent continuous phoneme recognition using discrete HMMs. He used a modified gradient descent training algorithm to update the HMM parameters, where the effect of "unreliable" low probability values was reduced by biasing the derivative expressions. The denominator of the MMI objective function was computed using the recognition looped phonetic model.

Perhaps the most significant evidence in favour of MMIE are the results published by Normandin [84, 23] on a continuous digit recognition task where in one case he achieved a reduction in error rate of 40%. Training was carried out using a modified version of the IBM-proposed extension of the Baum-Welch algorithm to rational[3] objective functions [46]. In the context of connected digit recognition, Normandin [84] proposed a modification to the MMIE algorithm, named, the "corrective" MMI training algorithm. This algorithm starts with MLE trained HMMs. Each iteration is a two step process. First recognition is performed on the training data to remove all correctly recognised utterances. The MMIE algorithm is then applied to the remaining part of the training data. The aim is to correct as many errors as possible from the training set in the hope that this will improve results on the test set. In the ideal case, for the correctly recognised utterances the numerator and denominator of the contribution to the MMI derivative will be very similar and consequently cancel out. However, in the case when the utterance is correctly recognised only because its path is marginally better than another path, "corrective" MMI training may result in oscillations. In practice, the results obtained by Normandin using "corrective" MMI training were very similar to the results obtained when using the full training set. The algorithm is computationally more efficient since computing the Viterbi path during recognition requires less computation than for the full likelihood used in the denominator of the MMIE objective function. Unfortunately, the algorithm cannot be applied to continuous phone recognition, since the phone models have small average duration, and accurate sentence recognition is very rare.

3.5 Other discriminative methods

3.5.1 Minimum Discrimination Information (MDI)

Another training paradigm, *minimum discrimination information* (MDI) training was proposed by researchers at AT&T [34, 35]. The discrimination information is a measure of closeness between two probability distributions under a given set of constraints. The MDI approach was proposed and theoretically studied in the context of autoregressive HMMs. In this context, the observed acoustic signal is associated with a sequence of partial covariance

[3] The objective function in MMIE is a rational objective function.

matrices which characterise the source distribution. The MDI estimator tries to choose a parameter set λ such that the discrimination information measure between the distribution attributed to the source and the model distribution is minimised. Since the MDI approach is based on a measure of discrimination, the parameter estimation procedure is not entirely influenced by the chosen model form. The expected performance of the MDI approach is not yet known since the method has not been fully implemented and studied. Both MMI and ML modelling approaches can be reformulated as MDI modelling approaches [36, 37]. In [37], the ML, MMI and MDI modelling approaches were also shown to be optimal in a minimal average discrimination information sense when used for simultaneous estimation of all acoustic models.

3.5.2 The H-criteria

The study in [47] presents a general family of estimators jointly referred to as the H-criteria. The various criteria are constructed as a weighted linear combination of the entropies of the joint and marginal distributions of words W and acoustics A. An H estimator $\hat{\lambda} = g_{(h_1,h_2,h_3)}(w, a)$ is obtained by minimising the corresponding H-criterion defined by

$$H_\lambda^{(h_1,h_2,h_3)} \triangleq h_1 H_\lambda(w, a) + h_2 H_\lambda(w) + h_3 H_\lambda(a)$$

Thus, MLE is a minimum cross-entropy estimate given by $g_{(1,0,0)}(w, a)$, CMLE is a minimum cross conditional entropy estimate obtained by $g_{(1,0,-1)}(w, a)$ and MMIE is given by $g_{(1,-1,-1)}(w, a)$. The problem of selecting the optimal decoder is redefined as the problem of choosing the appropriate H estimator. The authors describe an example where MLE, CMLE, and MMIE all select the wrong decoder, however, a different H-criterion succeeds in finding the correct decoder. In practice, the three parameters in $H_\lambda^{(h_1,h_2,h_3)}$ offer only two degrees of freedom. Furthermore, the language model is fixed which leaves one to consider the criteria given by $H_\lambda^{(1,-1,h_3)}$. Re-visiting equation 3.11 reveals that h_3 appears as a weighting factor applied to the denominator of the MMI objective function. In a set of preliminary experiments, the H-criteria was applied to a single speaker version of the IBM $20,000$ word recogniser. The H criteria were obtained by setting $h_3 = 0.0 \ldots 1.0$ in steps of 0.1. A gradient hill-climbing algorithm was run for 50 iterations in each case. An improvement in decoding accuracy of 4.8% was noticed after training the original MLE-derived models.

3.5.3 Minimum Classification Error (MCE)

An alternative to the conventional distribution estimation approach to decoder design is to formulate it as an optimisation problem in which the objective is to achieve minimum classification error on the training set. It should be understood, however, that if the amount of training data is insufficient, minimising the empirical classification error for the training set does not guarantee minimum error rate on the test data. Although the application of minimum classification error estimation to speech recognition is relatively new, several different implementations have emerged in recent publications [74, 43]. In the former reference, Ljolje et al. proposed a training scheme where the empirical error rate of the recogniser is estimated by the following function

$$\vartheta(\lambda) = 1 - \frac{1}{R} \sum_{\tilde{w} \in \mathcal{V}} \sum_{r: w_r = \tilde{w}} 1_{\omega_\lambda(\tilde{w})}(a_r) \tag{3.16}$$

where $1_{\omega_\lambda(\tilde{w})}(a_r)$ is a decision function which returns 1 if a_r is recognised as $\tilde{w}$ and 0 otherwise. This function constitutes the average number of utterances from the training data which were misclassified by the MAP decoder for a given set of acoustic and word models. The decision function is not differentiable, hence, $\vartheta(\lambda)$ cannot be optimised directly. A possible differentiable approximation is given by

$$1_{\omega_\lambda(\tilde{w})}(a_r) \approx \frac{P_\lambda(a_r|\mathcal{M}_{w_r})P_\lambda(w_r)}{\sum_{\hat{w}} P_\lambda(a_r|\mathcal{M}_{\hat{w}})P_\lambda(\hat{w})} \tag{3.17}$$

Comparing the above and CMLE objective function (equation 3.8) reveals that the two approaches use identical statistics in a rather different way. In CMLE, the sum of the logarithm of these statistics is maximised with respect to λ while, the pure sum of the same components is maximised in the MEE approach. One can argue that the latter approach is more appropriate since the sum of the logarithms used in the CMLE approach causes the CMLE derivative to be dominated by the smallest term in the summation e.g. the least favourable utterance. Consequently, the search in CMLE will concentrate on utterances far from the decision boundary, while the optimisation in MEE will receive an equal contribution from all utterance. In practice, CMLE will be easier to implement since the logarithm in the sum provides a natural normalisation when computing the derivatives of the HMM parameters.

More recently the Minimum Classification Error (MCE) training for HMMs was proposed in [25]. This approach is based on the definition of a general loss function, introduced within the framework of an adaptive discriminative learning paradigm, known as Generalised Probabilistic Descent (GPD) [59]. Several researchers have evaluated the MCE/GPD approach on a variety of isolated and continuous word recognition tasks, [94, 38]. In many cases, the name GPD has been used loosely to cover both the definition of the objective function and the actual learning algorithm used to optimise the function. However, in many of the examples, the convergence proofs provided by GPD are not needed since all of the training data is available. The objective function in MCE is defined by

$$f_{MCE}(\lambda) = \frac{1}{R} \sum_{r=1}^{R} (l_\lambda(d_\lambda(a_r)))$$

where $l_\lambda(.)$ is the individual utterance loss. The utterance loss, in turn, is defined in terms of the mis-classification measure $d_\lambda(a_r)$

$$l_\lambda(d_\lambda(a_r)) = \frac{1}{1 + e^{-d_\lambda(a_r)}}$$

The mis-classification measure $d_\lambda(a_r)$ can be interpreted as a measure of the penalty incurred when correctly classifying a_r, i.e.

$$d_\lambda(a_r) = -g_\lambda(a_r, \mathcal{M}_{w_r}) + \log\left(\left[\frac{1}{K-1}\sum_{\tilde{w}} e^{\eta_1 g_\lambda(a_r, \mathcal{M}_{\tilde{w}})}\right]^{1/\eta_1}\right) \tag{3.18}$$

where $g_\lambda(a_r, \mathcal{M}_{w_r})$ is the "discriminant function" and K is the number of classes.

$$g_\lambda(a_r, \mathcal{M}) = \log\left(\left[\sum_{\theta\in\Theta}\{P_\lambda(a_r|\mathcal{M}, s)\}^{\eta_2}\right]^{1/\eta_2}\right) \tag{3.19}$$

In the above, the summation is taken over all possible state sequences in the model $\mathcal{M}$. An inspection of the MCE objective function reveals that the method is based on statistics similar to the ones used in the MMI objective function. In virtually all experimental work, the discriminant function used in MCE (equation 3.19) is evaluated by setting $\eta_2 \to \infty$ which corresponds to evaluating the likelihood of the best path, commonly referred to as the Viterbi likelihood. Furthermore, the mis-classification measure (equation 3.18) is often calculated by setting $\eta_1 \to \infty$ so that the numerator of the logarithm corresponds to the Viterbi likelihood of the utterance computed over the best incorrect model and the denominator of the logarithm is the Viterbi likelihood of the utterance given the correct model. Training only takes place for utterances with non-negative mis-classification measures e.g. incorrectly recognised training utterances. We can point out two potential problems with this approach

1. The Viterbi likelihood can be a poor approximation to the overall likelihood computed over all possible paths in the model. Hence, discriminating against a single confusable path may not affect the overall recognition result. This is particularly true for utterances of long duration and in the case when crude acoustic models are used.

2. Choosing the most confusable model to discriminate against may result in oscillations during training. Furthermore, correct classifications close to the decision boundary are not considered, hence, a further source of oscillations. In general, the algorithm in its most common implementation, is likely to take longer to arrive at the optimal parameter set since confusable utterances are dealt with one at a time.

In the spirit of MCE, section 3.6 describes a modification to the MMIE objective function, where a non-linear weighting function is used to concentrate the parameter estimation procedure on utterances close to the decision boundary.

3.5.4 Corrective training schemes

Bahl et al. [7] introduced the corrective training algorithm for HMMs as an alternative to the Baum-Welch algorithm. Whereas the BW algorithm attempts to increase the probability that the models generated the training data, corrective training attempts to maximise the recognition rate on the training data. Corrective Training was designed in analogy with an error-correction training procedure for linear classifiers. While the latter can be shown to converge, it has not been possible to prove convergence for corrective training. Leaving aside questions of convergence, corrective training is appealing from a pragmatic point of view. The models are not assumed to be correct and for any set of models, corrective training attempts to find statistics which make the models work. The algorithm has two components: (1) *error-correction learning* improves correct words and suppresses mis-recognised words, and (2) *reinforcement learning* improves correct words and suppresses near-misses. When applied to the IBM speaker-dependent, isolated word office correspondence task, this algorithm reduced the error rate by 16%. A possible extension of Corrective Training to continuous speech is more problematic. With isolated-word input, both error-correcting and reinforcement training are relatively straightforward, since all errors are simple substitutions. However, in continuous speech recognition, the errors can be insertions, deletions and substitutions. Lee et al. [70] proposed such an extension of corrective training to continuous speech recognition. The proposed algorithm hypothesises near-miss sentences for any given sentence. First, a dynamic programming algorithm produces an ordered list of likely phrase substitutions. Then, this list is used to hypothesise the near-miss sentences used in the reinforcement learning stage. The modified training procedure was applied to the 997-word DARPA continuous resource management task, using the speaker-independent database. An error rate reduction of more than 20% over the standard MLE-trained SPHINX System [68, 69] was reported.

3.6 Non-linear MMIE

In this section we shall propose a modification to the MMIE objective function. The non-linear MMIE described here is a derivative of the standard MMIE criterion. It is important to understand that the MMIE objective function does not rely on the explicit definition of classes. For example, it can be applied to both isolated and continuous training utterances without any necessary alterations.

In the non-linear MMIE, we introduce an utterance specific weighting criterion based on the mutual information calculated for that utterance. The derivative of the weighting function is then used to determine the utterance's contribution to the derivative of the objective function. The weighting function is designed such that during training emphasis is placed on confusable utterances close to the decision boundary. Little weight is given to tokens which are easily recognised or those which are clearly recognised incorrectly. There are two reasons why the modified objective function may provide improvement in performance

1. Utterances with low mutual information may not constitute a representative sample for the corresponding class. Hence, adjusting the model parameters in order to accommodate such data may have an adverse effect on generalisation.

2. A discriminative training procedure is entirely dependent on the accurate labelling of the data. In that respect, MMIE is a lot more sensitive to mislabelled data than

the corresponding MLE procedure. An utterance with low mutual information can suggest that the data actually corresponds to a different class or simply mean that the sample is naturally indistinguishable from another class even for a human listener.

Let us now consider the MMIE objective function for a single training utterance a_r with corresponding transcription w_r. In the analysis below, we shall assume that the model $\mathcal{M}_{w_r}$ corresponding to w_r does not incorporate the language model scores and the generic model $\mathcal{M}_{rec}$ has the language model probabilities as between-unit transitions.

$$f_{MMI}(\lambda) = \log \frac{P_\lambda(a_r|\mathcal{M}_{w_r})P_\lambda(w_r)}{P_\lambda(a_r|\mathcal{M}_{rec})} \tag{3.20}$$

We can use the following decomposition

$$P_\lambda(a_r|\mathcal{M}_{rec}) = P_\lambda(a_r|\mathcal{M}_{rec-r}) + P_\lambda(a_r|\mathcal{M}_{w_r})P_\lambda(w_r)$$

And substituting the above in equation 3.20 yields

$$\begin{aligned} f_{MMI}(\lambda) &= -\log \frac{P_\lambda(a_r|\mathcal{M}_{rec})}{P_\lambda(a_r|\mathcal{M}_{w_r})P_\lambda(w_r)} \\ &= -\log \frac{P_\lambda(a_r|\mathcal{M}_{rec-r}) + P_\lambda(a_r|\mathcal{M}_{w_r})P_\lambda(w_r)}{P_\lambda(a_r|\mathcal{M}_{w_r})P_\lambda(w_r)} \\ &= -\log \left(1 + \frac{P_\lambda(a_r|\mathcal{M}_{rec-r})}{P_\lambda(a_r|\mathcal{M}_{w_r})P_\lambda(w_r)}\right) = -\log\left(1 + \Upsilon(a_r)\right) \end{aligned}$$

The quantity $\Upsilon(a_r)$ will be referred to as the "mis-classification measure" and it provides the following useful information. If $0 \leq \Upsilon(a_r) < 1$ then one can conclude that the utterance a_r will be correctly recognised. When $\Upsilon(a_r) = 1$ it means that there are one or more competing paths whose total probability is equal to the probability of the correct path. However, we cannot predict the outcome of the recognition since the exact number of these paths and their respective likelihoods are not explicitly known.

Hochberg et al. [49] have shown that the MMIE criterion applies different relative weighting to the training utterances as a function of where they lie in the decision space. To illustrate this we consider two word classes $\mathcal{W}_1$ and $\mathcal{W}_2$ with models $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively. The recognition model $\mathcal{M}_{rec}$ is constructed by placing $\mathcal{M}_1$ and $\mathcal{M}_2$ in parallel and a uniform language model is assumed e.g. $P_\lambda(w_r) = \frac{1}{2}$ for $w_r \in \mathcal{W}_1, \mathcal{W}_2$. The mutual information criterion of the training set is given by

$$f_{MMI}(\lambda) = -\frac{1}{R}\sum_{a_r:w_r\in\mathcal{W}_2} \log(1+\Upsilon(a_r)) - \frac{1}{R}\sum_{a_r:w_r\in\mathcal{W}_1} \log(1+\Upsilon(a_r))$$

Assuming that the MMIE objective function is maximised using a gradient search on the parameter set of the models, we can express the derivatives as follows

$$\frac{\partial}{\partial\lambda} f_{MMI}(\lambda) = -\frac{1}{R}\sum_{a_r:w_r\in\mathcal{W}_1} \frac{\Upsilon(a_r)}{1+\Upsilon(a_r)}\frac{\partial}{\partial\lambda}\log\Upsilon(a_r) - \frac{1}{R}\sum_{a_r:w_r\in\mathcal{W}_2} \frac{\Upsilon(a_r)}{1+\Upsilon(a_r)}\frac{\partial}{\partial\lambda}\log\Upsilon(a_r)$$

Consider a training token $a_r : w_r \in \mathcal{W}_1$. If the utterance is recognised correctly and it lies far from the decision boundary then it will have little weight in the derivative

$$\Upsilon(a_r) \ll 1 \;\;\Rightarrow\;\; h(a_r) \equiv \frac{\Upsilon(a_r)}{1+\Upsilon(a_r)} \ll 1$$

If a_r is near the decision boundary[4] $\Upsilon(a_r) \approx 1 \;\;\Rightarrow\;\; h(a_r) \approx \frac{1}{2}$. Finally, if a_r is incorrectly classified and far from the decision boundary $\Upsilon(a_r) \gg 1 \;\Rightarrow\; h(a_r) \approx 1$. A similar argument applies to utterances belonging to word class $\mathcal{W}_2$. From the above analysis, MMIE appears to use mis-recognised utterances to update the parameters of the models such as to achieve better class separation. Furthermore, incorrectly classified tokens far from the decision boundary will receive a more substantial weighting in their derivatives.

In general, if we are dealing with isolated words with a uniform language model we can hypothesise the value of $\Upsilon(a_r)$ beyond which an utterance is clearly mis-recognised. For K different classes, we can say that if $\Upsilon(a_r) > K-1$ then a_r will be mis-recognised. Hence, we can hypothesise the decision boundary as being at $\Upsilon(a_r) = K/2$. Now, to allow the parameter estimation procedure to concentrate on utterances around the decision boundary, we introduce the following non-linear function

$$f_s(x) = \frac{1}{1+e^{-\nu(x+\xi)}} \tag{3.21}$$

applied to the mutual information measure for each utterance. The constants ν and ξ control the slope and mid-point of the sigmoid function. The non-linear MMIE criterion will be referred to as the Sigmoid-MMIE (SMMIE). The objective function of SMMIE is then defined as follows

$$f_{SMMI}(\lambda) = \frac{1}{R}\sum_{r=1}^{R} f_s(f_{MMI}(a_r,\lambda))$$

where

$$f_{MMI}(a_r,\lambda) = \log \frac{P_\lambda(a_r|\mathcal{M}_{w_r})P_\lambda(w_r)}{P_\lambda(a_r|\mathcal{M}_{rec})}$$

is the utterance-specific mutual information measure. In order to maximise $f_{SMMI}(\lambda)$ we use

$$\frac{\partial}{\partial\lambda} f_{SMMI}(\lambda) = \frac{1}{R}\sum_{r=1}^{R} \frac{\partial}{\partial x} f_s(x) \frac{\partial}{\partial\lambda} f_{MMI}(a_r,\lambda) \tag{3.22}$$

for $x = f_{MMI}(a_r,\lambda)$. The derivative of $f_s(x)$ is given by

$$\frac{\partial}{\partial x} f_s(x) = \nu\, f_s(x)\,(1 - f_s(x))$$

Figure 3.1 depicts a sigmoid weighting function and its derivative applied to the mis-classification measure $\Upsilon(a_r)$. The plot shows that the overall derivative of the objective function will be dominated by utterances with $\Upsilon(a_r) \to 1.6$ which is the hypothesised decision boundary for 8 classes with a uniform language model. The slope parameter ν will

[4] For two classes the decision boundary is defined exactly at $\Upsilon(a_r) = 1$.

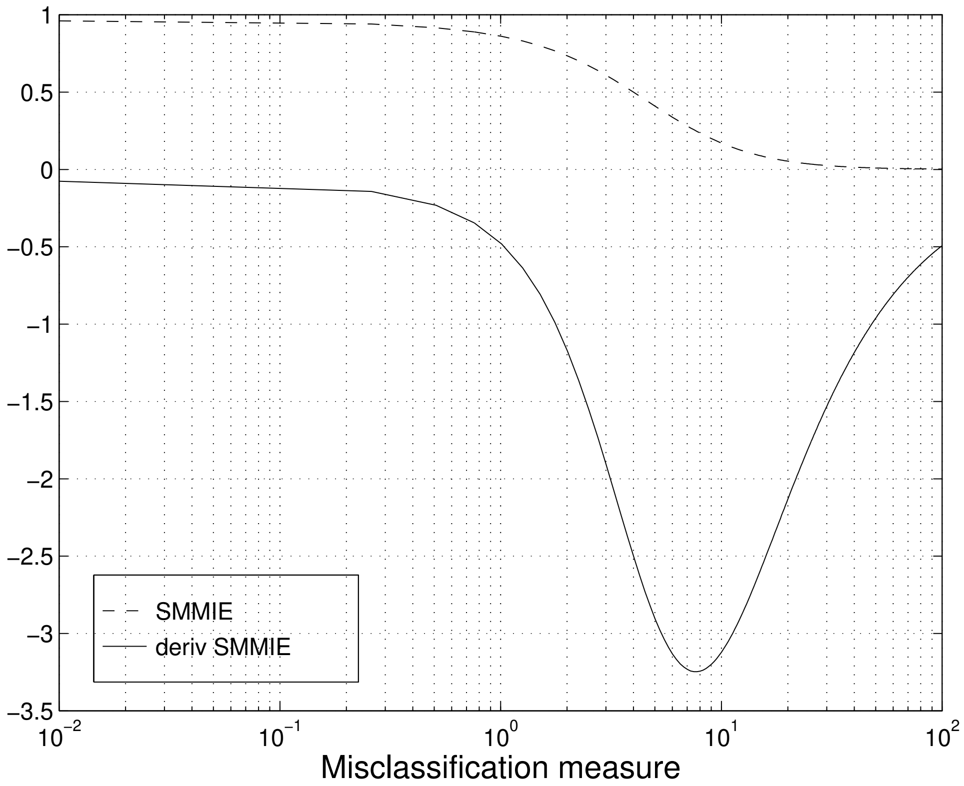

Figure 3.1: SMMI objective function and its derivative wrt $\Upsilon(a_r)$ ($\nu = 2.0$, $\xi = 1.6$)

have to be determined empirically. In general, a small value of ν will give a wider band of highly weighted utterances around the decision boundary and a larger value will make the peak of the derivative sharper. With smaller values of ν the value of the SMMI objective function will be dominated by relatively few utterances falling within the narrow boundary region.

The non-linear weighting function attempts to correct a criticism of MMIE i.e. the fact that it gives higher weighting to training tokens far from the decision boundary. In its formulation, SMMIE is somewhat similar to the MCE approach, however, it uses full likelihoods and attempts to discriminate against multiple confusable utterances at the same time. Comparing this weighting scheme with those implicitly used by various conventional classification schemes, it is of the same form as the multi-layer perceptrons. Niles et al. [83, 49] have shown how the mean squared error minimisation of HMM parameters within an MLP framework gives most weight to borderline tokens, whereas MMIE training gives rise to a weighting function which gives little weight to easily recognised utterances and larger weights to utterances far from the decision boundary.

With this in mind, the SMMIE approach is expected to perform at least as well as MMIE, with possible improvements in generalisation when the training utterances are not representative of the task.

3.7 Summary

This chapter has reviewed a number of current approaches to optimising the parameters of HMMs. Maximum Likelihood estimation (MLE) is the most common re-estimation criterion, however the assumptions which guarantee its optimality are never satisfied in practice. This has led to the conclusion that other training schemes such as MMIE and CMLE can yield better estimates of the HMM parameters with potential improvements in recognition performance. The MMIE/CMLE approaches seem reasonable since recognition is usually performed by finding the model with the largest *a posteriori* probability of generating the spoken utterance. Unfortunately the expression to optimise is rather complex and often has to be approximated.

Training schemes aiming at minimising the classification error of the recogniser were shown to use similar statistics to the MMIE algorithm, often gathered in a simpler way. In the spirit of the MCE approach, the MMIE objective function was modified to incorporate an utterance-specific non-linear weighting function. The modified objective function attempts to bias the re-estimation procedure towards utterances close to the decision boundary. The algorithm is expected to achieve better generalisation when there is a limited amount of training data and exhibit robustness to utterances which are not "representative" of the task. The following chapter will discuss different optimisation algorithms which can be used to implement MMIE training of HMM parameters.

Chapter 4

MMIE of HMM parameters

This chapter is devoted to the presentation of various optimisation algorithms which can be used to re-estimate the parameters of a hidden Markov model using a discriminative objective function such as the MMIE criterion. First, the general approach to function optimisation is reviewed. This is followed by a discussion of a variety of gradient-based training algorithms. Finally, we describe the discriminative training framework within which the MMIE training algorithm will be evaluated.

4.1 Introduction

As discussed in chapter 3, the successful application of HMMs to speech recognition tasks depends entirely on the definition of a meaningful objective criterion and the availability of a fast and effective training algorithm. The well-known Baum-Eagon inequality [10] provides an effective iterative scheme for finding a local maximum for homogeneous polynomials with positive coefficients over a domain of probability values. Polynomials of this type appear in various statistical problems dealing with the estimation of probabilistic functions of Markov chains using the maximum likelihood criterion. It is probably accurate to say that the powerful Baum-Welch algorithm is one of the main reasons for the introduction and wide use of HMMs in current speech recognition systems. In the previous chapter we showed that due to the modelling assumptions of the Markov model, it may be beneficial to estimate the parameters of our acoustic model using some other criterion such as MMIE/CMLE. Optimisation of these objective functions involves dealing with the ratio of two expressions, hence they are commonly referred to as rational objective functions.

4.2 Training algorithms

Although the algorithms discussed in this chapter vary substantially in their motivation, application and behaviour, they have the common property of being iterative ascent algorithms. Iterative refers to the way an optimal solution is reached through the generation of a succession of parameter sets from the current model. Ascent characterises the series of generated parameter sets such that the value of the objective function increases as training progresses. Ideally, the sequence of points generated in such a manner converges to a solution of the problem in a finite number of steps. Convergence and the computational effort involved are the two most important properties of an iterative training scheme. Computational effort is usually influenced by the number of function evaluations and degree of detail needed for the algorithm to succeed. Many of the algorithms discussed later in this chapter have been "borrowed" from the recently revived study of connectionist models [101].

We can distinguish between two rather different types of algorithms. In the Baum-Welch algorithm, finding a local maximum of the likelihood function is accomplished within the E-M (Expectation-Maximisation) framework. First, a state/frame alignment is produced for the training utterances. Then, the parameters of the models are re-estimated to maximise the value of the likelihood function, given the already calculated state/frame allocation. The second class of algorithms are the so called "general minimum methods". These methods are popular since many complex real world systems can be characterised by a function whose minimum will coincide with the optimal system's behaviour. As Thomas J. Acton [1] describes them

> *"They are the first refuge of the computational scoundrel, and one feels at times that the world would be a better place if they were quietly abandoned. But even if these techniques are frequently misused, it is equally true that there are problems for which no alternative solution method is known - and so we shall discuss them."*

To provide consistency with the literature, in describing these methods we will use the terms "descent" and "minimise" rather than "ascent" and "maximise". This is not a problem since every maximisation problem can be reformulated as a minimisation problem. How we intend to use the method being described should be clear from the context. A major advantage of the "minimum methods" family of algorithms is that they are relatively simple to perceive and implement. For example, many texts on numerical optimisation start by visualising an optimisation problem in two dimensions. The unpleasant fact is that these methods typically require many iterations to converge, and the methods providing faster convergence are often too complex to implement for larger systems.

4.3 The E-M algorithm and rational objective functions

Optimising the HMM parameters according to the MMIE criterion described in the previous chapter, typically, requires finding the local maximum of a rational objective function over domains of probability values. Recently [46], the Baum-Eagon inequality was extended to rational objective functions. In theory, the extension allows one to use an iterative E-M-like algorithm for maximising a variety of discriminative objective functions. In general, a rational objective function is defined as

$$\mathcal{R}(X) = \frac{\mathcal{S}_1(X)}{\mathcal{S}_2(X)}$$

where analogous to section 2.3.2, $\mathcal{S}_1(X), \mathcal{S}_2(X)$ are polynomials with real coefficients in variables $X = \{x_{ij}\}$. The problem of finding a growth transformation for $\mathcal{R}(X)$ is first reduced to one of finding a growth transformation for a specifically formed polynomial $\mathcal{P}_x(X)$. Gopalakrishnan et al. have shown that for any $X \in \mathcal{D}$ there exists a polynomial $\mathcal{P}_x(X)$ such that if $\mathcal{P}_x(T(X)) > \mathcal{P}_x(X)$, then $\mathcal{R}(T(X)) > \mathcal{R}(X)$. Typically, $\mathcal{P}_x(X)$ is a non-homogeneous polynomial, whereas the original proof of the Baum-Eagon inequality used the homogeneity property of a polynomial as a necessary condition. The work in [46] contains an extension of the original theorem which proves that the Baum-Eagon inequality also remains true for non-homogeneous polynomials with non-negative coefficients. Analogous to equation 2.9, the growth transformation for $\mathcal{R}(X)$ is given by

$$T(x_{ij}) = \frac{x_{ij}\left\{\frac{\partial}{\partial x_{ij}}\mathcal{R}(X) + C\right\}}{\sum_{j=1}^{q_i} x_{ij}\left\{\frac{\partial}{\partial x_{ij}}\mathcal{R}(X) + C\right\}} \tag{4.1}$$

Since faster convergence requires a small constant C and the determination of such a constant was found to be rather involved, in the experimental evaluation of the algorithm in [46] an approximate version of the growth transformation was used. At each iteration, the value of C was chosen to make all derivatives positive according to

$$C(x) = \max\left\{\max_{ij}\left\{\frac{\partial}{\partial x_{ij}}\mathcal{R}(X)\right\}, 0\right\} + \epsilon \tag{4.2}$$

where ϵ is a small positive constant. Unfortunately, for the above chosen value of C, the convergence properties of the algorithm are not guaranteed.

In [46], the extended Baum-Welch algorithm was used to estimate the parameters of the IBM 20,000 word recognition system according to the H-criterion. The algorithm was compared to a gradient hill-climbing algorithm in terms of the percentage improvement in the value of the objective criterion. After six iterations, the extended Baum-Welch algorithm improved the value of the objective function by 22.4% as opposed to a negligible improvement of 0.5% provided by the gradient-based algorithm.

In its original derivation, the algorithm was only applicable to HMMs with discrete output distributions. An extension of the algorithm to continuous density HMMs was proposed by Normandin and Morgera in [85]. Since then, the algorithm has been used extensively by the speech recognition group at CRIM [24, 23] to carry out MMIE training of HMMs for connected digit recognition. On average the algorithm required 8-12 iterations to provide optimal performance [84]. Beyond this, chaotic behaviour of the objective function was often observed, which was attributed to the modified[1] gradient expressions used.

When this research began in 1990, the extension of the algorithm to CMD-HMMs was not available and the empirical evidence of its success was not yet established. Consequently, our study of discriminative training was focused on investigating the applicability of conventional gradient search techniques to MMIE training of the HMM parameters. The remaining parts of this chapter describe these alternative methods.

[1]Similarly to Merialdo [76], Normandin made use of modified gradient expressions to direct the re-estimation procedure away from small-valued parameters.

4.4 Minimum methods

In this section we review the basic techniques used for iteratively solving unconstrained optimisation problems. A general optimisation problem can be expressed in the form:

$$\text{minimise} \qquad f(\boldsymbol{x}) \qquad \text{subject to} \qquad \boldsymbol{x} \in \Lambda \tag{4.3}$$

where f is a real-valued function and Λ is a subset of the n-dimensional Euclidean space E^n. Throughout this discussion Λ corresponds to the completely unconstrained set, however, stochastic constraints on certain parameters can be automatically accommodated via suitable mappings on the corresponding variables. The fact that these methods can be applied to any unconstrained problem makes them particularly suitable for optimisation of hybrid HMM/neural network architectures. However, their practical utility is entirely dependent on the complexity of the method and its speed of convergence.

In formulating 4.3, one is explicitly searching for a global extremum point of f over Λ. In practice, however, due to computational and theoretical constraints only a local solution can be found. Using function characteristics in the proximity of the current parameter vector optimisation can be achieved only locally. In general, global solutions can be found only if the function is unimodal, in which case any local critical point is a global one too. Using differential calculus, the point $\tilde{\boldsymbol{x}}$ at which f achieves a locally minimal value is characterised by $\boldsymbol{\nabla} f(\tilde{\boldsymbol{x}}) = 0$ and the Hessian matrix of second derivatives $\boldsymbol{\nabla}^2 f(\tilde{\boldsymbol{x}})$ is positive semi-definite. The vector of first derivatives $\boldsymbol{\nabla} f(\tilde{\boldsymbol{x}})$ is given by

$$\boldsymbol{\nabla} f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial}{\partial x(1)} f(\boldsymbol{x}) \\ \frac{\partial}{\partial x(2)} f(\boldsymbol{x}) \\ \cdots \\ \frac{\partial}{\partial x(i)} f(\boldsymbol{x}) \\ \cdots \\ \frac{\partial}{\partial x(n)} f(\boldsymbol{x}) \end{pmatrix}$$

where $x(i)$ is the i^{th} parameter in the function. In describing the optimisation algorithms, sometimes we will have to refer to the parameter set at a particular iteration of the training procedure. The notation $\boldsymbol{x}_k$ will be used to denote the parameter vector of the function at iteration k. The change in parameter value will be denoted by $\Delta \boldsymbol{x}_k$.

4.4.1 HMM parameter derivatives

In order to perform any of the gradient based algorithms described in the following sections we need to be able to differentiate the chosen objective function with respect to the HMM parameter set λ. In this section we will define the gradient expressions for the different HMM parameters. In previous chapters, the training set was assumed to consist of a number of samples (w, a) where w describes the identity of the spoken utterance and a is the acoustic representation. In this section we will redefine the training data as follows. A training data set $\boldsymbol{\mathcal{O}}$ is defined as being comprised of R training utterances, i.e.

$$\boldsymbol{\mathcal{O}} = \{\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_r, \ldots, \mathcal{O}_R\}$$

Each training utterance $\mathcal{O}_r$ is a sequence of observation vectors

$$\mathcal{O}_r = \boldsymbol{o}_{r,1}, \boldsymbol{o}_{r,2}, \ldots, \boldsymbol{o}_{r,t}, \ldots, \boldsymbol{o}_{r,T_r}$$

where T_r is the number of frames in $\mathcal{O}_r$. Finally, each observation vector $\boldsymbol{o}_{r,t}$ consists of D components

$$\boldsymbol{o}_{r,t} = \begin{pmatrix} o_{r,t,1} \\ o_{r,t,2} \\ \cdots \\ o_{r,t,d} \\ \cdots \\ o_{r,t,D} \end{pmatrix}$$

Using the above notation, the MMIE objective function is defined as

$$f_{MMI}(\lambda) = \frac{1}{R}\sum_{r=1}^{R}\Big\{\log P_\lambda(\mathcal{O}_r|\mathcal{M}_{w_r}) - \log P_\lambda(\mathcal{O}_r)\Big\} \quad (4.4)$$

$$= \frac{1}{R}\sum_{r=1}^{R}\log P_\lambda(\mathcal{O}_r|\mathcal{M}_{w_r}) - \frac{1}{R}\sum_{r=1}^{R}\log P_\lambda(\mathcal{O}_r|\mathcal{M}_{rec}) \quad (4.5)$$

where $\mathcal{M}_{w_r}$ is the model corresponding to the utterance's correct transcription and $\mathcal{M}_{rec}$ is the recognition model. The model $\mathcal{M}_{rec}$ is always synthesised from the family of HMMs according to the recognition grammar, hence, it is assumed that it also includes the language model scores at the unit boundary transitions. Inspection of equation 4.5 reveals that the MMI objective function is computed as the difference between two log-likelihoods calculated using different models. The first term on the right in equation 4.5 will be referred to as the *numerator likelihood* and the second term will be referred to as the *denominator likelihood*. We will define the log-likelihood of the training set $\mathcal{O}$ given a model $\mathcal{M}$ as

$$\mathcal{L}_\lambda(\boldsymbol{\mathcal{O}}|\mathcal{M}) = \frac{1}{R}\sum_{r=1}^{R}\log P_\lambda(\mathcal{O}_r|\mathcal{M}) \quad (4.6)$$

The partial derivatives of $\mathcal{L}_\lambda(\boldsymbol{\mathcal{O}}|\mathcal{M})$ with respect to the various HMM parameters are listed below. All intermediate derivations are given in appendix B.

1. **Transition probability** $a_{i,j}$ (combining B.2, B.9)

$$\frac{\partial}{\partial h_{i,k}}\mathcal{L}_\lambda(\boldsymbol{\mathcal{O}}|\mathcal{M}) = \quad (4.7)$$

$$\frac{1}{R}\sum_{r=1}^{R}\frac{1}{\mathcal{L}_\lambda(\mathcal{O}_r|\mathcal{M})}\sum_{t=1}^{T_r}\sum_{j=1}^{N}\alpha_i(t-1)a_{i,j}(\delta_{k,j}-a_{i,k})b_j(o_{r,t})\beta_j(t)$$

subject to

$$a_{i,j} = \frac{f_a(h_{i,j})}{\sum_k f_a(h_{i,k})} \qquad \text{and} \qquad f_a(x) = e^x$$

2. **Mixture component weight** $c_{j,m}$ (combining B.2, B.10, B.24)

$$\frac{\partial}{\partial u_{j,k}}\mathcal{L}_\lambda(\mathcal{O}|\mathcal{M}) = \frac{1}{R}\sum_{r=1}^{R}\sum_{t=1}^{T_r}\mathcal{C}(r,t,j)\ \sum_{m=1}^{M} b_{j,m}(\boldsymbol{o}_{r,t})c_{j,m}(\delta_{k,m}-c_{j,k}) \tag{4.8}$$

subject to

$$c_{j,m} = \frac{f_c(u_{j,m})}{\sum_k f_c(u_{j,k})} \qquad \text{and} \qquad f_c(x) = e^x$$

3. **Mean** $\mu_{j,m,d}$ *(diagonal covariance)* (combining B.2, B.10, B.13, B.16)

$$\frac{\partial}{\partial \mu_{j,m,d}}\mathcal{L}_\lambda(\mathcal{O}|\mathcal{M}) = \frac{1}{R}\sum_{r=1}^{R}\sum_{t=1}^{T_r}\mathcal{C}(r,t,j)\ c_{j,m}b_{j,m}(o_{r,t})\left[\frac{o_{r,t,d}-\mu_{j,m,d}}{\sigma^2_{j,m,d}}\right] \tag{4.9}$$

4. **Mean vector** $\boldsymbol{\mu}_{j,m}$ *(full covariance)* (combining B.2, B.10, B.13, B.14)

$$\frac{\partial}{\partial \boldsymbol{\mu}_{j,m}}\mathcal{L}_\lambda(\mathcal{O}|\mathcal{M}) = \frac{1}{R}\sum_{r=1}^{R}\sum_{t=1}^{T_r}\mathcal{C}(r,t,j)\ c_{j,m}b_{j,m}(\boldsymbol{o}_{r,t})\boldsymbol{W}^{-1}_{j,m}(\boldsymbol{o}_{r,t}-\boldsymbol{\mu}_{j,m}) \tag{4.10}$$

5. **Variance** $\sigma^2_{j,m,d}$ (combining B.2, B.10, B.13, B.21)

$$\frac{\partial}{\partial z_{j,m,k}}\mathcal{L}_\lambda(\mathcal{O}|\mathcal{M}) = \frac{1}{R}\sum_{r=1}^{R}\sum_{t=1}^{T_r}\mathcal{C}(r,t,j)\ c_{j,m}b_{j,m}(\boldsymbol{o}_{r,t})\frac{1}{2}\left\{\frac{(o_{r,t,d}-\mu_{j,m,d})^2}{\sigma^2_{j,m,d}}-1\right\} \tag{4.11}$$

subject to

$$\sigma^2_{j,m,d} = f_{\sigma^2}(z_{j,m,k}) \qquad \text{and} \qquad f_{\sigma^2}(x) = e^x$$

6. **Covariance matrix** $\boldsymbol{W}^{-1}_{j,m}$ (combining B.2, B.10, B.13, B.18)

$$\frac{\partial}{\partial \boldsymbol{L}_{j,m}}\mathcal{L}_\lambda(\mathcal{O}|\mathcal{M}) = \frac{1}{R}\sum_{r=1}^{R}\sum_{t=1}^{T_r}\mathcal{C}(r,t,j)\ c_{j,m}b_{j,m}(\boldsymbol{o}_{r,t})\left\{(\boldsymbol{L}^{-1}_{j,m})'-(\boldsymbol{o}_{r,t}-\boldsymbol{\mu}_{j,m})(\boldsymbol{o}_{r,t}-\boldsymbol{\mu}_{j,m})'\boldsymbol{L}_{j,m}\right\} \tag{4.12}$$

subject to

$$\boldsymbol{W}^{-1}_{j,m} = \boldsymbol{L}_{j,m}\boldsymbol{L}'_{j,m}$$

where $\boldsymbol{L}_{j,m}$ is the lower Choleski factor of $\boldsymbol{W}^{-1}_{j,m}$.

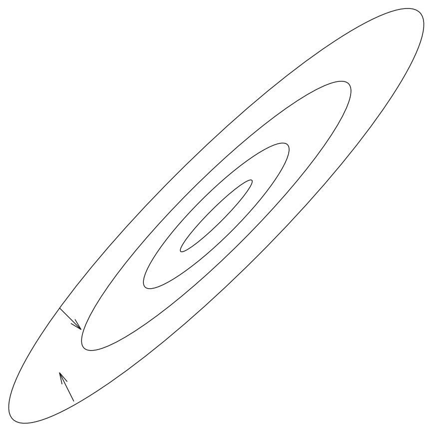

Figure 4.1: Difficulties with steepest descent

The common factor $\mathcal{C}(r,t,j)$ is defined as

$$\mathcal{C}(r,t,j) = \frac{1}{P_\lambda(\mathcal{O}_r|\mathcal{M})}\left\{\sum_{i=1}^{N}\alpha_i(t-1)a_{i,j}\right\}\beta_j(t) \tag{4.13}$$

$$= \frac{1}{P_\lambda(\mathcal{O}_r|\mathcal{M})}\frac{\alpha_j(t)\beta_j(t)}{b_j(\boldsymbol{o}_{r,t})} \tag{4.14}$$

4.4.2 Steepest descent

The method of steepest descent is the simplest method of minimising a multivariate function. At each iteration the values of the parameters are modified in the direction in which the objective function decreases most rapidly. The direction is described by the gradient of the surface at the current point in the parameter space. The magnitude of the modification is a constant proportion of the magnitude of the gradient. The proportion is commonly named the learning rate or step size and the parameter update is expressed as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta\boldsymbol{\nabla} f(\boldsymbol{x}_k) \tag{4.15}$$

where, following previous notation, $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$ are the old and the new parameter vectors respectively, and η is the learning rate.

Ideally η should be set to the value at which $f(\boldsymbol{x}_k - \eta\boldsymbol{\nabla} f(\boldsymbol{x}_k))$ is minimised. This is equivalent to performing a line search along the direction defined by the gradient. Unfortunately such schemes are impractical for complex functions of several thousand variables. For example, in the context of optimising the parameters of a set of HMMs, the likelihood function evaluations over several hundreds of training utterances is prohibitively expensive.

4.4.3 Momentum

Although steepest descent can be an efficient method it scales up poorly as tasks become larger and more complex. This is largely due to the fact that in a system with several thousand parameters the optimised function surface possesses properties which make this procedure slow to converge. The reasons behind this involve the magnitude of the components of the gradient vector and the selected direction of the search. Two main problems can be identified. The magnitude of the partial derivative of the function with respect to a parameter may be such that modifying a parameter by a constant proportion of that derivative will yield a minor reduction in the cost function. This occurs in two cases. Where the function surface is fairly flat along a parameter dimension, the derivative of the parameter will be small in magnitude. Thus the value of the parameter is adjusted by a small amount and many steps are required to achieve a significant reduction in the cost function. Alternatively, where the function surface is highly curved along the parameter dimension, the derivative of that parameter is large in magnitude. Consequently the value of the parameter is adjusted by a large amount which can result in oscillation.

Another reason for the slow convergence of the steepest descent algorithm is that the direction of the gradient vector may not point directly towards the minimum of the objective function. This problem is illustrated in figure 4.1. where contours of the function are long and narrow, hence the gradient points the quickest way to the floor of the valley but not to its centre.

Momentum attempts to solve these problems by introducing an extra term in the parameter update equation.

$$\begin{aligned} \Delta \boldsymbol{x}_k &= (1-\zeta)\eta \boldsymbol{\nabla} f(\boldsymbol{x}_k) + \zeta \Delta \boldsymbol{x}_{k-1} && (4.16) \\ &= (1-\zeta)\eta \sum_{i=0}^{t} \zeta^i \boldsymbol{\nabla} f(\boldsymbol{x}_{k-i}) && (4.17) \end{aligned}$$

where ζ is the momentum factor that determines the relative contribution of the current and past partial derivatives to the current change in parameter value. This contribution is the exponentially weighted sum of the parameter's current and past partial derivatives where ζ is the base and the time from the current iteration is the exponent. Without the use of momentum (ζ set to 0) the update rule is identical to the steepest descent rule. The use of momentum has the effect of damping oscillations in values of the parameters. When consecutive derivatives of a parameter possess the same sign, the exponentially weighted sum grows large in value and the parameter is adjusted by a large amount. Similarly when consecutive derivatives have opposite signs, this sum becomes small in magnitude and the parameter is adjusted by a small amount. Two known limitations of momentum can be specified. First, there exists an upper bound on how large an adjustment momentum can make to a parameter. A second limitation is that the exponentially weighted sum may have a sign opposite to the sign of the parameter's current derivative.

4.4.4 Newton's method and conjugate directions

In this approach it is assumed that the function being optimised can be locally approximated by a quadratic function (truncated Taylor series expansion)

$$f(\boldsymbol{x}) \simeq f(\boldsymbol{x}_k) + \boldsymbol{\nabla} f(\boldsymbol{x}_k)(\boldsymbol{x}_k - \boldsymbol{x}) + \frac{1}{2}(\boldsymbol{x}_k - \boldsymbol{x})'\boldsymbol{\nabla}^2 f(\boldsymbol{x}_k)(\boldsymbol{x}_k - \boldsymbol{x}) \tag{4.18}$$

The quadratic function on the right can be minimised exactly at

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[\boldsymbol{\nabla}^2 f(\boldsymbol{x}_k)\right]^{-1} \boldsymbol{\nabla} f(\boldsymbol{x}_k)' \tag{4.19}$$

Newton's method has the very desirable property that its order of convergence is two. However, this is only guaranteed if the algorithm is started sufficiently close to the optimal point. In order to exploit the algorithm at points remote from the solution the algorithm requires two modifications. The first modification addresses the problem that in expression 4.19 the objective may actually increase due to non-quadratic terms in $f(\boldsymbol{x}_k)$. This is rectified with the introduction of a search parameter η

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta \left[\boldsymbol{\nabla}^2 f(\boldsymbol{x}_{\boldsymbol{x}_k})\right]^{-1} \boldsymbol{\nabla} f(\boldsymbol{x}_k)' \tag{4.20}$$

where η is selected to minimise f. The second modification relates to the properties of $\boldsymbol{\nabla}^2 f(\boldsymbol{x}_k)$. In general, the second term on the right in the above expression yields a direction $\boldsymbol{u}_k$ of descent in the form

$$\boldsymbol{u}_k = -\boldsymbol{M}_k \boldsymbol{\nabla} f(\boldsymbol{x}_k)' \tag{4.21}$$

The descent property of $\boldsymbol{u}_k$ is only preserved if $\boldsymbol{\nabla} f(\boldsymbol{x}_k)\boldsymbol{M}\boldsymbol{\nabla} f(\boldsymbol{x}_k)' > 0$. This is usually achieved by requiring $\boldsymbol{M}$ to be positive definite. Using the above expression the methods of steepest descent and the pure form of Newton's method are automatically accommodated by setting $\boldsymbol{M}_k$ to the identity matrix $\boldsymbol{I}$ or $\boldsymbol{\nabla}^2 f(\boldsymbol{x}_k)$ respectively. An intermediate solution is offered by setting

$$\boldsymbol{M}_k = \left[\rho_k \boldsymbol{I} + \boldsymbol{\nabla}^2 f(\boldsymbol{x}_k)\right]^{-1} \tag{4.22}$$

where ρ_k is a non negative number for which $\boldsymbol{M}_k$ is positive definite. In the Levenberg-Marquardt type methods, a Choleski decomposition $\boldsymbol{L}\boldsymbol{L}'$ of $\left[\rho_k \boldsymbol{I} + \boldsymbol{\nabla}^2 f(\boldsymbol{x}_k)\right]$ is employed to check for positive definiteness. Since $\boldsymbol{L}\boldsymbol{L}'$ is relatively simple to compute, this can be used to iteratively increase the value of ρ_k until such factorisation becomes possible. Matrix inversion is further avoided by using the Choleski factors to compute the direction vector $\boldsymbol{u}_k$ through solving $\boldsymbol{L}\boldsymbol{L}'\boldsymbol{u}_k = \boldsymbol{\nabla} f(\boldsymbol{x}_k)$, which is relatively simple since $\boldsymbol{L}$ is triangular.

The Conjugate Gradient algorithm is motivated by the desire to accelerate the slow convergence of the steepest descent algorithm whilst avoiding the computation of the Hessian matrix. The method aims at a more efficient exploration of the search space through a sequence of linearly independent search directions. The algorithm was originally formulated and analysed for purely quadratic problems. Its extension to non-quadratic problems requires a line search to be performed along each new direction. Although the line search can be directed using cubic and quadratic fits, it still requires several evaluations of the function. In the classic application of the algorithm, it is always assumed that evaluating the function is computationally cheap when compared to evaluating the gradient vector. In the HMM framework, evaluating the MMIE objective function will require computing either the forward or the backward likelihood for both the numerator and denominator. Referring to section 4.4.1 it can be seen that in order to compute the derivative expressions we require both the α and the β matrices. Initially it seems that evaluating the likelihood function will require half the computation used to compute the derivatives. As will be discussed later, once the α matrix is computed, the backward pass can be heavily pruned using the lookahead information already stored in the α matrix, hence the computational effort for computing the gradients is much less than expected.

If we are considering maximising the MMIE objective function using Newton's method, a natural question to ask is how much computation is required to compute the Hessian of the MMIE objective function. In order to derive this number and interpret it let us consider the transition probabilities. From section 4.4.1, the derivative of the likelihood function $P_\lambda(\mathcal{O}|\mathcal{M})$ with respect to a transition a_{ij} is given by

$$\frac{\partial}{\partial a_{i,j}} P_\lambda(\mathcal{O}|\mathcal{M}) = \sum_{t=1}^{T} \alpha_i(t-1) b_j(\boldsymbol{o}_t) \beta_j(t) \tag{4.23}$$

For the second derivatives one has to consider differentiating the above equation with respect to another transition, say $a_{k,l}$

$$\begin{aligned}
&\frac{\partial^2}{\partial a_{i,j} \partial a_{k,l}} P_\lambda(\mathcal{O}|\mathcal{M}) = \\
&\quad \sum_{t=1}^{T} \left\{ \frac{\partial}{\partial a_{k,l}} \alpha_i(t-1)\ b_j(\boldsymbol{o}_t) \beta_j(t) + \alpha_i(t-1)\ b_j(\boldsymbol{o}_t) \frac{\partial}{\partial a_{k,l}} \beta_j(t) \right\} = \\
&\quad \sum_{t=1}^{T} \left\{ \left[\sum_{t_1=1}^{t-1} \alpha_k(t_1-1) b_l(\boldsymbol{o}_{t_1}) \bar{\beta}_l(t_1, t) \right] b_j(\boldsymbol{o}_t) \beta_j(t)\ + \right. \\
&\quad \left. \alpha_i(t-1)\ b_j(\boldsymbol{o}_t) \left[\sum_{t_1=t}^{T} \bar{\alpha}_k(t-1, t_1-1) b_l(\boldsymbol{o}_{t_1}) \beta_l(t_1) \right] \right\}
\end{aligned}$$

where $\bar{\beta}_l(t_1, t)$ is the probability of generating the observation sequence $\boldsymbol{o}_{t_1}^{t}$ starting from state l and $\bar{\alpha}_k(t, t_1)$ is the probability of generating the sequence $\boldsymbol{o}_t^{t_1}$ and finishing at state k. This is equivalent to finding the probability of generating all possible sub-sequences $\boldsymbol{o}_{t_1}^{t}$ starting in state i by taking transition $a_{i,j}$ and arriving at state l with the transition $a_{k,l}$. For an HMM with N states and an utterance of length T this will require computation of the order N^3T^2. With the resources available when this research began, experiments to evaluate such algorithms were considered impractical to perform.

Finally, due to the peculiarities of the HMM framework, the fore-mentioned second order algorithms were considered unsuitable for optimising any likelihood-based functions. The following sections will review three optimisation schemes which require only local first derivative information.

4.4.5 Local optimisation

Local optimisation methods consider local changes for each individual parameter. The transition from the steepest descent procedure to a more sophisticated local algorithm can be considered as implementing the following four heuristics proposed by Jacobs [55].

1. Individual learning rates should be employed for different function parameters. This follows the observation that the step size appropriate for some parameter is not necessarily appropriate for other parameters.

2. Following the fact that function surfaces may possess different properties along different regions of a single parameter dimension, every learning rate should be allowed to vary over time.

3. When the derivative of a parameter possesses the same sign for several consecutive steps, the learning rate for that parameter should be increased. When the sign of the derivative behaves in this manner, it is frequently the case that the function surface along that parameter dimension has a small curvature, and therefore continues to slope in the same direction for some significant distance. By increasing the learning rate for this parameter, the number of iterations required for the value of this parameter to traverse this distance can be reduced.

4. When the sign of the derivative of a parameter alternates for several iterations, the step size for that parameter should be decreased. Change in the sign of the derivative often indicates that the function surface at the current point in parameter space along that parameter dimension possesses a high curvature. In order to prevent the value of the parameter from oscillating, this value should be adjusted by a smaller amount.

In support of heuristic one, some preliminary experiments were carried out for optimising the mean vectors in an HMM system according to the MMIE objective function using the vanilla gradient descent procedure. The algorithm did not produce the desired result. This was traced to the magnitude of the adjustments made to the mean parameters at each iteration. The changes in value for most parameters were found to be unacceptably small to warrant any significant change in the value of the objective function. The reason for this is that the re-estimation formula for the means (see equation 4.9) involves the reciprocal of the corresponding variance parameter, and variances were found to vary by as much as 10^3 even within the same distribution (see figure 4.2). Consequently, the same learning rate parameter was too large for some parameters and too small for others. In order to prevent instability, the learning rate had to be set to a small number, consequently the gradient search was focused on means with small variances. However, in an HMM system with $10^3 - 10^5$ parameters to be updated at each iteration it is almost impossible to observe the desired improvement in the value of the objective function.

It is important to realise that in a system where different step sizes are employed to govern the changes of different function parameters, the chosen direction to move is not the direction of the gradient. Thus, local optimisation methods are not gradient descent

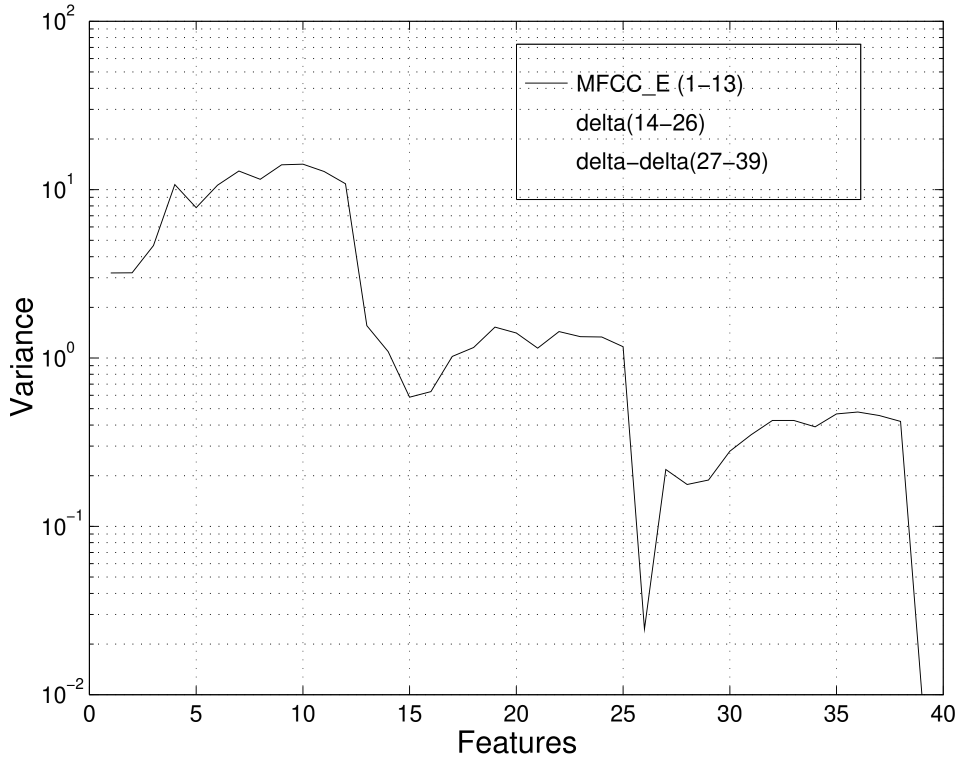

Figure 4.2: Plot of a `MFCC_E_D_A` variance vector from the ISOLET database

methods. The parameter update procedure utilises information from both partial derivatives and estimates of the curvature of the function's surface given by the step size adaptation for each individual parameter. Several possible implementations have been discussed in the literature which attempt to speed up the steepest descent algorithm. We shall consider three methods suitable for implementing MMIE optimisation of HMM parameters.

4.4.5.1 Delta-bar-Delta learning rule

The *delta-bar-delta* learning rule was proposed by Jacobs as an extension of the *delta-delta* learning rule introduced by the same author as a slight variation of the update strategy proposed by Sutton [105]. The rule consists of a parameter update rule and learning rate update scheme. Let $\boldsymbol{\eta}_k$ be the learning rate vector at iteration k and $\eta_k(i)$ the learning rate for parameter i. Then, $\boldsymbol{\eta}$ is updated as

$$\boldsymbol{\eta}_{k+1} = \boldsymbol{\eta}_k + \Delta\boldsymbol{\eta}_k$$

where $\Delta\boldsymbol{\eta}_k$ is a vector of changes for each learning rate, given by

$$\Delta\eta_k(i) = \begin{cases} \kappa & \text{if } \bar{\delta}_{k-1}(i)\nabla_i f(\boldsymbol{x}_k) > 0 \\ -\phi\eta_k(i) & \text{if } \bar{\delta}_{k-1}(i)\nabla_i f(\boldsymbol{x}_k) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.24}$$

and

$$\bar{\delta}_k(i) = (1-\zeta)\nabla_i f(\boldsymbol{x}_k) + \zeta\bar{\delta}_{k-1}(i)$$

The above equation represents the exponential average of current and past derivatives. In the original paper [55], Jacobs used $\delta_k(i) = \nabla_i f(\boldsymbol{x}_k)$, hence the name "*delta-bar-delta*". The rationale behind this method is as follows: if the current derivative and the exponential average of the previous derivatives have the same sign, then the learning rate for that weight is incremented by a constant amount κ. If the current derivative and the exponential average have opposite signs, then the learning rate is decremented by a proportion ϕ of its current value. This somehow supports the heuristic that if the parameter changes are oscillating, the minimum is somewhere between the oscillations, hence the reduced learning rate. Overall, the *delta-bar-delta* rule increments learning rates linearly, but decrements them exponentially. Linear increment allows a gradual growth of the value of the learning rate, whilst, exponential decrement ensures rapid decrease and positive value.

4.4.5.2 The RProp algorithm

The algorithm described below is a slight variation of the RProp procedure discussed in [102]. The algorithm can also be considered as a variation of the *delta-bar-delta* update rule where a separate learning rate is employed for each parameter, however, parameter updates are carried out using only the sign of the derivative. The learning rate update rule is given by

$$\eta_k(i) = \begin{cases} \min(1.2\eta_{k-1}(i), \eta_{max}) & \text{if } \nabla_i f(\boldsymbol{x}_k)\nabla_i f(\boldsymbol{x}_{k-1}) > 0 \\ \max(0.5\eta_{k-1}(i), \eta_{min}) & \text{if } \nabla_i f(\boldsymbol{x}_k)\nabla_i f(\boldsymbol{x}_{k-1}) < 0 \\ \eta_{k-1}(i) & \text{otherwise} \end{cases} \tag{4.25}$$

and the change in parameter $\boldsymbol{x}_k(i)$ is given by

$$\Delta x_k(i) = sign[\nabla_i f(\boldsymbol{x}_k)]\eta_k(i) \tag{4.26}$$

The change in parameter value is the magnitude of the learning rate for that parameter in the direction of the derivative. Learning rates are incremented and decremented exponentially. In the original version of the algorithm, no parameter update takes place if the current and the previous derivative possess different signs. When dealing with the heterogeneous set of parameters in an HMM, it was found beneficial to set different η_{min} and η_{max} for each different type of parameters e.g. transitions, mixture weights, means and variances.

4.4.5.3 Approximate second order

This method can be considered as implementing the update rule in Newton's method using an approximation to the Hessian matrix $\boldsymbol{\nabla}^2 f(\boldsymbol{x}_k)$. Two simplifying assumptions can be made. First, all HMM parameters can be assumed to affect the value of the function independently, e.g.

$$\boldsymbol{\nabla}^2 f(\boldsymbol{x}_k) = diag\left\{\frac{\partial^2 f(\boldsymbol{x}_k)}{\partial x_k(1)^2}, \frac{\partial^2 f(\boldsymbol{x}_k)}{\partial x_k(2)^2}, \ldots, \frac{\partial^2 f(\boldsymbol{x}_k)}{\partial x_k(n)^2}\right\}$$

Second, the value of $\frac{\partial^2 f(\boldsymbol{x}_k)}{\partial x_k(i)^2}$ can be approximated by the difference of first derivatives

$$\frac{\partial^2 f(\boldsymbol{x}_k)}{\partial x_k(i)^2} \approx \frac{\nabla_i f(\boldsymbol{x}_k) - \nabla_i f(\boldsymbol{x}_{k-1})}{\Delta x_{k-1}(i)} \tag{4.27}$$

substituting equation 4.27 into equation 4.20 gives

$$\Delta x_k(i) = \eta \frac{\Delta x_{k-1}(i)}{\nabla_i f(\boldsymbol{x}_k) - \nabla_i f(\boldsymbol{x}_{k-1})} \nabla_i f(\boldsymbol{x}_k) \tag{4.28}$$

For η set to 1.0 expression 4.28 transforms into the update strategy of Fahlman's *QuickProp* [40]. The behaviour of the update rule given by equation 4.28 with $\eta = 1.0$ is as follows. If the current gradient is smaller than the previous one but in the same direction, the parameter will change again in the same direction. The step taken may be large or small depending on how much the gradient was reduced by the previous step. If the current slope is in the opposite direction from the previous one, then we have stepped beyond the minimum. In this case, the next step will place us somewhere between the current and the previous position. The third case occurs when the current gradient is in the same direction as the previous but is of the same size or larger in magnitude. If we were to blindly follow the formula we would end up taking an infinite step or moving in the wrong direction. The third case occurs naturally since the update rule given by equation 4.20 will converge to the nearest turning point. In order to handle this special case, we adopt the method used by Fahlman in *QuickProp*. No parameter change is allowed to be greater in magnitude than φ times the previous update for that parameter. If the change computed by the update formula is too large, infinite or in the opposite direction to the current gradient, we instead use φ times the previous change as the current change. Fahlman found that the optimal value of φ is problem specific and for his experiments he used $\varphi = 1.75$. He also observed that if φ is too large the system behaved chaotically and failed to converge. A bootstrap process is also used to provide initial values of the parameter changes. More generally, if the previous gradient is zero or non-existent then the current change in parameter is calculated using plain gradient descent.

4.5 Discriminative training framework

4.5.1 Overview

The MMIE objective function is a rational objective function whose numerator is based on the likelihood of each utterance computed according to the correct transcription and whose denominator is based on statistics computed by matching each training utterance against the recognition grammar. Inspection of the derivative expressions given in section 4.4.1 reveals that similarly to the update equations in the Baum-Welch algorithm (section 2.3.2) the parameter derivatives require the calculation of the forward/backward probabilities for each training utterance (section 2.2.3). The rational nature of the MMIE objective function suggests a logical way of structuring the training process. The discriminative training framework which will be used to carry out MMIE training is outlined in figure 4.3.

The top left branch of the diagram shows the calculation of the forward/backward probabilities for each utterance using the generic recognition model e.g. the looped phonetic model in the case of continuous phone recognition. The right branch of the diagram shows the calculation of the forward/backward probabilities from the alignment of each training utterance against the "correct" HMM. In both cases the forward/backward probabilities are used to accumulate statistics which will be used in the parameter re-estimation procedure. In the case of the Baum-Welch[2] algorithm these statistics are in the form of state/mixture occupancy counts and expected values of the distribution parameters. In the case of MMIE optimisation, the statistics are the derivative expressions of the log-likelihood function with respect to each HMM parameter. In each case, after all training utterances have been processed, the collected statistics are stored in external files. The logical separation between the calculation of derivatives and the actual parameter update procedure allows for the relatively easy implementation of MMIE variants such as the H-criterion. Furthermore, the calculation of derivatives can be performed concurrently on multiple processors by splitting the training data into several equal chunks and gathering statistics for each of them separately.

The bottom branches of the diagram in figure 4.3 show the two post-processing steps which perform the re-estimation of the HMM parameters. The module on the right shows the Maximum Likelihood estimation step where the statistics from the "correct" alignments are used to adjust the parameters of the HMMs so as to increase the value of the likelihood objective function. In this case, the statistics are automatically assumed to be count-based although it is possible to optimise the likelihood function using a gradient search technique. The corresponding right branch of the digram performs HMM parameter updates according to a discriminative objective function. In the case of MMIE, the statistics accumulated from the denominator likelihood are simply subtracted[3] from the corresponding statistics gathered in the numerator pass. The resulting derivatives are then used by any of the previously discussed gradient-based algorithms to carry out adjustments to the HMM parameters. Virtually all "local" optimisation techniques involve a quantity which describes past changes in the values of the parameters. Such data is accumulated and stored into the "Update statistics" file as the training progresses from one iteration to another.

Although intuitively very similar to MMIE, the SMMIE objective function cannot be used directly in the above re-estimation framework. This is because the derivatives from the

[2]Incidently, the right branch of the diagram in figure 4.3 summarises the organisation and behaviour of the `HERest` re-estimation tool from HTK [117].

[3]Re-estimation is carried out in the *log*-domain.

numerator and denominator of the MMIE objective function are accumulated separately and the combination of the two is carried out as a post-processing step. Consequently, the update routines have no knowledge of the mutual information measure for each training utterance and subsequently, the derivative weighting cannot be carried out. In order to implement SMMIE training, the original framework of figure 4.3 was enhanced to incorporate a routine which collects statistics from the numerator and denominator simultaneously - figure 4.4. Thus, the utterance specific derivative weighting can be carried out as soon as the probabilities from the numerator/denominator α/β-passes are available. The parameter update procedure is based on the MMI re-estimation module from the original framework.

4.5.2 Implementation

The above described discriminative training frameworks were implemented as extensions of the HTK toolkit [117]. In particular, the α/β passes and accumulation of derivatives were implemented as modifications of a previously developed generic implementation of the embedded Baum-Welch algorithm (`HNRest`) [61]. For each training utterance, the tool loads a specified network file which describes the structure of the model to be synthesised (figure 4.3). An HMM instance is created and attached to each node in the network. The full forward (α) and backward (β) probabilities are then computed for each HMM instance. In order to save memory, the α and β matrices are computed column by column, and only columns corresponding to active models are created. Once, the forward and the backward probabilities are calculated, a final pass through the frames of the utterance is made in order to calculate and accumulate the derivatives of the likelihood function with respect to the parameters of each HMM. After all training utterances have been processed, the tool dumps the accumulated derivatives to external binary files.

The implementation of SMMIE is based on a modified version of `HNRest` which simultaneously maintains two sets of derivative accumulators and aligns training utterances against two separate networks at the same time. In general, the modular fashion in which `HNRest` was originally designed allowed for a rapid reorganisation of the code with minimal extra programming effort.

The calculation of the α/β probabilities is carried out within the token passing paradigm [115] where each forward/backward probability is represented by a token held within the corresponding HMM state. The α and β matrices created for each training utterance are potentially very large. In practice, their size is dramatically reduced by computing and storing columns of α and β only for HMM instances which are within the beam of active models. The width of the beam is controlled by a forward and a backward pruning thresholds. At the start of the utterance, there is only one model in the beam. At each time frame, partial paths are extended by, first, propagating tokens between the states of the active models and, second, by propagating tokens from the final states of active models into their successors. At this stage, many previously inactive models are brought into the beam. This corresponds to hypothesising all possible extensions of the currently maintained partial paths which are consistent with the structure of the model. After this step, all active models are examined and the maximum forward probability score is found. The beam is then narrowed, such that any models within the pruned beam have at least one state with an α score within a threshold from the maximum. The backward pruning is very similar to the forward pruning mechanism, however, one can make use of the information stored in the α matrix to further narrow the beam. In this respect, the α matrix can be considered as a lookahead which, for every state/time instance gives the probability of generating the not yet seen portion of the utterance. Hence on the backward pass, the maximum alpha-beta product is used to prune the models rather than the maximum value of β on its own. As will be seen later, the pruning mechanism plays an essential role in reducing the computational effort during parameter estimation.

4.6 Summary

Most implementations of MMIE for HMM parameters rely on gradient descent to optimise the objective function. An alternative approach is to make use of the the extension of the Baum-Eagon inequality to rational objective functions, however, the convergence of this algorithm depends entirely on the appropriate selection of the value of the constant used in the re-estimation formulae. In this chapter we considered three optimisation schemes which are easy to implement and only make use of local first derivative information. Previous results have indicated that these "local" methods work rather well when applied to the optimisation of the energy function used to train multi-layer perceptrons. Finally, a discriminative training framework was described which will be used to carry out MMIE/SMMIE training of HMM parameters. The following chapter presents an empirical evaluation of the MLE, MMIE and SMMIE training algorithms.

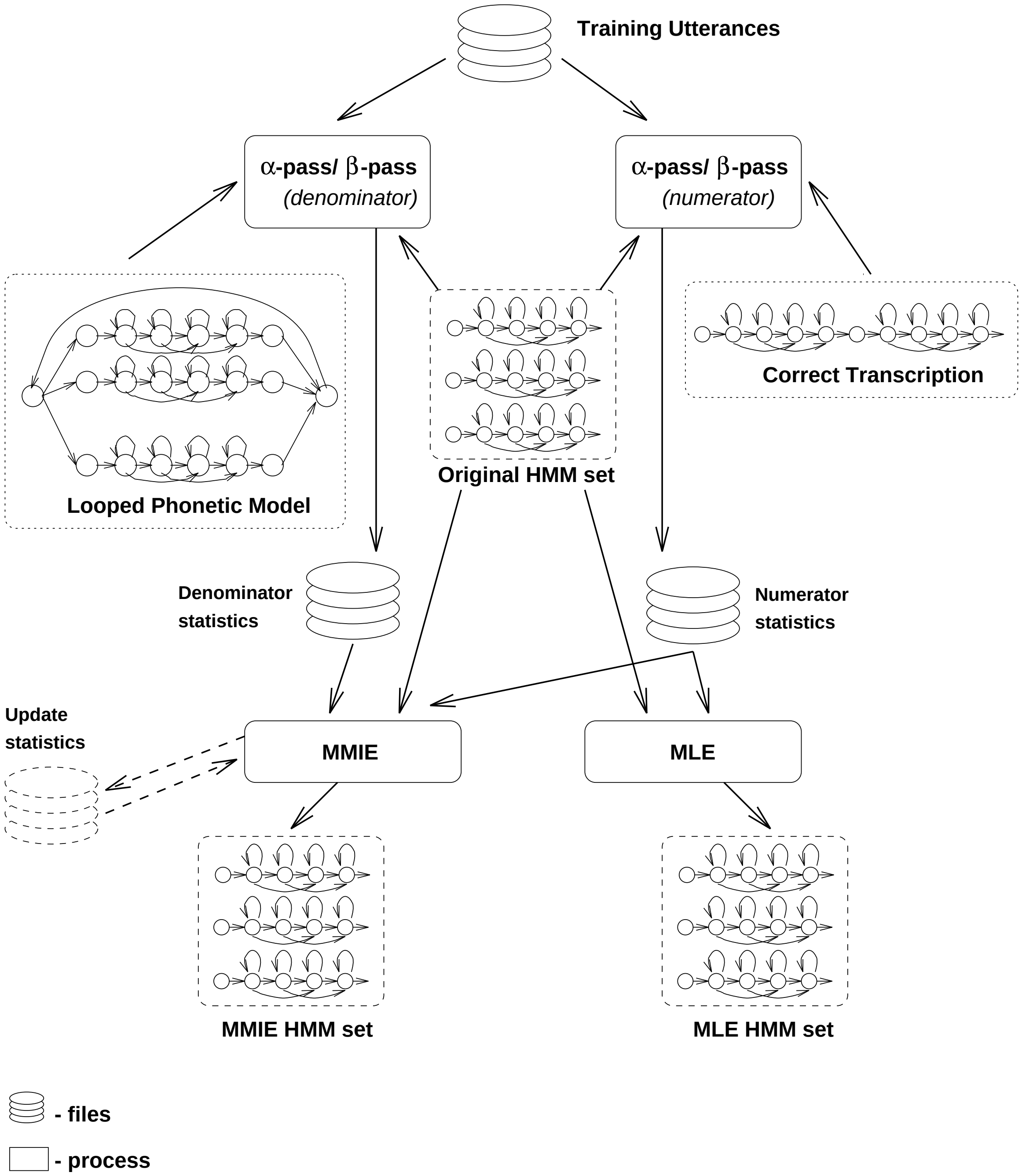

Figure 4.3: Estimation of HMM parameters using MLE and MMIE - application to continuous phone recognition

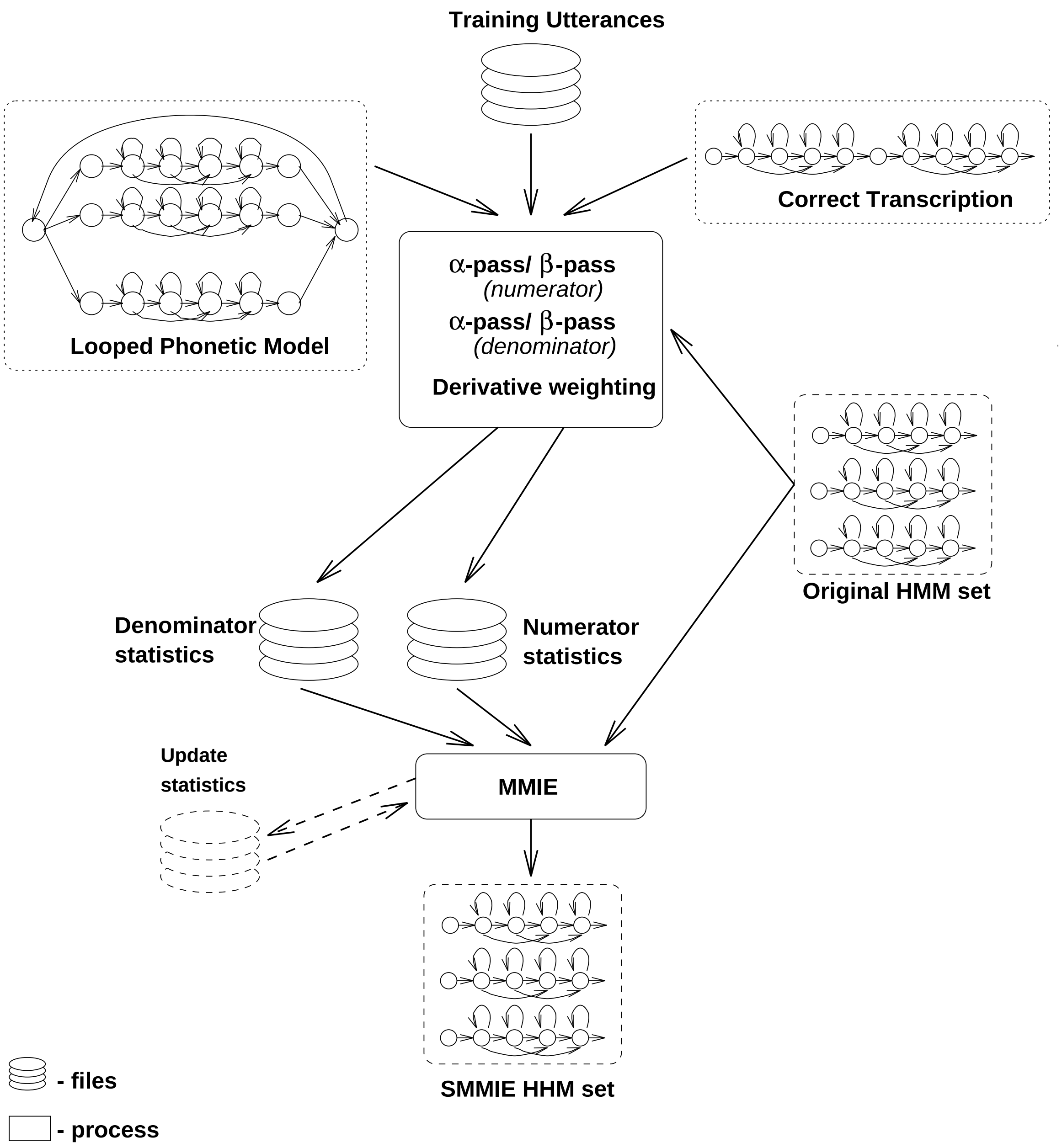

Figure 4.4: SMMIE optimisation of HMM parameters - application to continuous phone recognition

Chapter 5

Evaluation of Discriminative Training

Chapter 3 described a variety of objective functions which can be used to optimise the parameters of an HMM-based speech recognition system. Two techniques were considered in detail: Maximum Likelihood estimation (MLE) and Maximum Mutual Information estimation (MMIE). MLE is the most common approach to HMM parameter estimation. In this approach, each model is trained using data from the class to which the model corresponds. On the other hand, MMIE is discriminative in nature and as such it has many theoretical advantages over MLE. However, MMIE training involves a number of practical difficulties. For example, with a large number of classes, the algorithm requires orders of magnitude more computation than the corresponding MLE case. Furthermore, due to the lack of theoretical guidance, past research on MMIE has tended to use somewhat slow and unreliable steepest descent methods. With this in mind, in this chapter we present an empirical comparison between MLE, MMIE and the proposed non-linear version of MMIE (SMMIE) on a variety of speech recognition tasks. In particular, experimental results are presented on the BTL E-set database, the OGI ISOLET database, and the well known TIMIT continuous speech database. The first section of this chapter, also includes an empirical study of four local optimisation techniques which can be used to optimise the HMM parameters according to a discriminative objective function.

5.1 The choice of training algorithm

As discussed in chapter 3 the successful application of any discriminative training scheme depends to a large extent on the availability of an effective optimisation algorithm. Ideally the algorithm should result in a reasonable improvement in the value of the discriminative objective function after only a small number of iterations. The Baum-Welch algorithm used for MLE training is theoretically proven to converge. At the same time, a good value of the likelihood function is usually obtained after a small number of iterations. In the case of MMIE, steepest gradient descent and the extended Baum-Welch algorithm [46] are the only training procedures which are guaranteed to converge. However, as discussed in the

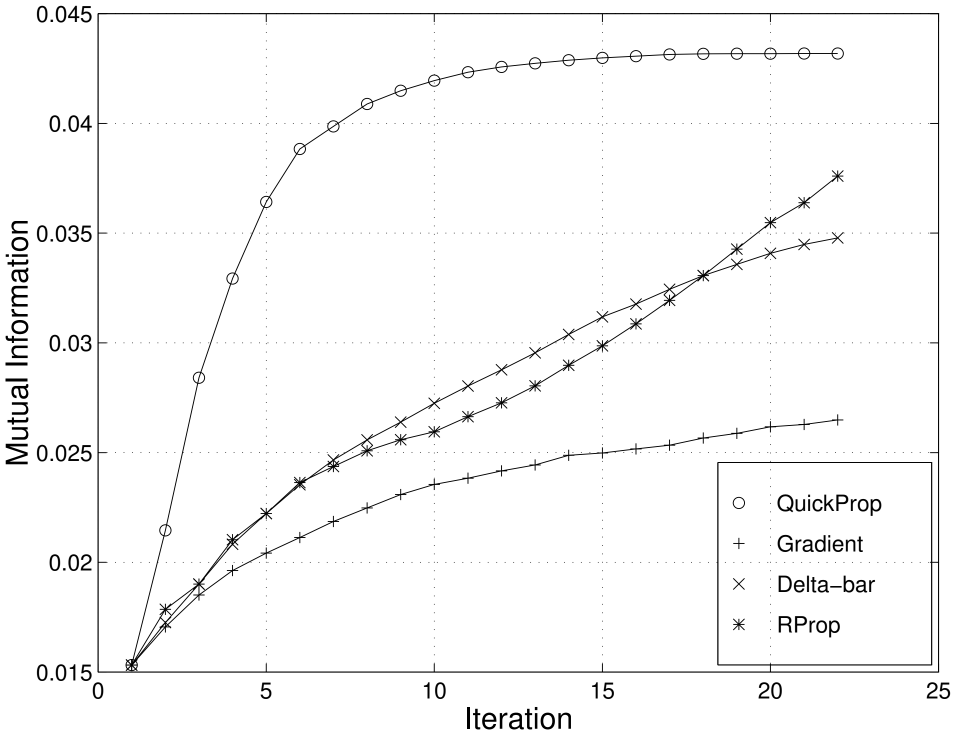

Figure 5.1: Comparison of optimisation algorithms.

previous chapter, the surface of a function of several thousands variables is likely to be rather complex in which case the steepest gradient descent algorithm will require a large number of iterations to find an optimal solution. Similarly, the extended Baum-Welch algorithm is only guaranteed to converge for a sufficiently large value of the constant term used in the re-estimation formulae, in which case, the speed of convergence makes the algorithm much less useful in practice. In the case of any other optimisation scheme, a theoretical proof of convergence is not available. Hence, the only way to judge the effectiveness of such an algorithm is to demonstrate its utility in practice.

In this section we shall present an empirical comparison between four local optimisation schemes: a modified version of the *Gradient Descent* algorithm, the *Delta-bar-Delta* adaptation rule, the *RProp* algorithm and the approximate second-order algorithm, which, we shall refer to as *QuickProp* following its similarities to the optimisation technique proposed by Fahlman in [40]. The task chosen to evaluate the algorithms was the speaker independent recognition of the members of the British English E-set {"B", "C", "D", "E", "G", "P", "T" & "V"}. For these experiments we used the single mixture diagonal covariance model set described in the following section. The last nine states of each HMM were tied across all models to provide common modelling of the vowel sound. It is important to realise that in this comparison we are not interested in the recognition performance of the system, we shall merely compare the rate of change in the value of the MMIE objective function for each one of the selected four algorithms.

Starting from the MLE trained models, the evaluation of each optimisation algorithm consisted of optimising the models' parameters for 22 iterations of discriminative training according to the MMIE objective function. In more detail, the four algorithm evaluated were

1. *Modified Gradient Descent.* A preliminary experiment using straight-forward gradient descent exhibited very slow convergence. This, as explained earlier, was attributed to the large dynamic range of the derivative values. The basic algorithm was then modified to use a separate learning rate for each HMM parameter type e.g. a distinct learning rate was used for all transitions, all means, all variances/covariances and all mixture component weights. The momentum term for all parameters was set $\zeta = 0.3$. The initial step size was set automatically according to

$$\eta(x) = 0.05 \max_{\forall x} \frac{|x|}{|x'|} \qquad \text{where} \qquad x' = \frac{\partial}{\partial x} f(\boldsymbol{x}) \tag{5.1}$$

   where the maximum was taken over all HMM parameters of that type.

2. *Delta-bar-Delta* (section 4.4.5.1). The algorithm was evaluated using $\phi = 0.5$ and $\zeta = 0.3$. However, difficulties were encountered in setting the value of the learning rate increment κ. This was due to the fact that an appropriate increment for the learning rate of one parameter wasn't necessarily useful for another parameter. The algorithm was modified, so that individual learning rates were incremented exponentially e.g. multiplied by $\kappa = 1.6$, if two consecutive derivatives possess the same sign. The initial learning rate for each parameter was set using expression 5.1.

3. *RProp algorithm* (section 4.4.5.2). The algorithm was modified to incorporate separate η_{min}, η_{max} for each parameter type. The corresponding values were set for each iteration according to

$$\eta_{max}(x) = 0.10 \max_{\forall x} |x| \qquad \text{and} \qquad \eta_{min}(x) = 0.001 \max_{\forall x} |x|$$

   The initial learning rates for all parameters were set separately for each parameter type using

$$\eta(x) = 0.05 \max_{\forall x} |x|$$

4. *QuickProp* (section 4.4.5.3). The algorithm was evaluated using $\varphi = 1.75$ as suggested by Fahlman. The individual learning rates were bootstrapped using expression 5.1.

The convergence plots of the four different algorithms are shown in figure 5.1. *QuickProp* appears to be the most effective algorithm. After only 10 iterations the value of the MMIE objective function is very close to its theoretical maximum 0.0432. The modified gradient descent algorithm provides steady but slow convergence. This is attributed to the common learning rate used within each parameter type and the lack of learning rate adaptation.

The *Delta-bar-Delta* learning rule exhibits faster convergence properties than the modified steepest descent algorithm, however the changes in the value of the objective function are marginal when compared to the *QuickProp* algorithm. The algorithm's behaviour was found to depend on choosing an appropriate value for κ. Too large an increment resulted in oscillations whilst slow convergence was observed for a small value of κ. The performance of the *RProp* algorithm is similar to the performance of *Delta-bar-Delta* algorithm. The algorithm performs marginally worse in the early stages of the training and marginally better during the last 4 iterations.

Overall, it seems that successful optimisation of the MMIE objective function requires rapid adaptation of the value of each individual learning rate parameter. The plots presented in figure 5.1 show the "typical" behaviour of each algorithm. Although it was possible to obtain faster convergence by tweaking the parameters of each adaptation rule, no drastic alteration in overall convergence was observed. In general, faster convergence in the early stages of the training was followed by erratic behaviour in later iterations and complete failure to converge. However, this is not to say that *QuickProp* is in general a "good" optimisation scheme and the remaining three are useless. It is strongly suspected that these local optimisation methods are to a large extent task-specific.

The suitability of any algorithm to a particular task depends on how well the adaptation rules captures the properties of the curvature of the objective function. However, we have shown that *QuickProp* is the most effective algorithm when applied to the re-estimation of HMM parameters according to the MMIE objective function. Another advantage of this algorithm is that it is simple to implement and it requires one to set a single variable parameter φ. The convergence plots of the *QuickProp* algorithm for different values of φ are shown in figure 5.2. Larger values of φ provide faster convergence, however when $\varphi = 3.0$ the algorithm exhibits chaotic behaviour and fails to converge. The corresponding plot is prematurely terminated since the updated HMM set after the eighth iteration caused the pruning mechanism in the following alignment to fail. In all experiments with MMIE training in this thesis, the QuickProp algorithm will be used exclusively to update the parameters of the HMMs. Finally, the *QuickProp* algorithm was devised as a neural network optimisation techniques. As such, it represents the ideal tool to adapt the parameters of the adaptive input transformations which will be introduced in chapter 8.

5.2 Experiments on the BTL E-set database

The first task chosen to evaluate the performance of the proposed discriminative training methods was the speaker independent recognition of the members of the British English E-set. A description of the database is given in appendix A.

5.2.1 Baseline results and parameter tying

Each member of the E-set was modelled by a left-to-right HMM with 15 emitting states and no skip transitions. A diagonal or full covariance mixture output distribution was associated with each emitting state. In each case, the parameters of the models were estimated from

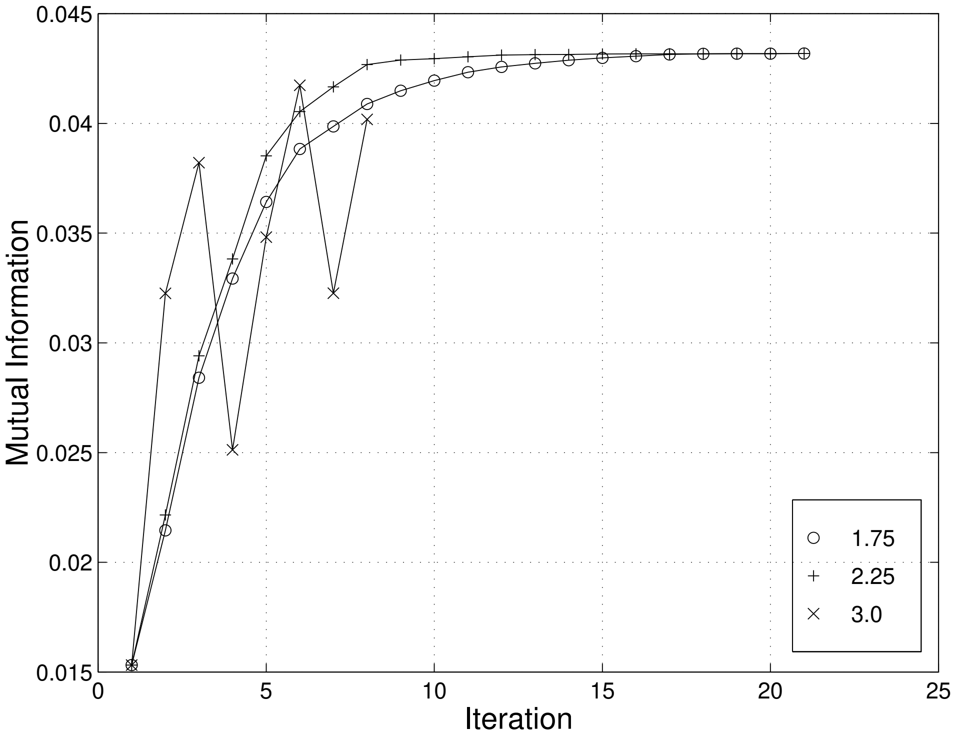

Figure 5.2: Convergence of the *QuickProp* algorithm for different values of φ.

the full training set (1239 utterances) according to the MLE criterion using 8 iterations of the Baum-Welch algorithm. As discussed in chapter 2, the sharing of parameters in an HMM system is a powerful mechanism for adjusting the structure of the models in order to obtain reliable parameter estimates from limited amounts of training data. A preliminary set of experiments was carried out to investigate the sharing of parameters in the output distributions of the models. The results from these experiments are presented in table 5.1. The first column indicates the number of mixture components and the type of covariance matrix used. For the latter, `Diag` denotes diagonal covariance; `Full` denotes full covariance; `GCov` denotes a single covariance matrix shared by all distributions in all models (Grand Covariance); and `GMCov` denotes a single covariance matrix for all distributions in each individual model (Grand Model Covariance). The second column in the table indicates which HMM states share the same mixture distribution across all models. A clear improvement in performance is visible when each of the final 9 states shares the same distribution across all models. This is due to the fact that virtually all of the information necessary to discriminate between the different classes is concentrated in the consonant part of each utterance[1]. Separate modelling of the vowel sound in each class increases the

[1] It is assumed that the first 6 states of each word model implicitly model the consonant part of the utterance, whilst the final 9 states model the vowel.

Type	States shared	log(P)	%Acc train	%Acc test	Param.
`1/Diag`	no tying	-29.645	90.88	81.21	5760
`1/Diag`	7-15	-30.385	91.04	84.50	2736
`1/Full`	7-15	-27.681	98.87	92.29	19152
`1/GCov`	7-15	-32.685	88.70	84.09	1704
`1/GMCov`	7-15	-29.836	95.00	90.65	4056

Table 5.1: The effect of sharing HMM parameters, MLE-trained models.

Type	log(P)	%Acc train	%Acc test	Param.
`1/Diag`	-30.385	91.04	84.50	2736
`1/Full`	-27.681	98.87	92.29	19152
`3/Diag`	-28.660	95.24	88.76	8379
`4/Diag`	-28.354	96.85	90.65	11172
`5/Diag`	-27.892	97.43	90.40	13965
`7/Diag`	-27.263	98.22	90.32	19551

Table 5.2: Baseline MLE results on the BTL E-set database.

chance of a mismatch occurring in the final stages of the Viterbi search. This observation also indicates that the available training data is not enough to obtain reliable estimates of the distribution parameters for the vowel in each individual class.

For all subsequent experiments on this database, the sharing of the final 9 states was always enforced. The results in table 5.1 also reveal that full covariance output distributions provide more accurate modelling than distributions with diagonal covariance matrices. This is indicated by the higher log-likelihood per frame as calculated on the training set and the actual improvement in recognition performance on the test set. Sharing a full covariance matrix across all the states of a single model provides a very compact but effective system. Table 5.2 shows the baseline Maximum Likelihood results for different number of mixture components. The full covariance system still provides the best performance although in terms of number of parameters, it is roughly equivalent to the 7 mixture component diagonal covariance system which is showing signs of undertraining.

5.2.2 Discriminative training results

In these experiments, discriminative training according to the MMIE and the SMMIE objective functions was performed on the MLE-trained model sets described in the previous section. Adaptation of the HMM parameters was carried out for 12 iterations using the *QuickProp* algorithm. The recognition results are given in table 5.3. In all experiments, the MMIE-trained models achieved 100% recognition accuracy on the training set which suggests the need for larger training databases. The full covariance model set provides the best performance in this case with an overall recognition accuracy of 94.01%. For comparison

Type	MMIE		SMMIE	
	%Acc train	%Acc test	%Acc train	%Acc test
`1/Diag`	100.00	91.05	98.22	92.21
`1/Full`	100.00	93.85	98.87	94.59
`3/Diag`	100.00	91.39	99.03	92.04
`4/Diag`	100.00	91.55	99.52	91.87

Table 5.3: MMIE and SMMIE results on the BTL E-set database.

Woodland [109] reported 95.5%/92.1% on the same training and test sets respectively using discriminative state specific input transformations. For the diagonal covariance single mixture distribution models, MMIE has managed to reduce the error rate by 42%. However, as the number of mixture components is increased the relative improvement in performance provided by MMIE decreases. The last two columns in table 5.3 show the recognition performance of the same model sets optimised according to the SMMIE objective function. For these experiments the parameters of the sigmoid function (equation 3.21) were set to $\nu = 0.001$, $\xi = 1.6094$. The latter corresponds to the hypothetical decision boundary for 8 classes and uniform language model, calculated as described in section 3.6. The result of 92.21% accuracy is very close to the original MLE trained full covariance model set which has 7 times as many parameters as the single mixture diagonal covariance model set. In general, parameter adaptation according to the SMMIE objective function has provided a consistent improvement in performance over the corresponding MMIE trained models. However, these results were found to be rather sensitive to the appropriate selection of the ν parameter which controls the slope of the non-linear weighting function. Too small a value of ν makes SMMIE more similar to MMIE, whilst a larger value of ν discards a significant number of training utterances which, in turn, does not alter the recognition performance. Finally, the result of 94.59% accuracy is very good by itself, however, it still falls short of the best accuracy figure of 96.0% published by Ayer [4].

5.3 Evaluation on the ISOLET database

The ISOLET database is an isolated speech, English alphabet database. Its full description is given in appendix A. The task was seen as a natural progression from the BTL E-set database for evaluating the proposed discriminative training methods. In terms of number of utterances recorded per class the ISOLET database is no larger than the BTL E-set database. However, the different partitioning of the data (80% training data, 20% test data), the larger speaker population and the overall appeal as a realistic speech recognition application makes this task a more representative speech recognition problem.

Type	MFCC_E	MFCC_E_D	MFCC_E_D_A
1/Diag	86.47	91.67	92.76
4/Diag	88.91	95.00	95.13
8/Diag	89.55	95.45	95.26
12/Diag	90.19	95.83	95.71
16/Diag	90.38	95.51	95.96

Table 5.4: MLE results on the ISOLET database using different feature vectors.

Type	MLE	MMIE	SMMIE	Param.
1/Diag	91.67	93.33	93.26	16536
1/Full	96.03	96.60	96.47	124020
2/Diag	93.91	94.74	94.67	33708
2/Full	96.09	96.34	96.41	248676
4/Diag	95.00	95.70	95.77	67416
8/Diag	95.45	96.09	96.21	134832
12/Diag	95.83	96.22	96.28	202248
16/Diag	95.51	95.77	95.90	269664

Table 5.5: MLE and discriminative training results on the ISOLET database.

5.3.1 Baseline results

Each letter in the alphabet was modelled by a word-level left-to-right HMM with 15 emitting states and no skip transitions. As before, the last nine states of models corresponding to members of the American English E-set (9 classes) were tied together to provide common modelling of the vowel sound. Tying other states according to the alphabet pronunciations (table A.1) produced no improvements in performance. There was a total of 318 distinct states in the system with diagonal or full covariance mixture output distribution. Four iterations of MLE training were used to re-estimate the parameters of the baseline model sets. Table 5.4 shows comparative results from using three different feature sets. It is clear from these results that the MFCC_E_D feature set provides the optimal balance between performance and number of parameters used. The use of second differential parameters yields some improvement in performance, however, the relative gain is reduced as the number of mixture components is increased. The full set of MLE results for diagonal and full covariance model sets is given in the second column of table 5.5. The single mixture full covariance system provides the best performance of 96.03% which is comparable with the best result to date of 96.00% published in [27]. The confusion matrix from the recognition experiment using this model set is given in table 5.6. Analysis of the distribution of recognition errors shows that 65% of the errors take place within the E-set and another 19% between the letters {M,N}.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	% Acc
A	59	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	98.33
B	0	51	0	1	2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	5	0	0	0	0	85.00
C	0	0	59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	98.33
D	0	0	0	49	0	0	3	0	0	0	0	0	0	0	0	1	0	0	0	7	0	0	0	0	0	0	81.67
E	0	1	0	1	57	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	95.00
F	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00
G	0	0	1	0	0	0	57	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	95.00
H	0	0	0	1	0	0	0	59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	98.33
I	0	0	0	0	0	0	0	0	59	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	98.33
J	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00
K	4	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	93.33
L	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00
M	0	0	0	0	0	0	0	0	0	0	0	0	55	5	0	0	0	0	0	0	0	0	0	0	0	0	91.67
N	0	0	0	0	0	0	0	0	0	0	0	0	7	53	0	0	0	0	0	0	0	0	0	0	0	0	88.33
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	100.00
P	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	59	0	0	0	0	0	0	0	0	0	0	98.33
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	100.00
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	100.00
S	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	58	0	0	0	0	0	0	0	96.67
T	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	2	0	0	0	55	0	0	0	0	0	0	91.67
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	100.00
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	57	0	0	0	2	95.00
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	100.00
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	100.00
Y	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	59	0	98.33
Z	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56	93.33

Table 5.6: Confusion matrix for MLE-trained single mixture full-covariance models (ISOLET).

5.3.2 Discriminative training results

In these experiments, the parameters of the MLE trained model sets were optimised using the MMIE and SMMIE algorithms. The *QuickProp* algorithm was used to adapt the HMM parameters for 12 iterations. The results from these experiments are presented in table 5.5. To our knowledge, the performance figure of 96.60% accuracy achieved using single mixture full covariance HMMs represents the best result published on this database. The confusion matrix from this experiment is given in table 5.7. The MMIE training has resulted in 5% fewer errors within the E-set and 25% fewer errors in the {M,N} set. As before, in all cases the MMIE-trained models achievd 100.00% recognition accuracy on the training set.

For the SMMIE experiments, the parameters of the non-linear weighting function were set to $\nu = 0.008$, $\xi = 2.6390$. The latter corresponds to the hypothesised decision boundary for 26 classes using a uniform language model. The application of the algorithm did not resulted in any substantial improvement in performance. In fact, in certain cases the application of the algorithm actually caused minor degradations in performance. As mentioned earlier, the behaviour of the algorithm is highly sensitive to the appropriate selection of the ν and ξ parameters. It is possible that $\nu = 0.008$ is not the right choice, however, in the preliminary trials used to determine this value, no substantial change in performance was noticed. Another explanation for this behaviour came from the histogram distribution of

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	% Acc
A	59	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	98.33
B	0	52	0	1	2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	4	0	0	0	0	86.67
C	0	0	59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	98.33
D	0	0	0	50	0	0	3	0	0	0	0	0	0	0	0	1	0	0	0	6	0	0	0	0	0	0	83.33
E	0	1	0	1	56	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	93.33
F	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00
G	0	0	1	0	0	0	57	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	95.00
H	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00
I	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00
J	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00
K	4	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	93.33
L	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00
M	0	0	0	0	0	0	0	0	0	0	0	0	56	4	0	0	0	0	0	0	0	0	0	0	0	0	93.33
N	0	0	0	0	0	0	0	0	0	0	0	0	5	55	0	0	0	0	0	0	0	0	0	0	0	0	91.67
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	100.00
P	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	58	0	0	0	1	0	0	0	0	0	0	96.66
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	100.00
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	100.00
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	100.00
T	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	2	0	0	0	56	0	0	0	0	0	0	93.33
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	100.00
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	57	0	0	0	2	95.00
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	100.00
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	100.00
Y	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	59	0	98.33
Z	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	57	95.00

Table 5.7: Confusion matrix for MMIE-trained single mixture full-covariance models (ISOLET).

mutual information measure of the utterances in the training set. This was found to be fairly uniform which suggests that there is a good match between the models and the data. The latter observation can be attributed to the larger training set and the stringent quality filtering carried out whilst collecting the database.

In general, the application of discriminative training has been more effective in improving the recognition performance of the "compact" model sets, e.g. the ones that use fewer mixture components.

5.3.3 Computational considerations

The quantities needed to optimise the MMIE objective function are calculated using the Forward-Backward (FB) algorithm. In MLE training the procedure is performed once using the correct transcription of the utterance. In the case of MMIE the algorithm is performed twice, once using the correct transcription of the utterance and the second time using the recognition model. In E-set recognition, the recognition model consisted of the 8 class models placed in parallel. In the case of alphabet recognition we have to consider 26 models at the same time which is more than three times as many models than in the previous E-set experiments. As mentioned in the previous chapter, the computational complexity of the Forward-Backward algorithm can be reduced by using separate pruning thresholds for

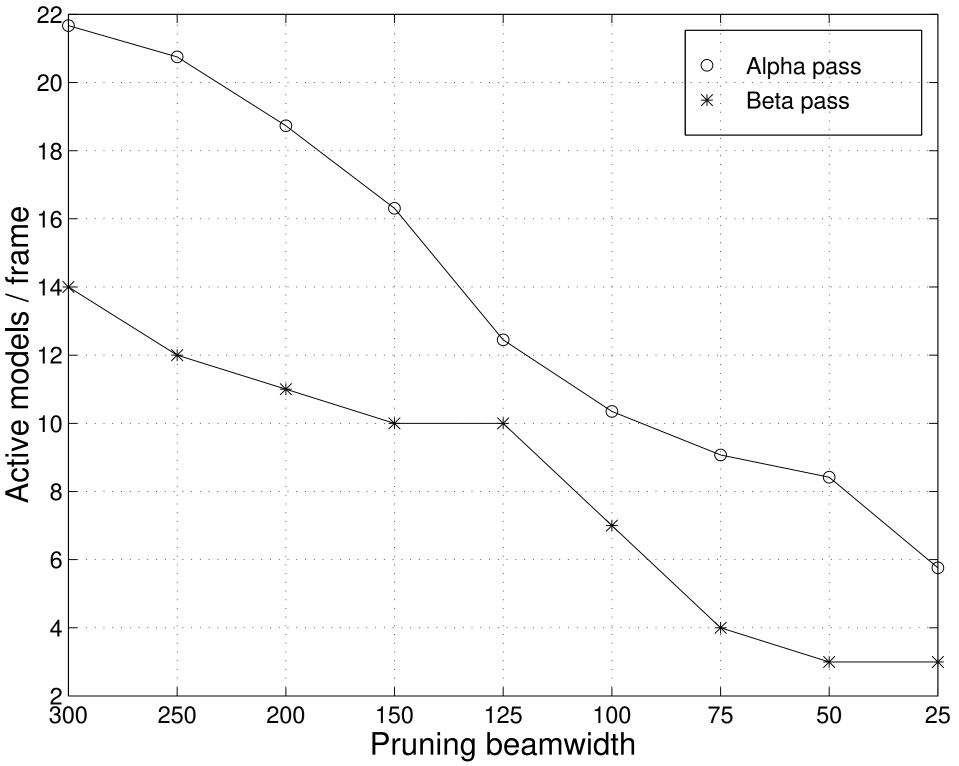

Figure 5.3: The effect of forward-backward pruning (ISOLET).

the forward and the backward passes. The choice of pruning thresholds is task specific and reflects on how well the acoustic models fit the training data. A number of experiments have been performed in order to determine the optimal values of the pruning thresholds[2]. Figure 5.3 shows the average number of active models per frame for each pass in the FB algorithm when applied to the recognition model[3]. In general smaller threshold will reduce the number of active models, however, this may introduce search errors and in the extreme case the FB algorithm may even fail to find a path. For all discriminative training experiments presented on this database, the forward pruning threshold (α-pass) was set to 200 and the backward pruning threshold (β-pass) was set to 75.

[2] Similar experiments were carried out on the BTL E-set database, however, due to the similarities between the two tasks, here we consider only the database with more classes.

[3] The application of the FB algorithm to the correct model is not considered since the word-level HMM remains active throughout the alignment. This comes from the fact the pruning mechanism operates at the HMM level rather than at state level. Having a separate vowel HMM in the case of E-set recognition as opposed to tying states across models is a partial solution to the problem. However, the computational complexity of the alignment when using the correct model is negligible.

Type	MFCC_E		MFCC_E_D		MFCC_E_D_A	
	%Corr	%Acc	%Corr	%Acc	%Corr	%Acc
1/Diag	53.01	48.87	59.78	54.79	62.94	55.58
4/Diag	57.09	53.25	66.29	61.95	68.29	61.59
8/Diag	58.43	54.80	68.33	64.26	70.55	64.38
10/Diag	58.71	55.26	68.83	64.64	71.25	65.20
24/Diag	59.93	56.55	70.59	66.51	73.71	68.69

Table 5.8: Comparison of using different parameter sets (TIMIT).

Type	%Corr	%Sub	%Del	%Ins	%Acc	Param.
1/Diag	62.94	26.60	10.46	7.36	55.58	11232
1/Full	67.79	20.57	11.64	4.46	63.33	117936
2/Diag	65.27	25.43	9.30	7.12	58.14	22752
2/Full	69.42	19.65	10.92	4.14	65.28	236160
4/Diag	68.29	23.47	8.25	6.69	61.59	45504
8/Diag	70.55	21.90	7.55	6.17	64.38	91008
12/Diag	71.93	20.87	7.19	6.00	65.93	136512
16/Diag	72.52	20.40	7.08	5.29	67.22	182016
20/Diag	72.86	20.19	6.94	5.07	67.79	227520
24/Diag	73.71	19.63	6.67	5.02	68.69	273024

Table 5.9: Baseline MLE results using diagonal and full covariance output distributions.

5.4 Evaluation on the TIMIT database

This section describes the continuous phone recognition experiments performed on the TIMIT database. A detailed description of the database and the general experimental setup is given in appendix A. Each of the 48 phone classes in the system was modelled by a left-to-right HMM with 3 emitting states and no skip transitions. No tying was applied to any of the HMM parameters. The full training set of 3696 utterances was used for both MLE and MMIE training. All tests were performed on the core test set of 192 sentences. Results are presented in terms of percentage correctly recognised phones and overall recognition accuracy. For more details on the scoring procedure see section A.4. Unless stated otherwise, a phone-level bigram language model with a scale factor of (2.0) was used in the recognition experiments.

5.4.1 Baseline results

Four iterations of MLE training were used to re-estimate the parameters of the baseline model sets. Table 5.8 presents comparative results from models using different feature sets. Contrary to the results from a similar experiment on the ISOLET database, the perfor-

Type	%Corr	%Sub	%Del	%Ins	%Acc
1/Diag	67.79	22.40	9.81	6.79	61.00
4/Diag	71.61	19.04	9.34	6.39	65.23
8/Diag	73.31	17.95	8.75	5.67	67.64
12/Diag	74.25	17.59	8.16	5.65	68.59
16/Diag	74.80	17.17	8.02	5.05	69.76
20/Diag	75.08	16.80	8.12	5.03	70.05
24/Diag	75.20	16.59	8.21	4.77	70.44

Table 5.10: TIMIT results using MMIE training and diagonal output distributions.

mance figures indicate a clear gain in recognition accuracy when using the 39 dimensional feature set comprising of 13 MFCCs, energy component and their respective first and second differentials. The `MFCC_E_D_A` feature set was used exclusively in all subsequent experiments. The performance of the MLE trained model sets is given in table 5.9. The best performance figures of 73.71% correct/68.69% accuracy are very good for a baseline system using context independent HMMs. A somewhat surprising observation is that the two full covariance distribution systems did not provide the expected gain in performance as suggested by the two previous results on the isolated tasks. For example, the 8 mixture component diagonal covariance system has 23% fewer parameters and yields higher accuracy than the single mixture component full covariance system.

5.4.2 MMIE experiments

Similarly to the previous discriminative training experiments, the baseline MLE-trained model sets were optimised according to the MMIE objective function using 18 iterations of the *QuickProp* algorithm. Due to the somewhat disappointing performance of the full covariance model sets and the extra computational overhead involved in their use, the MMIE algorithm was only applied to diagonal covariance mixture distribution model sets. The results from these experiments are presented in table 5.10. MMIE has provided a consistent improvement in accuracy over the corresponding MLE results. The MLE and MMIE performance figures are graphically compared in figure 5.4. In particular, MMIE has provided 12.2% reduction in error rate for the single mixture component system and 5.6% reduction in the 24 mixture component system. In the latter case, the improvement in overall recognition accuracy comes from a 15% reduction in substitution errors. Unfortunately, in certain cases, reduction in the number of substitution errors made, is offset by an increase in the number of deletions.

In a set of preliminary experiments, the application of SMMIE to continuous speech recognition was found to be rather less effective than the corresponding isolated cases. This is intuitively expected, since the SMMIE objective function gives a different weight to each utterance according to the mutual information measure and the value of this measure has a relatively uniform distribution over the training utterances. Indeed, a small number of

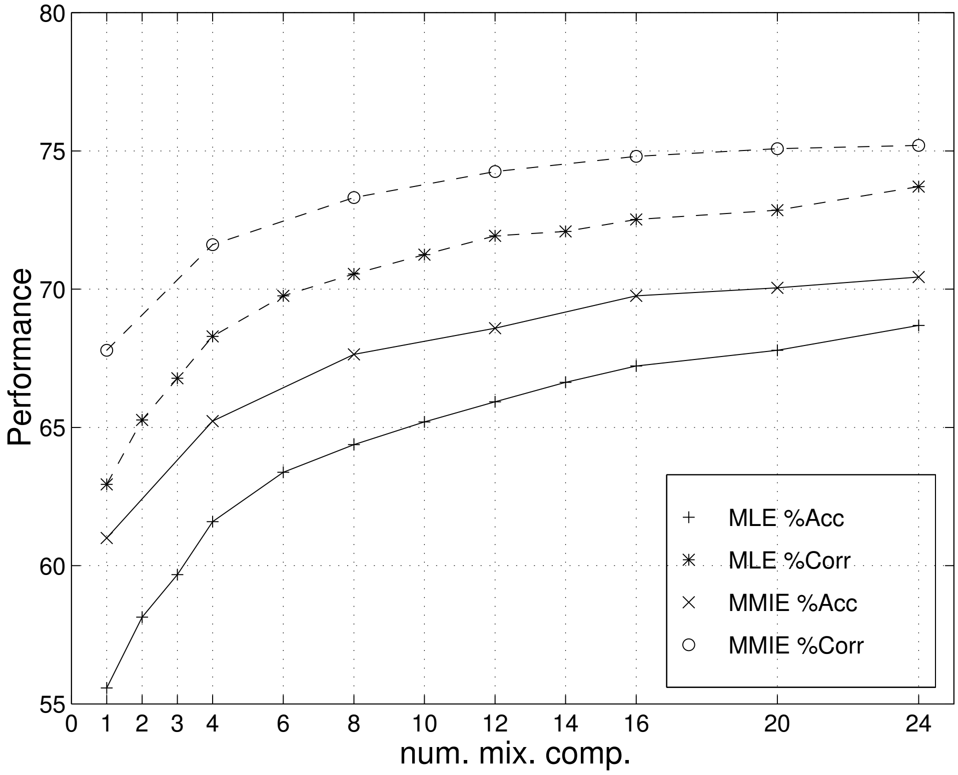

Figure 5.4: Performance of MLE and MMIE trained models (TIMIT).

training utterances were found to have significantly inferior mutual information than the sample average. However, their removal from the training process did not alter the recognition performance. This observation follows from the fact that even the most complex system uses a mere 273024 parameters which are estimated from a total of 1124823 observation vectors. Finally, finding the hypothesised decision boundary is also rather involved due to the language model scores and the vast number of distinct paths in the model. An alternative solution is to manually derive this value from the histogram of mutual information measure on the training set.

5.4.3 Computational considerations

Table 5.5 shows the effectiveness of the forward-backward pruning for both the recognition model and the correct transcription. For the correct transcription, the reduction in computation is significant since even with only 3 active models on the forward pass, the pruning does not alter the overall likelihood of the utterance. In the case of the correct transcription, the overall reduction in complexity is typically 16 times on the forward pass and 20-24 times on the backward pass. However, the forward-backward pruning is rather

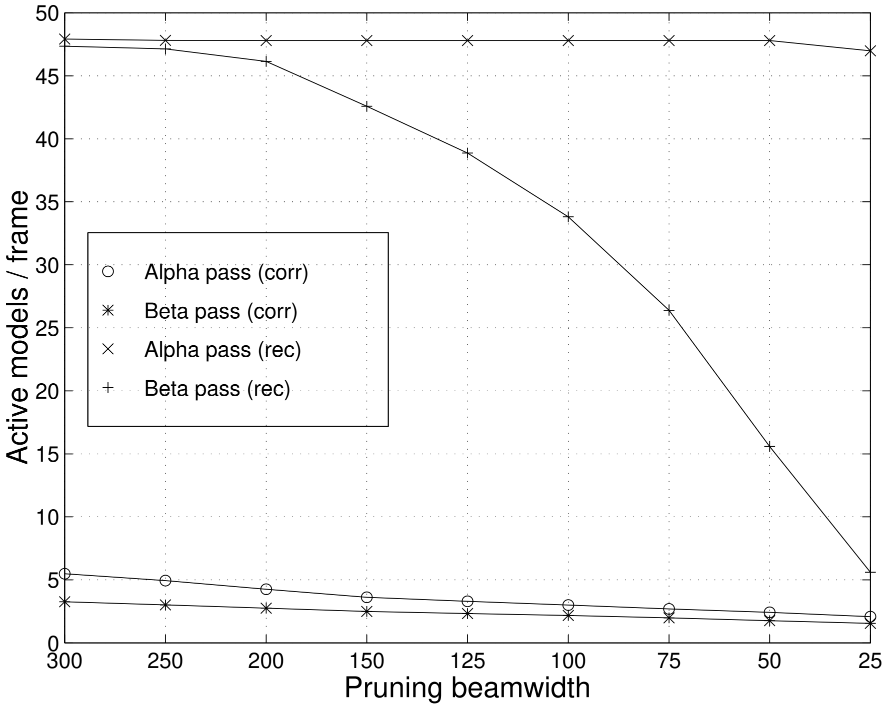

Figure 5.5: The effect of forward-backward pruning (TIMIT).

less effective when applied to the looped phonetic model. Virtually all models (48) remain active on the forward pass, no matter how narrow the pruning beam is. This is attributed to the short average duration of each phone model, e.g. only 3 observations are needed for a token[4] to traverse each phone model. On average, 6-10 models are required to be active on the backward pass. In general, with more detailed acoustic models forward-backward pruning is more effective, however, the MMIE training algorithm still requires 12-15 times more computation than standard MLE training.

5.4.4 The use of long-span language models

In this experiment, the standard bigram language model was replaced by a fourgram back-off language model. The language model was constructed using a recently developed extension of HTK. The language model training data came from the transcriptions of the full TIMIT training set (3696 sentences). The language model was generated using the absolute discounting method [82] where each n-gram ($n = 2, 3, 4$) contributed an equal amount to the total unseen "mass" used in calculating the probabilities of unseen events. Trigrams which occur less than 3 times and fourgrams which occur less than 10 times were considered to introduce unwanted bias towards the grammar of the training utterances and were discarded.

[4]The Forward-Backward algorithm is implemented using the token-passing paradigm [115].

Component	Entries	Cut-off	core (192) 7215 tokens	full (1344) 50754 tokens
bigram	1692	0	16.35	16.33
trigram	8055	2	14.71	14.70
fourgram	2470	9	14.51	14.58

Table 5.11: Perplexity tests on the core and full TIMIT test sets.

Component	requested	found	approx	backed
bigram	1121	99.1%	0.9%	0.0%
trigram	5203	78.5%	20.0%	1.5%
fourgram	7023	25.9%	20.3%	53.8%

Table 5.12: Fourgram access statistics on the TIMIT core test set.

The language model parameters and the results from perplexity tests are given in table 5.11. The trigram language model provides 10% reduction in perplexity whilst the fourgram model provides a further reduction of 1%. Table 5.12 shows the statistics about the language model access during recognition. Only 25% of the requested fourgrams are found in the model, the rest are either computed using the corresponding trigram and the fourgram's back-off weight or fully backed-off to the corresponding trigram. The improvement provided by the trigram language model is due to the high proportion (78%) of requested trigrams explicitly contained in the model. Recognition experiments with the fourgram language model were performed using the single-pass decoding algorithm described in [86, 106]. The decoder incorporates a dynamically grown non re-entrant recognition network which is built on-the-fly. A new copy of the full phone network is grown as a successor of each active phone model. Thus, different recognition paths are maintained separately which allows for the relatively easy incorporation of language model scores.

The fourgram language model was used in conjunction with the best performing set of models. The results are given in table 5.13. The use of fourgram language model has resulted in further reduction in error rates of 5.24%/3.82% for the MLE and MMIE trained models respectively. To our knowledge, the performance figures of 71.57%/75.51% accuracy/correct represent the best result obtained on the TIMIT database using context independent HMMs. In fact, these figures come close to the recently published results

Type	%Corr	%Sub	%Del	%Ins	%Acc
MLE, `24/Diag`	74.16	18.13	7.71	3.84	70.33
MMIE, `24/Diag`	75.51	16.87	7.62	3.94	71.57

Table 5.13: TIMIT experiments with a fourgram language model.

by Young and Woodland [116] of 72.3%/76.7% using 1176 state clustered right context dependent bi-phone HMMs and a bigram language model.

5.5 Summary

This chapter has presented an experimental evaluation of MLE and MMIE. In virtually all experiments, MMIE has provided a consistent improvement in performance over MLE. However, the different nature of the three different databases has resulted in different relative gains in performance. We have also made the following observations

1. On average, discriminative training provides larger improvements in performance for isolated word tasks than when applied to the continuous speech recognition task. This can be explained with the fact that a phone is a very abstract linguistic unit with a short average duration whose actual realisation depends on neighbouring context.

2. Parameter tying has provided a mechanism for sharing parameters between word-level HMMs with substantial gains in performance.

3. The MLE-trained, full covariance models have provided the best performance on the two isolated tasks, however, this is not the case for continuous speech where diagonal mixture distributions give superior performance. For the isolated tasks, the full covariance model sets were further trained using MMIE and achieved state of the art performance.

4. The use of second differentials in the feature set is vital to achieve good performance on the continuous phone recognition task, however, they are of a limited value when applied to the isolated alphabet database. Furthermore, in the former case, second differential parameters appeared to be more effective with models of higher complexity.

5. The use of a fourgram back-off language model was investigated in the context of continuous phone recognition and found to yield an improved recognition accuracy over the standard bigram language model for both MLE and MMIE trained models.

6. The successful application of discriminative training has been possible through the use of an efficient training algorithm. The *QuickProp* algorithm is an approximate second order optimisation strategy which is simple to implement and very effective when used to implement MMIE training.

7. Forward-Backward pruning is an important technique, which has also contributed to the successful application of MMIE. The method provides a mechanism for discarding unimportant models from the FB algorithm with substantial computational savings in both the isolated and continuous cases.

8. The SMMIE algorithm provides a flexible scheme for selecting representative training utterances. Unfortunately, its application is limited to the isolated tasks where decision boundaries are more clearly defined. In particular, with limited amounts of

training data the algorithm was found to offer consistent performance improvements over MMIE.

On the two isolated word recognition tasks, MMIE always achieved perfect recognition on the training set which leads one to discuss the generalisation abilities of the MMIE algorithm. Indeed, MMIE seems to be very good at memorising the training data, however, good recognition performance on the training data does not always translate into improved recognition performance on an independent test set. In MLE training, a mismatch between the performance on the training data and the performance on the test data is often attributed to the training data not being "representative" of the task. In the case of MMIE the differences in performance are more substantial and a possible explanation is that the algorithm attempts to model acoustic events which are peculiar to the utterances of the training set and have no global significance for the task. It is certainly true that certain parameters in the HMM framework are more globally significant for the task than others. For example, in the case of BTL E-set recognition the full covariance system used four times as many parameters and achieved better recognition results on the test set than the single mixture diagonal covariance system. At the same time, both systems achieved perfect recognition on the training set.

Finally, in all experiments carried out, MMIE training was shown to be more effective for the systems which use fewer mixture components. On the continuous speech recognition task, MMIE only provided a marginal performance improvement when 24 mixture component models were used. Indeed this observation is in accordance with the theory which shows that when the training sample is large, and as the model distributions get closer to the "true" distributions of the source, performance differences between MLE and MMIE will vanish.

Chapter 6

Input Transformations

A significant recent development in improving the performance of speech recognition systems has been the addition of new feature components to the observation vectors. However, because of the larger dimensionality of the observation vector the number of model parameters and the computational requirements have also increased. To improve recognition performance it is not feasible to indefinitely increase the size of the observation vectors, nor is it satisfactory or practical to run recognition experiments on manually chosen subsets of the feature space in order to decide on the best performing set with a reasonable number of parameters. In such cases, it is clearly desirable to understand which components of the feature vectors provide the greatest contribution to the recognition performance and to discard the least useful components. This chapter introduces the theory and application of dimensionality reduction methods which can be used to provided compact and informative feature vectors for an HMM-based speech recognition system.

6.1 Introduction

Until a few years ago each observation vector contained only spectral coefficients as estimated from the speech signal over a period ranging from 20 to 50 ms. However it has been shown [45] that improved recognition performance can be achieved by including the time derivatives of the spectral coefficients in the frame vectors. More recently, the further addition of the second-order time derivatives of the spectral coefficients has been shown to yield improved recognition performance for large vocabulary speech recognition [51]. The improved recognition performance obtained by using first and second differential parameters shows a limitation of the conventional HMM framework. Although the derivative information is directly available from the static parameters, the observation independence assumption prevents the Markov model from tracking the rate of change in the observation vector components because it considers each observation vector in isolation from its immediate predecessors and successors.

Nowadays, it is common to transform the spectral parameter representations into a "cepstrum-like" domain by applying the discrete cosine transform (DCT) to the output of the filter bank or the linear predictive coefficients (LPCs). First and second differential parameters are then computed from the basic cepstral coefficients. The (DCT) is a very important tool in the current speech recognition technology since the resulting cepstral features are largely uncorrelated which makes the diagonal covariance Gaussian distributions (section 2.2.2) somewhat more appropriate. The basic feature sets and their derivation as used in the experiments presented in this thesis are given in appendix A. The following sections will describe 1) feature selection algorithms which can be used to rank-order the components of the feature vector; and 2) feature extraction algorithms which transform the spectral/cepstral coefficients and provide a method of selecting a subset of parameters from the transformed domain.

6.2 Classes and distance measures

Speech recognition is about recognising time variant patterns which form higher level structures such as phones, words and sentences. Speech recognition using hidden Markov models makes the assumption that speech patterns are quasi-stationary and as such their time variability can be modelled as a sequence of states and associated probability distributions. Developing a speech recognition system involves training a set of whole or sub-word models and then testing unknown utterances against these models during recognition. Several of the feature selection and feature extraction techniques discussed in this chapter are based on the concept of "improving discrimination between classes". With this property, their application to static pattern recognition is a straight forward task. However, the success of these techniques to the speech recognition problem will depend to a large extent on choosing an appropriate class definition.

In the HMM framework each model consists of a sequence of states where each state can be thought of as modelling a particular class of speech sound. There is no explicit correspondence between HMM states and any particular sub-word unit. The set of utterance frames that map to a given state are assumed to belong to that class of speech sound and form a set of possible acoustic realisations for that class. Each observation vector can be regarded as a pattern in multi-dimensional space and the distribution of frames corresponding to a particular state is assumed to be a multivariate Gaussian mixture density. Taking a simplified view, the recognition of a single utterance is accomplished by matching the sequence of utterance frames against each model in turn and the model with the highest likelihood of producing the observation sequence is chosen. At the frame level, the recognition process becomes one of assessing the probability of a particular utterance frame belonging to a given state. Due to the inherent confusability in speech sounds, the distributions for different classes will overlap. This overlap is a potential source of recognition errors, however, there is no guarantee that improving class discrimination at the frame/distribution level will result in better overall recognition accuracy.

The frame log probability computed by a Gaussian output distribution is given by

$$\log b(\boldsymbol{o}) = \log\left[(2\pi)^{-D/2}|\boldsymbol{W}|^{-1/2}\right] - \frac{1}{2}(\boldsymbol{o}-\boldsymbol{\mu})'\boldsymbol{W}^{-1}(\boldsymbol{o}-\boldsymbol{\mu})$$

Assuming that $\boldsymbol{W}$ is the same for all states, and dropping the first term in the above expression gives the popular Mahalanobis distance. When $\boldsymbol{W}$ is assumed to be diagonal and the variances of all features are the same, the second term becomes equivalent to the squared Euclidean distance. In the HMM framework, the first term can be interpreted as a state-specific weighting factor derived from the covariance of the distribution. The Mahalanobis and Euclidean distances are invariant to non-singular rotations of the feature space, and the Mahalanobis distance is further invariant to scaling transformations.

6.3 Class separability

Class separability measures are usually generalisations of the F-ratio concept (see section 6.4.2). In a typical speech recognition task, improved class separability will not always yield a gain in recognition performance. This will depend to a large extent on the class definitions themselves and also on the definition of a meaningful measure of class separability which explicitly or implicitly relates to the overall recognition performance of the system. Measures of class separability are usually based on in-class and out-of-class statistics from the data representing the various classes. The in-class scatter measure for each class i is given by the matrix

$$\boldsymbol{W}_i = E[(\boldsymbol{x}_i - \boldsymbol{\mu}_i)(\boldsymbol{x}_i - \boldsymbol{\mu}_i)']$$

where $\boldsymbol{x}_i$ denotes all samples representing class i and $\boldsymbol{\mu}_i$ is the mean vector for class i. Assuming that the features representing each class are normally distributed, the in-class scatter measure is equivalent to the covariance matrix of the Gaussian distribution modelling that class. This equivalence is important since it gives a convenient way of computing $\boldsymbol{W}_i$ as a by-product in the HMM training procedure. The average in-class scatter is given by the pooled in-class covariance matrix, that is

$$\boldsymbol{W} = \frac{1}{n}\sum_i \boldsymbol{W}_i$$

where n is the number of classes. Ignoring in-class variation and representing each class by its mean $\boldsymbol{\mu}_i$, the between-class covariance can be computed as

$$\boldsymbol{B} = E[(\boldsymbol{\mu}_i - \boldsymbol{\mu}_g)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_g)']$$

where $\boldsymbol{\mu}_g$ is the mean of $\boldsymbol{\mu}_i$. Since the desired features should have small in-class covariance and large between-class covariance, measures of separability involve a ratio whose numerator is based on $\boldsymbol{B}$ and whose denominator is based on $\boldsymbol{W}$. Four such measures are described by Fukunaga [44] and the following two are of a particular interest since they remain invariant under non-singular rotations and scaling of the feature space

$$J_1 = tr(\boldsymbol{W}^{-1}\boldsymbol{B}) \tag{6.1}$$

$$J_2 = |\boldsymbol{W}^{-1}\boldsymbol{B}| \tag{6.2}$$

The J_1 measure can be interpreted by applying a similarity transformation $\boldsymbol{U}$ which simultaneously diagonalises $\boldsymbol{W}$ and $\boldsymbol{B}$. If such a transformation exists, J_1 can be interpreted as the sum of the F-ratios of the features in the transformed domain. The J_1 measure will have the same value before and after the transformation provided that all features are retained.

The determinant of each covariance matrix is equal to the product of its eigenvalues, and the eigenvalues in turn are proportional to the lengths of the axes of the hyper-elliptical contours of equal probability. Hence, the J_2 measure compares the scatter of the within-class and between-class distributions in terms of the squares of their multi-dimensional volumes.

6.4 Feature selection

Feature selection is a technique for reducing the dimensionality of the input vector by selecting a subset of the original feature set. This is accomplished by choosing a figure of merit to evaluate the performance of each individual feature and then selecting the best $\bar{D}$ features to use thereafter. However, the $\bar{D}$ features that give the best recognition performance when used jointly may not be the same as the $\bar{D}$ highest ranking individual features according to the chosen criterion. Evaluating the performance of each individual subset by performing recognition experiments is a computationally expensive and somewhat tedious task. In practice, this problem is usually ignored and it is common to assume that unless the features are linearly dependent, then each new feature will further aid correct classification and the performance of the selected subset will be very close to that of the best set of $\bar{D}$ features.

6.4.1 Selection based on power spectrum resolution

The power spectrum resolution can be used to visualise the ability to accurately represent the various classes in a speech recognition task using a particular feature set. When using cepstral parameters, it is well known that higher order cepstral coefficients contribute less to the power spectrum resolution. If the extra resolution provided by these higher-order coefficients is judged to be unimportant for a particular recognition task, the resolution of the power spectrum can be lowered by removing these parameters without any loss in recognition accuracy. In speech recognition, the cepstral parameters are more important than their first derivatives, which in turn, are more important than the second derivatives. This information can be used to manually devise a subset of the original set, where we include more cepstral coefficients than delta cepstral coefficients and more delta cepstral coefficients than delta delta cepstral coefficients [87]. Such selection makes sense since the finer detail attributed to the power spectrum by the higher order coefficients is likely to be lost when computing the delta parameters.

6.4.2 Selection using the F-ratio

A measure that can be used to evaluate the effectiveness of a particular feature is the F-ratio. It is defined as the ratio[1] of the between class variance $\boldsymbol{B}$ and the within-class

[1] Using matrix notation, this is given by $W^{-1}B$.

variance $\boldsymbol{W}$, where $\boldsymbol{W}$ and $\boldsymbol{B}$ are strictly diagonal. In the context of feature selection for pattern classification, the F-ratio can be used to select the features which maximises the separation between different classes and minimises the scatter within classes. The following assumptions are made when using the F-ratio as a method for reducing dimensionality:

- the feature vectors within each class must have Gaussian distribution;
- the features are uncorrelated;
- the variance within each class must be equal.

In practice the above conditions are rarely satisfied and the F-ratio cannot be used to evaluate more than one feature unless all the features are uncorrelated. A set of correlated features can be transformed into a set of independent ones, for which the F-ratios can be used to select the best combinations of features. This approach will be discussed as a feature extraction method.

6.4.3 Selection based on recognition performance

The relative merit by which speech recognition systems are judged is the recognition performance, hence, in judging the usefulness of each feature it makes sense to compute the contribution of each feature to the recognition performance and use it as a figure of merit to select subsets of features. In order to accomplish this, Paliwal [87] has used each individual feature to recognise all utterances in the training set. This is equivalent to running D recognition experiments - one for each feature in the original set. The D features are then rank-ordered using the error rate as a figure of merit and the top $\bar{D}$ features are selected. The method has the advantage that it makes a single assumption about the features being uncorrelated. However, the selected feature subset can depend on the speech recogniser used and on the actual training data used to measure the error rate. This approach bears a resemblance to the *knock-out* and *add-on* algorithms discussed in [88] which reduce and grow sets of features respectively one feature at a time based on overall recognition accuracy.

Bocchieri and Wilpon have described a feature selection method based on statistics gathered from the recognition errors made on the training set [17]. The Viterbi algorithm is used during training to provide (state, mixture)/frame alignment of each training utterance according to the correct transcription model $\mathcal{M}_{corr}$ and the generic recognition model $\mathcal{M}_{rec}$. The average distance vector is computed in each case according to

$$\mathcal{D}_d(\mathcal{M}) = \frac{1}{T}\sum_{t=1}^{T}\left\{\frac{(o_{t,d}-\mu_{\theta_t,\psi_t,d})^2}{\sigma^2_{\theta_t,\psi_t,d}}\right\} \qquad \text{for } d=1,\ldots,D \tag{6.3}$$

In the case when $\mathcal{M}_{rec}$ is used to produce the alignment, the summation over t is replaced by observation sequences corresponding to insertion and substitution errors. Since the Viterbi (state, mixture)/frame alignment is also used to compute ML estimates of the parameters of the HMMs, $\mathcal{D}_d(\mathcal{M}_{corr}) = 1.0$ for $d = 1,\ldots,D$. Therefore the feature components are rank-ordered according to the values of $\mathcal{D}_d(\mathcal{M}_{rec})$ which is also interpreted as the ratio of a

F-ratio		*Error rate*		*Class distance*	
$rank_{1-20}$	$21-40$	$rank_{1-20}$	$21-40$	$rank_{1-20}$	$21-40$
C_2	ΔC_9	C_1	C_8	ΔE	$\Delta^2 C_6$
ΔE	ΔC_{11}	ΔE	ΔC_{10}	$\Delta^2 E$	$\Delta^2 C_4$
C_1	ΔC_8	C_2	ΔC_{11}	C_4	$\Delta^2 C_1$
C_3	ΔC_{10}	ΔC_1	ΔC_5	C_6	C_9
C_4	$\Delta^2 C_3$	ΔC_2	ΔC_8	C_1	ΔC_8
C_{10}	ΔC_6	C_3	C_{12}	ΔC_4	$\Delta^2 C_5$
C_{11}	$\Delta^2 C_2$	$\Delta^2 E$	ΔC_7	C_5	$\Delta^2 C_7$
ΔC_2	ΔC_7	ΔC_3	ΔC_6	ΔC_6	$\Delta^2 C_9$
C_5	ΔC_{12}	C_4	$\Delta^2 C_7$	ΔC_5	ΔC_{10}
C_8	$\Delta^2 C_4$	$\Delta^2 C_1$	$\Delta^2 C_9$	C_2	$\Delta^2 C_8$
ΔC_1	$\Delta^2 C_{10}$	C_5	ΔC_{12}	ΔC_3	C_{11}
ΔC_3	$\Delta^2 C_9$	C_{10}	$\Delta^2 C_8$	C_3	ΔC_{11}
C_9	$\Delta^2 C_5$	C_9	$\Delta^2 C_4$	C_7	C_{10}
$\Delta^2 E$	$\Delta^2 C_{11}$	C_{11}	$\Delta^2 C_6$	ΔC_2	$\Delta^2 C_{10}$
C_7	$\Delta^2 C_8$	$\Delta^2 C_3$	$\Delta^2 C_{10}$	ΔC_1	C_{12}
C_6	$\Delta^2 C_7$	$\Delta^2 C_2$	$\Delta^2 C_{11}$	ΔC_7	$\Delta^2 C_{11}$
C_{12}	$\Delta^2 C_6$	C_7	$\Delta^2 C_5$	ΔC_9	ΔC_{12}
ΔC_5	$\Delta^2 C_{12}$	ΔC_4	$\Delta^2 C_{12}$	$\Delta^2 C_3$	$\Delta^2 C_{12}$
ΔC_4		C_6		$\Delta^2 C_2$	
$\Delta^2 C_1$		ΔC_9		C_8	

Table 6.1: LPC cepstral feature ordering using different criteria: *F-ratio* and recognition *Error rate* figures of merit as used by Paliwal; *Class Distance* method based on recognition error rate as studied by Bocchieri and Wilpon.

between-class and within-class scatter measure. There are two potential drawbacks of this method. First, it requires errors to be defined explicitly in terms of their boundaries and it is also incapable of dealing with deletion errors explicitly. The second drawback is the fact that the Viterbi (state, mixture)/frame alignment can be a poor approximation to the sum of all possible paths through the model. This has the effect of removing multiple competing paths in favour of a single path with a marginally better score.

6.4.4 Previous results

Bocchieri and Wilpon [17] investigated the performance of their feature selection scheme by performing experiments on a variety of databases. The feature rank-order was derived using 3444 sentences from the training portion of the TIMIT database. The data was parametrised using 12 LPC cepstral coefficients, their first derivatives (computed using full regression) and their second derivatives (computed using simple differences). The first and second derivatives of the energy were also included to make up a 38 dimensional parameter vector. Subsets of the original feature vector were evaluated on test utterances from several databases including the TIMIT database, the TI connected digit database and an "in-house" E-set database. The derived rank-order is reproduced in the column labelled "*Class distance*" in table 6.1. On the TIMIT database the authors achieved a baseline phone accuracy of 59.8% using 3 state, 16 Gaussian component mixture per state context independent models. The feature vector was successfully reduced to 24 elements with a marginal improvement in performance (60.0%) and then further reduced to 18 elements with minimal loss in performance (59.2%).

A comparison of different feature selection schemes was carried out by Paliwal in [87]. In his study the author performed multi-speaker recognition experiments on an isolated alpha-digit database recorded over the telephone using an LPC-based feature set identical to the one used by Bocchieri and Wilpon. The feature rank-order using the *F-ratio* and the recognition *Error rate* on the training set are reproduced in table 6.1. Without any degradation in performance, the feature set was reduced to 24 and 16 components for the *F-ratio* and the recognition *Error rate* criteria respectively. In fact, in both cases the performance was marginally better than the result obtained using the original 38 dimensional feature vector. In addition, Paliwal also performed experiments using several manually derived feature subsets. These include the first 8-10 cepstral coefficients, 3-8 delta cepstral coefficients, 3-4 delta-delta cepstral coefficients and ΔE and $\Delta^2 E$. Surprisingly, these manually selected subsets of the original feature vector were found to perform better than any of the other feature reduction methods investigated in that study.

6.5 Feature extraction

Feature extraction methods reduce dimensionality by projecting the original D dimensional feature space onto a smaller subspace through a transformation. Each feature in the transformed set is computed as a linear combination of all elements in the original feature set. A number of such transformations are reported in the literature. In speech coding work the discrete form of the Karhunen-Loeve (K-L) transformation is used to de-correlate the samples within a given frame of speech. The discrete cosine transform (DCT) is a data-independent transformation with performance close to the optimal K-L transformation. The K-L transformation has the sole aim of de-correlating a set of features or samples. Linear discriminant analysis (LDA) takes this process one step further and focuses on the between class variations in the signal to find directions of maximum separability.

6.5.1 Principal component analysis (PCA)

In the method of principal components, dimensionality reduction is achieved by projecting the original D-dimensional feature space on a $\bar{D}$ dimensional subspace and finding the orientation of the subspace which best preserves the information available in the original space. The first principal component of a pattern vector is the projection of that sample onto the direction of largest variance as estimated over all samples. The rationale behind principal component analysis is the assumption that the direction of maximum variance contains most of the information about the various classes that the input patterns represent. Indeed, principal component analysis is identical in formulation to the K-L transformation used in speech coding where a transformation is derived which approximates each D-dimensional pattern vector by a $\bar{D}$ dimensional vector which minimises the mean square error of the different representations over all samples. The resulting transformation is given by a $D \times \bar{D}$ unitary matrix $\boldsymbol{U}$ whose columns are the eigenvectors corresponding to the $\bar{D}$ largest eigenvalues of the grand covariance matrix of the data. When no dimensionality reduction is performed e.g. $\bar{D} = D$, this transformation amounts to a rotation in the feature space and in this case the Mahalanobis distance and the Euclidean distances remain invariant. However, three potential drawbacks can be identified

- Principal components are not invariant to parameter scale. If the d^{th} parameter in each input vector is multiplied by a large constant then the variance of that parameter will become larger and larger and the first principal component will approach this scaled parameter. The problem stems from the fact that scaling up parameters cannot possibly affect the amount of information encoded in them. Principal components can be made invariant to transformations of the input vectors by using diagonal matrices to scale the parameters to achieve unit variance.

- PCA is based on a transformation derived from the grand covariance matrix of the data - e.g. the matrix computed over all samples without any knowledge about classes. In a typical speech recognition setup a large proportion of training data corresponds to background silence/noise, typically at the start and the end of each utterance. Consequently, the grand covariance matrix will be dominated by the covariances of the distributions employed in the silence model. If this is the case, without dimensionality reduction the PCA transformation will simply diagonalise the covariance matrix of the silence model. With reduced dimensionality, the transformation will rotate the vectors such as to preserve primarily the information which will be useful in recognising silence.

- The final drawback of PCA is the fact that the direction of maximum variation does not necessarily contain most of the information which aids class discrimination. This problem is depicted in figure 6.1 where projections of the sample vector onto the first principal component does not provide any information as to what class the input sample belongs. Indeed, the direction of maximum class separability is perpendicular to the first principal component.

6.5.2 Linear discriminant analysis (LDA)

Linear discriminant analysis (LDA) is a technique used in statistical pattern recognition, where a linear transformation is found for extracting a set of features which are most important for discriminating between different classes. LDA is also equivalent to finding a linear transformation of the feature space which maximises the J_1 and J_2 measures of class separability. Consequently, the quality of the selected set of features will depend on

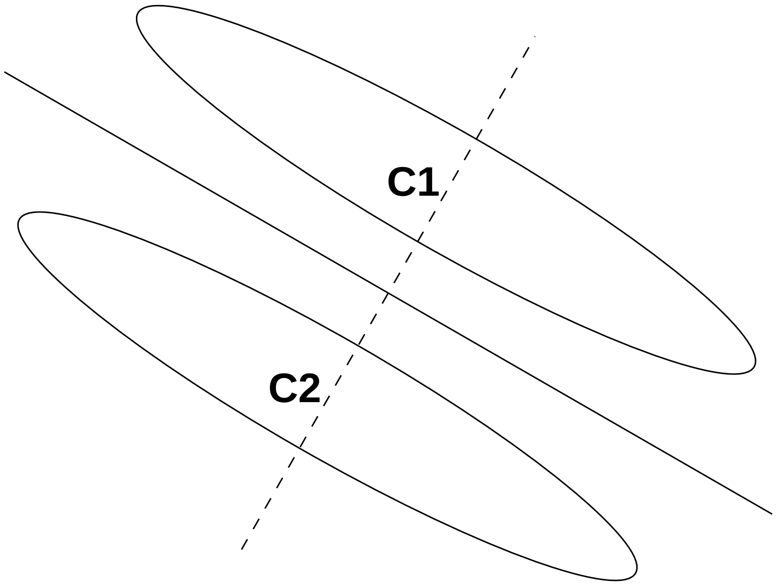

Figure 6.1: Gaussian classes with equal covariance matrices

how accurately J_1 and J_2 measure class-separability and how well class discrimination aids improved overall recognition performance. LDA makes the following three assumptions: 1) each class can be represented by a single Gaussian distribution with in-class covariance matrix $\boldsymbol{W}_i$; 2) the covariance matrices of all classes are identical i.e. $\boldsymbol{W}_i = \boldsymbol{W}$ for all i; and 3) the class centroids themselves can be represented by a single Gaussian distribution with between-class covariance matrix $\boldsymbol{B}$. The assumption that the features are uncorrelated is not necessary, hence LDA can be considered as an extension of the F-ratio feature selection method that maximises the F-ratio of the data in the transformed space.

Finding the matrix $\boldsymbol{S}$ used in the LDA transformation involves a simultaneous diagonalisation of the $\boldsymbol{W}$ and $\boldsymbol{B}$ matrices. The matrix $\boldsymbol{S}$ is found such that

$$\boldsymbol{S}'\boldsymbol{W}\boldsymbol{S} = \boldsymbol{I}$$

and

$$\boldsymbol{S}'\boldsymbol{B}\boldsymbol{S} = \boldsymbol{\Gamma}$$

where $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{\Gamma}$ is diagonal. This operation can be split into three steps. The first two steps together perform a full rank factorisation of $\boldsymbol{W}$ and consist of a rotation $\boldsymbol{R}_1$ to diagonalise $\boldsymbol{W}$

$$\boldsymbol{R}_1'\boldsymbol{W}\boldsymbol{R}_1 = \boldsymbol{\Lambda} = diag(\Lambda_1, \Lambda_2, \ldots, \Lambda_D)$$

followed by a scaling which transforms the hyper-ellipsoids into hyper-spheres

$$\boldsymbol{\Lambda}^{-1/2}\boldsymbol{R}_1'\boldsymbol{W}\boldsymbol{R}_1\boldsymbol{\Lambda}^{-1/2} = \boldsymbol{I} \tag{6.4}$$

The columns of $\boldsymbol{R}_1$ are the eigenvectors of $\boldsymbol{W}$ and $\boldsymbol{\Lambda}$ is the diagonal matrix of corresponding eigenvalues. In fact, the matrix $\boldsymbol{R}_1$ is equivalent to the transformation matrix of PCA. Since $\boldsymbol{R}_1$ is orthogonal, the above expression can be rearranged as

$$\boldsymbol{W} = (\boldsymbol{R}_1\boldsymbol{\Lambda}^{1/2})(\boldsymbol{\Lambda}^{1/2}\boldsymbol{R}_1') = \boldsymbol{S}_1\boldsymbol{S}_1'$$

The whitening transform $\boldsymbol{S}_1$ is then used to rotate and scale both covariance matrices.

The second stage in LDA involves diagonalising the transformed $\boldsymbol{B}$ matrix which is accomplished by finding the eigenvector matrix $\boldsymbol{R}_2$ and the eigenvalues $\boldsymbol{\Gamma}$ of the resulting $\boldsymbol{S}_1'\boldsymbol{B}\boldsymbol{S}_1$ matrix. This is equivalent to solving

$$|\boldsymbol{S}_1'\boldsymbol{B}\boldsymbol{S} - \boldsymbol{\Gamma}\boldsymbol{I}| = 0$$

Using equation 6.4 and the fact that $|\boldsymbol{A}||\boldsymbol{C}| = |\boldsymbol{A}\boldsymbol{C}|$

$$|\boldsymbol{S}_1'\boldsymbol{B}\boldsymbol{S}_1 - \boldsymbol{\Gamma}\boldsymbol{S}_1'\boldsymbol{W}\boldsymbol{S}_1| = |\boldsymbol{S}_1'||\boldsymbol{B} - \boldsymbol{\Gamma}\boldsymbol{W}||\boldsymbol{S}_1| = 0$$

and pre-multiplying by $\boldsymbol{W}^{-1}$

$$|\boldsymbol{W}^{-1}\boldsymbol{B} - \boldsymbol{\Gamma}\boldsymbol{I}| = 0$$

confirming that this simultaneous diagonalisation problem results in the problem of finding the eigenvalues of $\boldsymbol{W}^{-1}\boldsymbol{B}$ which in turn can be used to evaluate J_1.

The orthogonal matrix of eigenvectors $\boldsymbol{R}_2$ performs the required diagonalisation of the transformed $\boldsymbol{B}$ matrix giving

$$\boldsymbol{R}_2'\boldsymbol{\Lambda}^{-1/2}\boldsymbol{R}_1'\boldsymbol{W}\boldsymbol{R}_1\boldsymbol{\Lambda}^{-1/2}\boldsymbol{R}_2 = \boldsymbol{I}$$

and

$$\boldsymbol{R}_2'\boldsymbol{\Lambda}^{-1/2}\boldsymbol{R}_1'\boldsymbol{B}\boldsymbol{R}_1\boldsymbol{\Lambda}^{-1/2}\boldsymbol{R}_2 = \boldsymbol{\Gamma} = diag(\Gamma_1, \Gamma_2, \ldots, \Gamma_D)$$

The final linear transformation $\boldsymbol{S}$ is expressed as a product of three matrices representing two rotations and a scaling

$$\boldsymbol{S} = \boldsymbol{R}_1\boldsymbol{\Lambda}^{-1/2}\boldsymbol{R}_2 \tag{6.5}$$

The final step in LDA is dimensionality reduction. The final transformation is given by a matrix $\boldsymbol{U}$, where $\boldsymbol{U}$ is a $D \times \bar{D}$ matrix whose columns are the eigenvectors corresponding to the $\bar{D}$ largest eigenvalues of the matrix $\boldsymbol{W}^{-1}\boldsymbol{B}$. This can be implemented by sorting the columns of $\boldsymbol{S}$ in decreasing eigenvalue order and using the first $\bar{D}$ columns of $\boldsymbol{I}$ to form a truncation matrix $\boldsymbol{F}$ such that

$$\boldsymbol{U} = \boldsymbol{R}_1\boldsymbol{\Lambda}^{-1/2}\boldsymbol{R}_2\boldsymbol{F} \tag{6.6}$$

Figure 6.2 (after Parsons [88]) shows the effect of the effect of the LDA transformation applied to three classes with identical covariance matrices. The original in-class distributions are shown in (a) as ellipses representing equal probability contours. The effect of the first transformation ($\boldsymbol{R}_1$) is to remove correlations between the features (b). This is followed by a scaling ($\boldsymbol{\Lambda}^{1/2}$) which "whitens" the distributions (c) after which, the elliptical contours have been transformed in to circles. The final stage (d) assumes that the class centroids are normally distributed around the mean of the data and performs a further rotation ($\boldsymbol{R}_2$) which orientates the classes for maximum separability.

With any feature extraction technique, after projecting the components of the feature vector onto a subspace, the physical meaning of each parameter is lost. As Brown [22] points out, this is not a problem provided that the output distributions in the HMMs do not rely on any prior information about the physical meaning of the original parameters. Sometimes a representation of the new feature set in terms of the original parameters can be useful [9] and in such cases further orthogonal transformations can be applied to undo the rotations in expression 6.6. Using spectral parametrisation, such inverse transformations can be used to study the enhancement provided by the LDA transformation of those spectral features which contribute most to maximising class separability.

6.5.3 Confusion data analysis

Doddington [32] and Woodland [109] have both investigated methods for improving the discrimination capabilities of standard HMMs via the introduction of state-specific discriminative transformations. In the phonetic discriminant model, the aim is to create a metric which enhances the discrimination between the "true" phonetic state and all other competing states which are confusable with it. This is achieved with the introduction of state-specific linear discriminative transformations which transform each feature vector modelled by that state. The observation likelihood is then computed as the Euclidean distance between the transformed feature vector and the reference vector. The approach was originally outlined in [16] where it was applied to an isolated word task using a DTW template based recogniser. It was later extended to the SI recognition of connected digit strings using continuous density HMMs. In the latter case, the number of states in the HMMs were set to the average length of the utterances they modelled which, the author argued, allowed for a "more sensitive temporal model".

The phonetic discriminant model is made up of two parts. The first part is the usual in-class model characterised by the mean vector and the covariance matrix of the underlying data for that state. The second part is the *confusion data* model which is characterised by the mean vector and the covariance matrix of all speech frames that are incorrect for the given state but are "plausible" based on probability calculations. Using this information a state-specific linear transformation is derived which improves the discrimination between data modelled by the "true" distribution and acoustically confusable data modelled by other states in the HMM. The transformation for each state is computed using the algorithm for computing the LDA transformation described in section 6.5.2. For the calculation, the average in-class covariance matrix $\boldsymbol{W}$ is replaced by the covariance matrix $\boldsymbol{W}_i$ from the Gaussian distribution of the given state, and the between-class covariance matrix $\boldsymbol{B}$ is replaced by the covariance matrix of the confusion data for the given state, re-centred around the in-class mean. After applying the transformation, each component of the feature vector is uncorrelated for both the in-class and out-of-class (confusion) data, and each feature of the in-class data has unity variance. To improve discrimination, dimensions in which the confusable data has smaller variance than the in-class data are discarded.

A major step in deriving the transformations is the process of gathering confusion data. In Doddington's work [32] this is carried out by producing state/frame alignments for each training utterance using the correct transcription and a "free" grammar which allows all possible word sequences. The free grammar is biased against the correct word sequence using a "prejudice" parameter. The confusion model is computed from data corresponding to those portions of the alignment using the free grammar which disagree with the correct word label. Woodland and Cole [109] have investigated a Viterbi based approach and a frame based approach to gathering confusion data. The former is similar to the method used by Doddington, however, multiple thresholds are used to define "near-misses" and confusion data is simultaneously accumulated for all thresholds. In the frame based approach, a frame is classified as confusable if its probability given a state from an incorrect model is within a certain distance of the frame's probability produced by the most likely state of the correct model.

6.5.4 Previous results

Brown performed experiments on an American E-set recognition task where he compared the performance of PCA and a technique called "linear discriminants" which is similar to LDA. Parametrisation was based on 20 features derived by an "ear model" from the output of a 20 channel filter-bank. Pairs of consecutive frames were joined to form a 40 dimensional observation vector. With fewer than 24 features, linear discriminants were always found to perform better than principal components. For both techniques, the performance initially improved as the dimension of the feature vector was reduced.

Woodland and Cole [109] evaluated the performance of the state-specific discriminative transformations on the E-set portion of the BTL database (appendix A). Using Viterbi confusion analysis the error rate was reduced by 11% for 16 features from the original 24 dimensional vector which was based on MFCCs and their first derivatives. The authors showed that the use of the correct Gaussian normalisation term computed in the reduced feature space is important for achieving high recognition accuracy.

Doddington [32] evaluated the performance of his state-specific "Phonetically Sensitive" transformation on the TI/NIST connected digit database. Using LPC-derived cepstral parametrisation, he achieved 0.5% (1.5%) word (sentence) error rates for digit strings of unknown length. Later, he discovered that the features in the Euclidean distance had been unintentionally weighted by their corresponding standard deviation which had a beneficial effect on the results published in [32]. This variance weighting scheme was further investigated in [33].

IMELDA is an application of LDA to the speech recognition problem developed by Hunt

Type	*Num. parameters*	*add/multiply*
Diag. Covariance	$M + 2D$	$2MD$
Full Covariance (1)	$M + (D^2 + 3D)/2$	$M(D^2 + D)$
Full Covariance (2)	$M + (D^2 + 3D)/2$	$M(D^2 + 3D)/2$

Table 6.2: Number of parameters/computational requirements for different types of output distributions with M mixture components and feature vector size D.

[52]. In IMELDA, linear discriminant analysis is applied to the output of a mel-scale filter bank to derive a global feature transformation. This transformation has been shown to give an improved set of features for speech recognition when compared to other representations [53, 54]. Results in [53] indicate that IMELDA can be used to transform heterogeneous sets of features and reduce dimensionality without loss in recognition performance and it can also provide robustness to signal degradation.

6.6 Computational considerations

In this section we examine the computational requirements for evaluating full and diagonal covariance Gaussian output distributions. As discussed previously each state in an HMM has an associated output distribution in the form

$$b_j(\boldsymbol{o}_t) = \sum_{m=1}^{M} c_{j,m} \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{W}_{j,m}|}} e^{-\frac{1}{2}(\boldsymbol{o}_t - \boldsymbol{\mu}_{j,m})' \boldsymbol{W}_{j,m}^{-1} (\boldsymbol{o}_t - \boldsymbol{\mu}_{j,m})}$$

The constant term in the Gaussian expression can be precomputed and stored. The majority of the computation is concentrated in the evaluation of the exponent. Table 6.2 lists the number of parameters in each case together with the number of add/multiply operations necessary to evaluate the exponent. In the full covariance case, straight forward implementation requires $(D^2 + D)$ add/multiply operations. However, $\boldsymbol{W}_{j,m}^{-1}$ is a positive definite matrix and as such it may be decomposed into

$$\boldsymbol{W}_{j,m}^{-1} = \boldsymbol{L}_{j,m} \boldsymbol{L}'_{j,m}$$

where $\boldsymbol{L}_{j,m}$ is a lower triangular matrix (Cholesky factorisation). Using expression 6.6 the Gaussian exponent term can be rewritten as

$$-\frac{1}{2} \left[(\boldsymbol{o}_t - \boldsymbol{\mu}_{j,m})' \boldsymbol{L}_{j,m} \right] \left[(\boldsymbol{o}_t - \boldsymbol{\mu}_{j,m})' \boldsymbol{L}_{j,m} \right]'$$

In this form, the computation will require only $(D^2 + 3D)/2$ add/multiply operations.

The above derivations show that in the case of state-specific input transformations as used by Woodland and Doddington, computational savings are only available if the dimensionality of the feature vectors is reduced by more than a half. This follows from the fact that the full discriminative transformations are not symmetric and as such they contain almost twice as many parameters as the corresponding covariance matrices in the Gaussian distributions.

6.7 Summary

This chapter has reviewed a variety of dimensionality reduction methods that can be used in the HMM framework to improve discrimination and reduce computational requirements. Feature selection methods are somewhat simpler than the feature extraction schemes since they do not alter the existing structure of the speech recognition system. Computational savings are thus immediately available when unimportant features are removed from the observation vectors. Feature extraction schemes are based on linear transformations of the original feature space. A single global transformation can be implemented as a matrix multiplication within the acoustic preprocessor. The most complex feature extraction schemes are the state-specific discriminative transformations described by Doddington and Woodland. The complexity of these models in terms of the number of parameters is almost twice as much as the number of parameters used in the corresponding full covariance Gaussian models. In terms of computation, they do not offer any savings unless the dimensionality is reduced by a half, however, the increased number of parameters does provide improved recognition accuracy.

Several of the feature selection schemes and virtually all feature extraction schemes rely on the appropriate "class" definitions which are typically chosen as the states of the HMMs. Hence, it is not clear how improving discrimination at frame/state level will affect the overall recognition performance which is measured at phone/word/sentence level.

In the light of the work reviewed in this chapter, the following two chapters will describe alternative ways for feature selection/extraction based on the mutual information criterion.

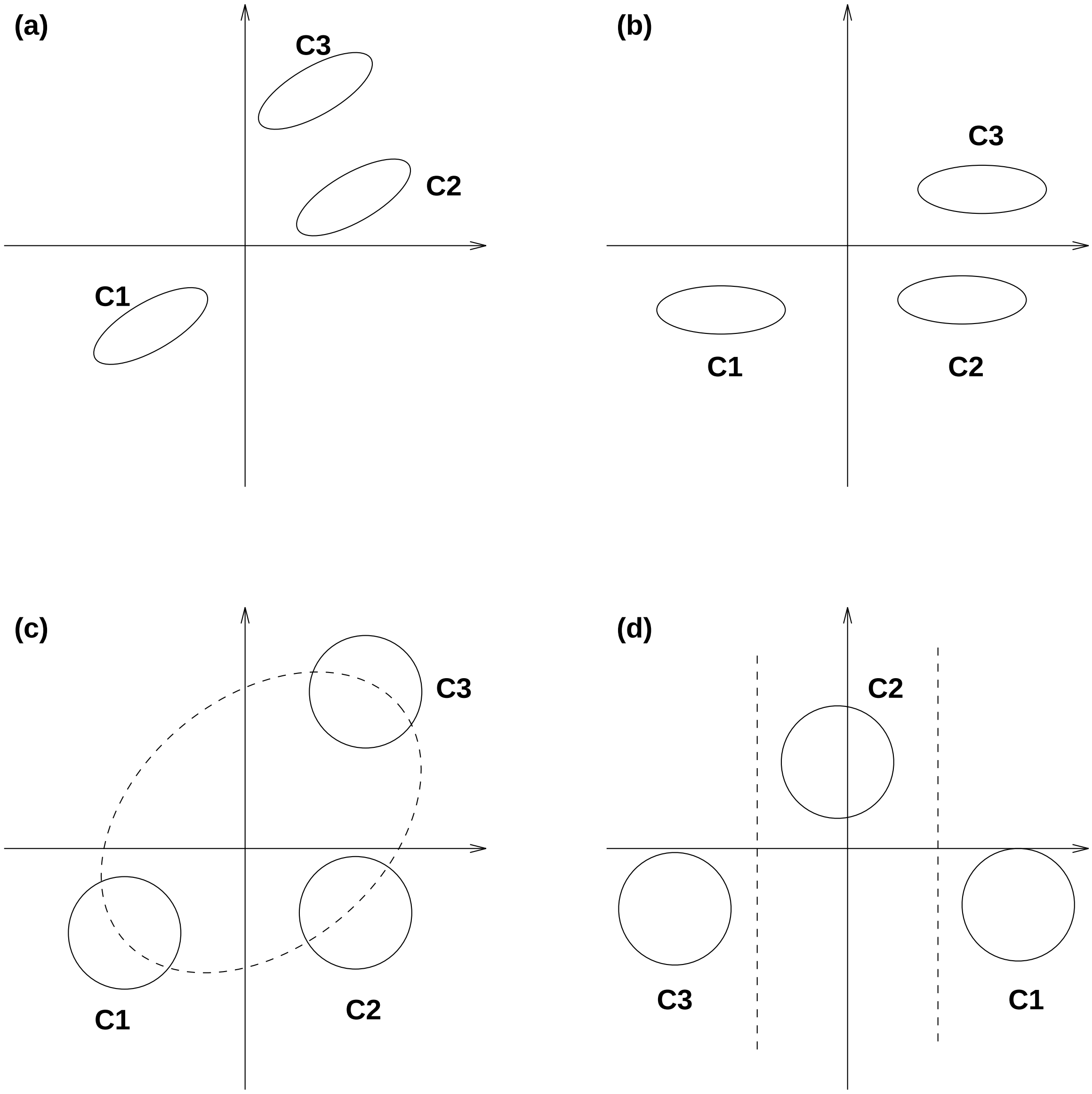

Figure 6.2: Effects of the LDA transformation

Chapter 7

Mutual Information based Feature Selection

The previous chapter outlined a variety of feature selection schemes where the components of the observation vector are ranked according to some discriminative criterion. Feature selection was found to be an important technique for reducing the number of parameters used in the output distributions in an HMM-based system. In this chapter we present an alternative feature selection scheme based on the mutual information measure between the models and the training data.

7.1 Motivation

As discussed earlier, the performance of a speech recognition system is judged by its ability to recognise unknown utterances correctly. Consequently, the relative merit of a feature should be valued by its contribution to the overall recognition performance. In chapter 3 we also discussed how maximising the mutual information between the acoustic model and the training data can result in improved recognition performance. Hence, using a mutual information related measure to rank the features in the observation vectors can provide information as to which features are more useful than others. Removing a feature from the observation vector will inevitably change the mutual information between the model set and the training data. In MMIE training the aim was to maximise this quantity. Hence, when reducing the feature set, it would be desirable to remove features which yield minimal decrease in the mutual information. As we discussed in the previous chapter, finding the best subset of features is a difficult task, consequently, one has to assume that all features are independent. In the remaining part of this chapter, we shall derive a method which will provide an estimate of how much the mutual information will change if a particular feature from the observation vector is deleted. Since the method attempts to minimise the change in the mutual information of the reduced feature set, the procedure will be called Minimum Mutual Information Change (MMIC) feature selection.

7.2 Theory

The MMIE objective function plays a central role in the definition of the saliency of a parameter. It would be prohibitively expensive to explicitly evaluate the change in the objective function by deleting each parameter in turn. Fortunately, it is possible to construct a local model of the function and analytically predict the effect of perturbing the parameters in the observation vector. Consistent with previous notation, in the following derivations it will be assumed that the training data consists of R training utterances where each training utterance $\mathcal{O}$ is a sequence of observation vectors

$$\mathcal{O} = \boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_t, \ldots, \boldsymbol{o}_T$$

The change in the objective function after deleting a component from the feature vector comes from the change in the values of the output probabilities in the model. The proposed feature selection algorithm will be applied to continuous density HMMs with mixture Gaussian distributions with diagonal covariance matrices. For the output distribution at state j and mixture component m we have

$$b_{j,m}(\boldsymbol{o}_t) = \prod_{d=1}^{D} \left\{ \frac{1}{\sqrt{2\pi\sigma^2_{j,m,d}}} e^{-\frac{(o_{t,d}-\mu_{j,m,d})^2}{2\sigma^2_{j,m,d}}} \right\} \tag{7.1}$$

The effect of removing features can be studied by introducing an exponent for each parameter in the form

$$b_{j,m}(\boldsymbol{o}_t) = \prod_{d=1}^{D} \left\{ \frac{1}{\sqrt{2\pi\sigma^2_{j,m,d}}} e^{-\frac{(o_{t,d}-\mu_{j,m,d})^2}{2\sigma^2_{j,m,d}}} \right\}^{u_d} \tag{7.2}$$

The elements of $\boldsymbol{u}$ will be referred to as the feature weights and the effect of removing feature component d will be equivalent to setting $u_d = 0$. As discussed in the chapter 4, a function of many variables can be locally approximated by a truncated Taylor series of the form

$$f(\boldsymbol{y}) \simeq f(\boldsymbol{x}) + \boldsymbol{\nabla}' f(\boldsymbol{x})(\boldsymbol{y}-\boldsymbol{x}) + \frac{1}{2}(\boldsymbol{y}-\boldsymbol{x})'\boldsymbol{\nabla}^2 f(\boldsymbol{x})(\boldsymbol{y}-\boldsymbol{x}) \tag{7.3}$$

In particular, in order to evaluate the components of the feature set we shall further simplify conditions by dropping the last term in the above expression and consider a first derivative approximation to the value of the MMIE objective function with respect to the weight vector $\boldsymbol{u}$. This will enable us to find a new $\hat{\boldsymbol{u}}$ with only d non-zero elements which yields minimal decrease in the value of f. Setting elements of $\boldsymbol{u}$ to zero has the ultimate effect of reducing the observation vector size which also entails reduction in the number of parameters used in the mixture densities. Recall from chapter 3 that the MMIE objective function is essentially the difference of two likelihoods.

$$f_{MMI}(\lambda) = \frac{1}{R}\sum_{r=1}^{R} \log P_\lambda(\mathcal{O}_r|\mathcal{M}_{w_r}) - \frac{1}{R}\sum_{r=1}^{R} \log P_\lambda(\mathcal{O}_r|\mathcal{M}_{rec}) \tag{7.4}$$

In order to estimate the change in the mutual information after deleting a component, we will have to approximate the changes in the likelihood for the numerator and denominator expressions. Combining equations 7.3 and 7.4 and using $\boldsymbol{u}$ as the parameter set λ gives

$$\begin{aligned} f_{MMI}(\hat{\boldsymbol{u}}) &\simeq f_{MMI}(\boldsymbol{u}) + \boldsymbol{\nabla}' f_{MMI}(\boldsymbol{u})(\boldsymbol{u} - \hat{\boldsymbol{u}}) \\ &= f_{MMI}(\boldsymbol{u}) + \sum_{d, \hat{u}_d = 0} \frac{\partial}{\partial u_d} f_{MMI}(\boldsymbol{u}) \end{aligned} \quad (7.5)$$

Since we are interested in fully removing components from the observation vector rather than modifying the individual exponents, it is clear by inspection of 7.5 that minimal change in the value of $f_{MMI}(\hat{\boldsymbol{u}})$ is guaranteed when setting the elements of $\hat{\boldsymbol{u}}$ to zero according to the increasing order of their corresponding derivative values. Differentiating equation 7.4 with respect to $\boldsymbol{u}$ yields

$$\frac{\partial}{\partial \boldsymbol{u}} f_{MMI}(\boldsymbol{u}) = \frac{1}{R} \sum_{r=1}^{R} \frac{1}{P_\lambda(\mathcal{O}_r|\mathcal{M}_{w_r})} \frac{\partial}{\partial \boldsymbol{u}} P_\lambda(\mathcal{O}_r|\mathcal{M}_{w_r})$$
$$- \frac{1}{R} \sum_{r=1}^{R} \frac{1}{P_\lambda(\mathcal{O}_r|\mathcal{M}_{rec})} \frac{\partial}{\partial \boldsymbol{u}} P_\lambda(\mathcal{O}_r|\mathcal{M}_{rec}) \quad (7.6)$$

From appendix B we have

$$\frac{\partial}{\partial \boldsymbol{u}} P_\lambda(\mathcal{O}|\mathcal{M}) = \sum_{t=1}^{T} \sum_{j=1}^{N} \left\{ \sum_{i=1}^{N} \alpha_i(t-1) a_{i,j} \right\} \beta_j(t) \frac{\partial}{\partial \boldsymbol{u}} b_j(\boldsymbol{o}_t) \quad (7.7)$$

where $b_j(\boldsymbol{o}_t)$ is a mixture Gaussian (section 2.2.2) with derivative

$$\frac{\partial}{\partial \boldsymbol{u}} b_j(\boldsymbol{o}_t) = \sum_{m=1}^{M} c_{j,m} \frac{\partial}{\partial \boldsymbol{u}} b_{j,m}(\boldsymbol{o}_t) \quad (7.8)$$

Differentiating equation 7.2 with respect to $\boldsymbol{u}$ gives

$$\begin{aligned} \frac{\partial}{\partial u_d} b_{j,m}(\boldsymbol{o}_t) &= b_{j,m}(\boldsymbol{o}_t) \log \left\{ \frac{1}{\sqrt{2\pi\sigma^2_{j,m,d}}} e^{-\frac{(o_{t,d} - \mu_{j,m,d})^2}{2\sigma^2_{j,m,d}}} \right\} \\ &= b_{j,m}(\boldsymbol{o}_t) \frac{1}{2} \left\{ -\log(2\pi\sigma^2_{j,m,d}) - \frac{(o_{t,d} - \mu_{j,m,d})^2}{\sigma^2_{j,m,d}} \right\} \end{aligned} \quad (7.9)$$

Combining equations 7.7, 7.8, 7.9 gives

$$\frac{\partial}{\partial u_d} P_\lambda(\mathcal{O}|\mathcal{M}) = \quad (7.10)$$
$$\sum_{t=1}^{T} \sum_{j=1}^{N} \mathcal{C}_{t,j}(\mathcal{O}, \mathcal{M}) \sum_{m=1}^{M} c_{j,m} b_{j,m}(\boldsymbol{o}_t) \frac{1}{2} \left\{ -\log(2\pi\sigma^2_{j,m,d}) - \frac{(o_{t,d} - \mu_{j,m,d})^2}{\sigma^2_{j,m,d}} \right\}$$

The common factor $\mathcal{C}_{t,j}(\mathcal{O}, \mathcal{M})$ is defined as

$$\mathcal{C}_{t,j}(\mathcal{O}, \mathcal{M}) = \frac{1}{P_\lambda(\mathcal{O}|\mathcal{M})} \frac{\alpha_j(t)\beta_j(t)}{b_j(\boldsymbol{o}_t)}$$

The derivatives of the MMIE objective function with respect to the elements of $\boldsymbol{u}$ are calculated as the difference of the corresponding derivatives from the numerator and the denominator log-likelihoods (equation 7.6).

7.3 The algorithm

The MMIC feature selection algorithm is summarised below

1. Train a set of models for the specified task using conventional MLE training.
2. Compute the derivatives of the mutual information objective function according to the formulae in the preceding section. The grammar used to compute the derivatives of the denominator of the objective function should be as close as possible or ideally identical to the task recognition grammar, incorporating all language constraints.
3. Order the observation vector components according to the value of the derivative of the corresponding weight.
4. Select the top $\bar{D}$ components to form the new feature vector.
5. Retrain the system from scratch using the new feature set and evaluate its performance.

In practice, retraining the system from scratch was found not to be necessary. Instead, the feature selection procedure was followed by 2-3 iterations of embedded Baum-Welch training to readjust the parameters of the models according to the new state/frame alignment.

7.4 Simplifications

Before we establish the relationship between the MMIC feature selection procedure and other methods we shall consider the following simplification. As we discussed in chapter 2 the total likelihood $P_\lambda(\mathcal{O}|\mathcal{M})$ of a speech utterance $\mathcal{O}$ given a model $\mathcal{M}$ can be calculated as the sum of the probabilities of all paths through the model. If we make the assumptions that 1) this sum is dominated by the most likely path (the Viterbi path) and 2) the probability of an observation vector is provided by the best mixture component, then we can calculate the exact contribution of each feature to the value of the MMIE objective function. We have

$$\begin{aligned}\log P_\lambda(\mathcal{O}|\mathcal{M}) &\approx \sum_t \log a_{\theta_{t-1},\theta_t} + \sum_t \log b_{\theta_t}(\boldsymbol{o}_t) \\ &= \sum_t \log a_{\theta_{t-1},\theta_t} + \sum_t \log c_{\theta_t,\psi_t} + \sum_t \log b_{\theta_t,\psi_t}(\boldsymbol{o}_t)\end{aligned} \tag{7.11}$$

where $\theta_t \in \Theta$ and $\psi_t \in \Psi_\theta$ are the state and mixture component respectively at time t. Hence, the exact contribution of each feature vector component to the overall log-likelihood is given by

$$\Omega(d) = -\frac{1}{2}\sum_{t}\left\{\log(2\pi\sigma^2_{\theta_t,\psi_t,d}) + \frac{(o_{t,d} - \mu_{\theta_t,\psi_t,d})^2}{\sigma^2_{\theta_t,\psi_t,d}}\right\} \tag{7.12}$$

When the Viterbi state/mixture approximation to the overall likelihood is good the above expression is equivalent to the derivative of log $P_\lambda(\mathcal{O}|\mathcal{M})$ with respect to u_d (equation 7.10) since

$$\frac{\alpha_j(t)\beta_j(t)}{P_\lambda(\mathcal{O}|\mathcal{M})} = \begin{cases} 1 & \text{if } \theta_t = j \\ 0 & \text{otherwise} \end{cases}$$

Hence, if the Viterbi state/mixture alignment is a good approximation to the overall likelihood the MMIC feature selection algorithm will produce exact results. The first term on the right in equation 7.12 can be interpreted as a state/mixture-specific variance weighting provided by the Gaussian normalisation term. The second term can be interpreted as the scaled distance between the d^{th} component of the observation vector and its corresponding mean value at state θ_t, mixture component ψ_t. Finally, for a fully converged maximum likelihood trained system we have

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{N}\mathcal{C}_{t,j}(\mathcal{O},\mathcal{M}_{corr})\sum_{m=1}^{M} c_{j,m}b_{j,m}(\boldsymbol{o}_t)\frac{(o_{t,d} - \mu_{j,m,d})^2}{\sigma^2_{j,m,d}} = 1.0 \tag{7.13}$$

The above expression follows from the Baum-Welch re-estimation formulae for the distribution parameters (equations 2.16, 2.18). The above equality will be used to establish the relationship of the MMIC algorithm and the feature selection scheme proposed in [17] (see section 7.6).

7.5 Discussion

The MMIC algorithm does not make any assumptions about the level of modelling. Thus, it is applicable to both word units and sub-word units. The selection procedure does not rely on appropriate class definitions - these are implicitly incorporated into the recognition model used to calculate the denominator likelihood. The use of the task-specific recognition model allows the propagation of the grammar constraints into the selection process. If the Viterbi state/mixture path approximates the overall likelihood well, the first derivative approximation to the value of the MMIE objective function will be exact.

One uncertainty about the algorithm lies in the crude first derivative approximation to the value of the MMIE objective function. Indeed, from numerical analysis [90], even the quadratic approximation to the value of the function becomes grossly inaccurate at points far from the desired solution. Unfortunately, more accurate approximation will require higher-order derivatives, which, in the HMM framework are prohibitively expensive to compute. The second limitation of the algorithm is the fact that in the parameter reduction stage, all components of the feature vector are considered independent. As soon as we discard a single component, the frame/state allocation is likely to change and thus the derivatives of the remaining components will also change. Ideally, feature selection should be carried out one feature at a time, with re-evaluation of the derivative expressions after every reduction. Unfortunately, the MMIC feature selection algorithm is not computationally cheap, due to the potentially large number of model states that need to be considered in order to evaluate derivatives in the denominator of the objective function. The above two problems became apparent in the initial set of experiments carried out. All derivatives of the MMIE objective function with respect to the feature weights were found to be negative which indicated that by deleting a parameter, the value of the mutual information measure would actually increase. Since this is the objective in MMIE training, the top $\bar{D}$ features with largest derivatives were selected. The performance of the system was dramatically impaired which was to the contrary of what the derivatives predicted. This prompted an investigation into the gradual reduction of the number of features, still deleting the feature with the largest weight derivative at each step. Indeed, removing the feature with the largest weight derivative exhibited an improvement in the mutual information criterion, however, subsequent evaluation of the parameter derivatives produced a severely altered feature rank-order.

7.6 Relationship to other methods

In this section we consider the discriminative feature selection method proposed by Bocchieri and Wilpon in [17], which was briefly outlined in chapter 6. In this method, the components of the feature vectors are rank-ordered according to the measure

$$J_d = \frac{\mathcal{D}_d(\mathcal{M}_{rec})}{\mathcal{D}_d(\mathcal{M}_{corr})} \tag{7.14}$$

where

$$\mathcal{D}_d(\mathcal{M}) = \frac{1}{T}\sum_{t=1}^{T}\left\{\frac{(o_{t,d} - \mu_{\theta_t,\psi_t,d})^2}{\sigma^2_{\theta_t,\psi_t,d}}\right\} \qquad \text{for } d = 1,\ldots,D \tag{7.15}$$

From the above, the method appears to use similar statistics to the MMIC algorithm, gathered in a different way - only the average scaled distances are considered, the variance weighting terms are ignored. Disregarding the latter discrepancy, the numerator/denominator in equation 7.14 are somewhat equivalent to computing the derivative of the denominator/numerator likelihoods respectively in the MMIE objective function with respect to u_d. In the practical implementation of the algorithm, the following simplifications are made. The statistics $\mathcal{D}_d(\mathcal{M}_{corr})$ are gathered over the most likely (Viterbi) state/mixture sequence. Furthermore, the Viterbi alignment is also used in the MLE procedure to optimise the parameters of the HMMs, hence, the denominator in equation 7.14 is assumed to be 1.0. This assumption is valid by the Maximum Likelihood training criterion as discussed in section 7.5. Consequently, the Bocchieri and Wilpon feature selection algorithm is solely based on values of $\mathcal{D}_d(\mathcal{M}_{rec})$. As for the denominator $\mathcal{D}_d(\mathcal{M}_{corr})$, the statistics gathered from the recognition model are computed using Viterbi alignment e.g. only the most likely path through the model is considered. Furthermore, only misaligned frames corresponding to substitution and insertion errors are taken into account. Finally, feature selection is performed by discarding features with small J_d. The authors argued that the components with

rank	feature	change	rank	feature	change
1	ΔE	0.0529	21	$\Delta^2 C_3$	0.0212
2	C_1	0.0500	22	C_9	0.0199
3	C_3	0.0478	23	ΔC_5	0.0187
4	ΔC_1	0.0454	24	$\Delta^2 C_7$	0.0181
5	$\Delta^2 E$	0.0422	25	$\Delta^2 C_4$	0.0176
6	C_2	0.0381	26	C_{10}	0.0174
7	E	0.0371	27	ΔC_8	0.0169
8	C_5	0.0353	28	$\Delta^2 C_6$	0.0168
9	ΔC_2	0.0348	29	$\Delta^2 C_8$	0.0162
10	C_4	0.0342	30	C_{11}	0.0162
11	C_7	0.0310	31	ΔC_9	0.0157
12	C_6	0.0290	32	$\Delta^2 C_{10}$	0.0150
13	ΔC_4	0.0290	33	C_{12}	0.0142
14	$\Delta^2 C_1$	0.0280	34	$\Delta^2 C_9$	0.0140
15	$\Delta^2 C_2$	0.0276	35	$\Delta^2 C_{12}$	0.0138
16	ΔC_3	0.0257	36	$\Delta^2 C_{11}$	0.0136
17	ΔC_6	0.0244	37	ΔC_{11}	0.0126
18	$\Delta^2 C_5$	0.0229	38	ΔC_{10}	0.0112
19	ΔC_7	0.0224	39	ΔC_{12}	0.0105
20	C_8	0.0217			

Table 7.1: Feature ordering according to the MMIC feature selection algorithm applied to the TIMIT data.

largest ratios give the highest contribution to the average distance between the "incorrect" HMM and the misaligned speech frames. Hence, intuitively such features will provide a better between-class separation.

The MMIC algorithm described in the previous section, automatically accommodates the Bocchieri & Wilpon discriminative feature selection scheme as a special case. Its main advantage lies in its theoretical derivation from the Mutual Information criterion and its ability to consider multiple confusable paths/mixture components in the feature selection procedure.

7.7 Implementation

The algorithm was implemented in the MMIE discriminative training framework outlined in section 4.5. The tool (`HNRest`) was extended to accommodate routines which calculate the derivatives of the likelihood function with respect to the exponent vector $\boldsymbol{u}$. The computation of derivatives is carried out in two passes - one for the numerator and another one for the denominator likelihoods. In a postprocessing stage the two sets of derivatives

Type	% Corr	% Sub	% Del	% Ins	% Acc
(13) `MFCC_E`	59.93	23.74	16.33	3.38	56.55
(26) `MFCC_E_D`	70.59	19.67	9.74	4.07	66.51
(39) `MFCC_E_D_A`	73.71	19.63	6.67	5.02	68.69

Table 7.2: Baseline results for 24 mixture MLE trained models (TIMIT)

are combined to calculate the selection criterion for each component of the original feature vector. The actual feature selection is carried out using a selection template in the form of a matrix with $\{0, 1\}$ elements. Once the template is constructed, it is used to transform the mean and variance vectors in all models. During subsequent recognition and training, the selection template is applied on-the-fly to the original data to produce the desired subset of features which are then seen by the output distributions of the HMMs.

7.8 Experimental evaluation

In this section we shall describe the MMIC feature selection experiments applied to continuous phone recognition on the TIMIT database. We start with the original $D = 39$ `MFCC_E_D_A` feature set. Using the MMIC feature selection algorithm the dimensionality is then reduced to $\bar{D}$ features and the performance of the system is evaluated on the test data. The baseline performance of the fundamental subsets of the original parameter sets are provided in table 7.2 to serve as reference points. The system uses the TIMIT monophone model set described in appendix A. The full training set of 3696 utterances was used to estimate the parameters of the model according to the MLE criterion using the embedded Baum-Welch algorithm. The 24 mixture Gaussian system was built from a single mixture system by gradually incrementing the number of mixture components for all states in the models.

The system was evaluated on the suggested TIMIT core test set of 192 sentences. During recognition, a phone bigram language model was applied with language model weight (exponent) of 2.0. The performance of 68.69% accuracy is very good for a system based on context independent HMMs. For example, Bocchieri & Wilpon [17] have reported 59.8% recognition accuracy using monophone HMMs with 16 mixture components using LPC-cepstral parameters and their first and second differentials.

For the feature selection experiments, the full training set was used to compute the approximate change in mutual information for each parameter in the observation vector. The ordering of features according to the MMIC criterion is given in table 7.1. Although it is not intuitively clear what the feature ordering represents we can offer the following interpretation. The three energy components appear high in the list which confirms common knowledge regarding their importance in speech recognition [93, 66]. The first 13 parameters also include the first 7 cepstral coefficients together with 3 of their first derivatives. The least important 10 entries in the table include the first and second derivatives of the

$\bar{D}$	% Corr	% Sub	% Del	% Ins	% Acc
(38)	73.89	19.39	6.72	4.98	68.91
(34)	73.17	19.92	6.92	5.11	68.05
(30)	72.58	20.11	7.30	4.93	67.65
(26)	72.16	20.36	7.48	5.03	67.12
(22)	71.30	20.60	8.11	4.75	66.54
(18)	69.58	21.73	8.69	4.16	65.42
(16)	69.02	21.83	9.15	4.16	64.86
(13)	66.54	23.02	10.44	3.80	62.74

Table 7.3: Performance of feature subsets selected using the MMIC feature selection algorithm.

last 4 coefficients, together with C_{11} and C_{12} themselves. The somewhat surprising fact is that in some cases, the second derivatives of a parameter appears as more important than its first derivative $\{C_5, C_{10}, C_{11}, C_{12}\}$. Table 7.3 shows the recognition performance of the system for a different number of selected features from the original `MFCC_E_D_A` observation vector. The plot in figure 7.1 shows the gradual decrease in performance as more and more features are removed. In [17], the authors managed to reduce the feature vector size from 38 components down to 18 with only a marginal decrease in the recognition accuracy. In our case, the decrease in performance over the same range amounts to 4%, which can be explained by the use of a possibly more "optimal" parametrisation (`MFCC_E_D_A`) and acoustic distributions with a large number of mixture components. The usefulness of the algorithm becomes apparent when comparing the performance of the fundamental `MFCC_E_D_A` subsets to the identically sized reduced feature sets in table 7.3. The performance figures of 72.16% and 66.54% correct provided by the MMIC reduced sets of size 26 and 13 features respectively, provide improved performance over the figures of 70.59% and 59.93% when using the `MFCC_E_D` and `MFCC_E` parameter sets respectively.

In chapter 6 we outlined a manual feature selection method based on the power spectrum resolution. This method has been used by Paliwal in his study of feature reduction methods [87]. The author also points out that the advantage of using this method lies in one's ability to justify the use of these feature sets on physical grounds, hence they can be expected to perform equally well on other tasks. In order to compare the performance of the MMIC feature selection method, we have devised a number of reduced feature sets. Their definition is as follows

$$\begin{aligned}
R(26) &= C_{1-10}, E, \Delta C_{1-8}, \Delta E, \Delta^2 C_{1-5}, \Delta^2 E \\
R(22) &= C_{1-9}, E, \Delta C_{1-7}, \Delta E, \Delta^2 C_{1-3}, \Delta^2 E \\
R(18) &= C_{1-8}, E, \Delta C_{1-4}, \Delta E, \Delta^2 C_{1-3}, \Delta^2 E \\
R(16) &= C_{1-8}, E, \Delta C_{1-3}, \Delta E, \Delta^2 C_{1-2}, \Delta^2 E
\end{aligned}$$

These manually derived feature sets were evaluated on the test data and the performance

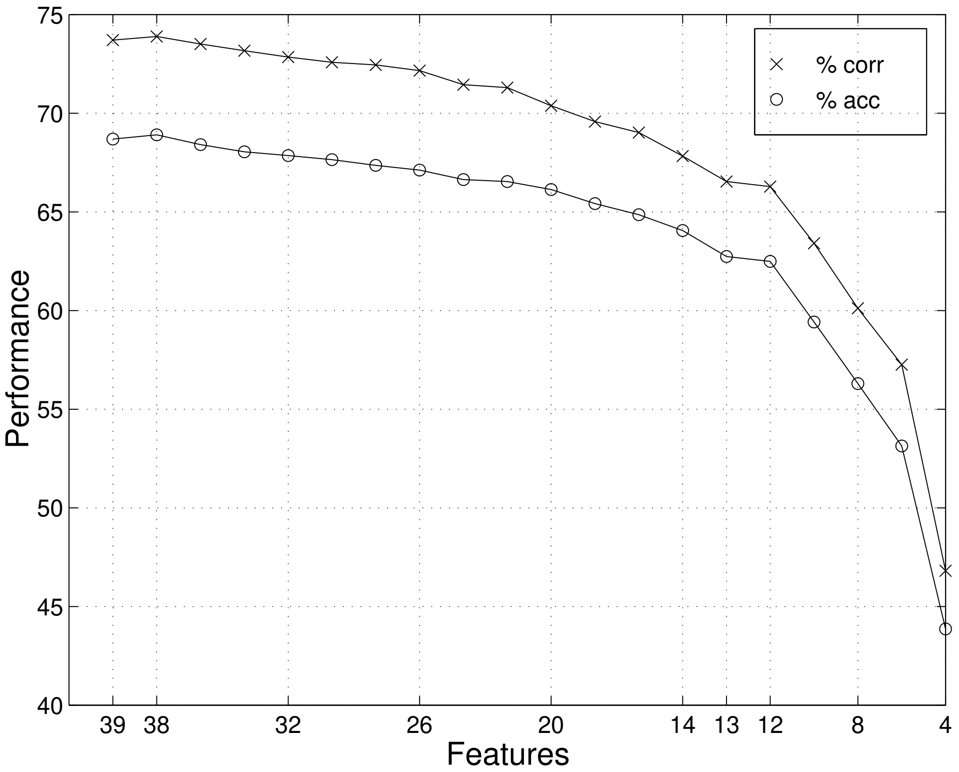

Figure 7.1: Recognition performance for different subset of the original features. Consecutive points on the graph represent a difference of two features, except for points explicitly labelled.

figures are given in table 7.4. It can be seen from these results that these feature sets perform marginally better than the corresponding MMIC feature sets. This observation is similar to findings in [87] where the manually selected feature sets gave the best performance.

Another explanation for these somewhat disappointing results is offered from the fact that in the MMIC feature selection algorithm each component in the feature vector is considered independently. Unfortunately, even for the cepstral coefficients this is not a valid assumption. Figure 7.2 shows the correlation matrix for the components of the `MFCC_E_D_A` feature vector. The two "valleys" in the corners of the diagram show correlations between the static cepstral coefficients and their second derivatives. Also, the three peaks on each side of the diagonal indicate correlations between the energy components $\{E, \Delta E, \Delta^2 E\}$ and the first cepstral coefficient C_1. These observations suggest that by taking cepstral parameter correlations into account, the MMIC selection scheme may provide improved feature subsets.

Type	% Corr	% Sub	% Del	% Ins	% Acc
$R(26)$	72.31	20.22	7.47	4.95	67.36
$R(22)$	71.38	20.75	7.87	4.67	66.71
$R(18)$	69.77	21.80	8.43	4.39	65.38
$R(16)$	69.36	21.69	8.95	4.09	65.27

Table 7.4: TIMIT results for manually derived feature sets, 24 mixture models

Figure 7.2: Correlation matrix for the components of the `MFCC_E_D_A` feature set calculated on the TIMIT database (3696 utterances)

7.9 Summary

This chapter has described a feature selection algorithm based on the mutual information criterion. A first derivative approximation of the changes in the mutual information measure are calculated for each feature vector component. The parameters are then sorted in increasing value order and the desired top $\bar{D}$ features are selected to form the new feature set. The algorithm achieves the desired goal of reducing the size of the observation vectors with gradual and modest degradation in performance. Unfortunately MMIC did not provide an improvement over the corresponding manually selected feature sets. This can be attributed to the various approximations made within the algorithm, in particular, the lack of modelling the correlations between the components of the feature vectors.

Chapter 8

Adaptive Input Transformations

Chapter 6 described several approaches to improving the discriminant abilities of hidden Markov models through the introduction of input transformations. One problem with these methods is that the criterion used to derive the transformations is not related to the objective function used to optimise the HMM parameters. Furthermore, once derived, these transformations remain unchanged throughout the training process. In this chapter we shall describe methods for integrating adaptive input transformations into existing HMM systems. During training, the parameters of the transformations are optimised jointly with the HMM parameters according to the MMIE objective function. Experimental evaluation of different HMM/input transformation (HMM/IT) topologies will be presented on the TIMIT continuous phone recognition task.

8.1 Introduction

As discussed in chapter 6, feature reduction methods apply one or many input transformations to the input data and a subset of the transformed output is then used as an observation vector for the HMMs. Three such transformations were discussed - the PCA transformation, the LDA transformation and the state-specific discriminative input transformations used by Woodland et al. [109] and Doddington [32]. The LDA transformation was shown to be optimal in the sense of optimising the J_1 measure of class separability which is based on the ratio of within-class and between-class scatter measures. Apart from the requirement for appropriate class definitions, the LDA transformation relies on the assumptions that the covariance matrices of all classes are identical and that the class centroids are normally distributed around the mean of the data. Unfortunately, these conditions are never satisfied in practice. Ayer [4] has carried out a set of simple experiments to verify the validity of the LDA assumptions. He analysed the shape and orientation of a number of within-class covariance matrices corresponding to intuitively different classes. The analysis used the largest eigenvalue and the ratio of the largest to the second largest eigenvalue to judge the shape of the distributions. The two different classes investigated were found to have significantly different shapes, and the variability within the class of voiceless fricatives {f,s} was found to be larger than the variability within the steady state vowel class {i:(B),i:(E)}[1]. The identical class covariance matrix assumption can be avoided by considering state-specific input transformations, however, this approach results in a vast number of parameters with computational savings available only if the output dimensionality is reduced by more than a half.

Input transformations can be considered as mapping patterns from the original feature space into patterns in the reduced subspace. As such, they resemble artificial neural networks commonly used as pattern classifiers. As it will be shown later, this correspondence will enable us to the use the back-propagation training algorithm from the field of neural networks, to derive a global optimisation scheme for the parameters of both the HMMs and the input transformations. The following section briefly outlines the theory of neural networks and their application in the field of speech recognition.

8.2 Background

8.2.1 Neural networks for speech recognition

An artificial neural network (ANN), also referred to as connectionist model, multi-layer perceptron (MLP) or parallel distributed processing (PDP) model, is a structure based on connecting many simple processing elements. These models are the outcome of the so called "holistic" approach to modelling human intelligence where the computational power of the brain is modelled by many simple processing elements. This approach dates back to the early 1960's. The recent revival of interest in the field has been prompted by the development of new learning algorithms and the advances in computer technology required to carry out experimental simulations. Since the mid 1980's, ANNs have been applied to a diverse selection of problems, including pattern classification and continuous speech recognition. The advantages of these models can be summarised as follows:

1. Parallelism. Neural networks implement a high degree of parallel computation using many simple, identical computational units. Hence, their implementation in hardware can provide huge performance benefits at low production cost;

2. Robustness and fault tolerance. Being massively parallel and having the information distributed to every computational element makes these networks highly insensitive to faults in the structure;

3. Adaptive learning. The structure and the parameters of the network can be adapted on-the-fly to achieve better performance;

4. Powerful approximations. A multi-layered network with non-linear computational elements can approximate, arbitrarily closely, any non-linear mapping.

[1]The members of this class are the vowel segments of two utterances corresponding to the letters "B" and "E" from the BTL E-set database.

Conventional ANN's are structured to deal with static patterns and the network parameters are typically estimated from pairs of input output patterns. As such, ANNs have been successful for characterising speech segments of limited duration [108]. However, speech production is inherently a time varying process and there is no established way of handling dynamic patterns in connectionist models. Several researchers have adapted the basic ANN's structure in order to enhance its ability to model time variant patterns. Waibel et al. have proposed the Time Delay Neural Network (TDNN) which incorporates basic speech pattern dynamics by expanding the input of the network to N adjacent frames. An alternative ANN structure for dealing with dynamic patterns has been proposed by Robinson [97]. The network uses unit time-delay recurrent connections to propagate a subset of the current network output back into the network input for the following time frame.

Various attempts have been made to assign an interpretation to the outputs of ANN classifier. Bridle [19] attempted to give an explicit probabilistic interpretation of the network outputs through probability scoring and a normalised exponential non-linear output function. Early work by Robinson et al. [99] used a dynamic programming based postprocessor with stochastic duration and bigram language models to perform continuous phone recognition using the output of a recurrent ANN. With the introduction of discriminative training in the HMM framework, several researchers have attempted to accommodate the HMM theory into the framework of ANNs. The Alphanet approach [18] is an example of a scheme which attempts to unify HMMs and adaptive neural networks. The idea is to view the forward likelihood calculation of the data as the forward pass in a large recurrent network of a special kind that produces sets of numbers which can be treated as posterior probabilities for the classes of interest. Partial derivatives of the chosen error criterion can be propagated back through the network. At the end of the backward pass accumulated partial derivatives can be used to re-estimate the parameters of the network so as to decrease the error function. Similar schemes have been studied by Young [111] and Niles et al. [83]. These approaches are significant in that they establish the relationship between HMMs and neural networks.

8.2.2 ANNs as input transformations - related work

There have been several attempts to combine the discriminative power of ANN's with the explicit time handling capabilities of HMMs. The most common approach is to use the outputs of the network as observation probabilities for the states of the Markov chain [95], [77]. The ANN is trained to compute the appropriate values according to the Least Mean Square (LMS) criterion using the error back-propagation algorithm. The LMS criterion is minimised over the frame/state allocation provided by the supervised (forced) Viterbi alignment. This approach has also been used by Robinson [98] for continuous phoneme recognition, using a recurrent neural network as a probability estimator.

A slightly different hybrid approach was discussed by Bengio et al. [14] where an ANN was used to compute an additional set of symbols considered as observation parameters in a discrete HMM framework. In this approach a vector-quantised code-book was generated for the extra observation symbols and added to the existing code-books for conventional parameters. The parameters of the HMMs and the ANN were optimised separately. Cardin et al. investigated a similar approach in [23] where a recurrent neural network was used to perform broad phonetic classification on input frames. The four outputs of the network and their first differentials were quantised using two separate code-books and added to the existing three code-books for standard cepstral coefficients.

Joint optimisation of HMMs and input transformations was suggested by Bridle [20] within the Alphanet framework. He argued that although cepstral coefficients offer a reasonable representation, there is reason to expect that non-linear transformations might be more sensitive to movements of spectrum peaks. No suggestions were given regarding the derivation of such non-linear transformations. In a later study [21] Bridle et al. presented a preliminary evaluation of joint optimisation of the HMM parameters and the weights of a single input transformation. The transformation was derived from LDA and was strictly linear. Furthermore, the HMM parameters were optimised using MLE, whilst the parameters of the transformation were adapted according to the MMIE criterion using gradient descent. The system was evaluated on a single speaker continuous phoneme recognition task with a total of 18 minutes of training data and 5 minutes of test data. Adapting the transformation did not improve recognition performance.

Bengio et al. [15] also investigated joint global optimisation of HMMs and ANNs applied to the recognition of plosive sounds on the TIMIT database. The system made use of three neural networks and 11 continuous density HMMs with 5 mixture components per state. Two of the networks were initially trained to perform plosive recognition and broad phonetic class classification. The third network was deterministically initialised to compute the principal components of the concatenated output vectors from the first two networks. The network output of 8 parameters formed the observation vectors for the HMMS. When the parameters of the ANNs and the HMMs were optimised separately, the system achieved recognition accuracy of 75%. Simultaneous optimisation of all parameters in the system according to the MMIE criterion resulted in improved recognition accuracy of 86%.

Another interesting approach is the Whole-word Adaptive LDA (WALDA) transformation proposed by Ayer [5]. A single global input transformation was derived using LDA. The parameters of the transformation were then optimised according to a discriminative objective function using gradient descent. The transformation was applied to the speaker independent recognition on the BTL E-set database. The result of 96% accuracy represents the best performance figure published up to date on that database. The training of the transformation required 50-100 iterations of gradient descent in order to achieve the desired results.

Most recently, Johansen et al. [57] have investigated the use of non-linear input transformations in the framework of continuous density HMMs. A single global transformation was initialised with small random weights and the current feature vector was added to the output of the transformation. The system which was globally optimised according to the MMIE criterion achieved 22% fewer errors on a TIMIT broad-class phone recognition task over the baseline MMIE-trained HMM system.

8.3 Integrating input transformations

The hybrid approaches discussed in the preceding section provide an elegant way of integrating the discriminative power of ANNs with the time handling structure of HMMs. Although the idea of joint optimisation of HMMs and input transformations parameters according to a discriminative objective function is not new, the empirical evidence in favour of its applicability and success is very limited. The following sections will describe a general framework for integrating input transformations into existing HMM systems. Part of our objective in this work was to investigate the benefits of sharing input transformations. The system we have implemented therefore allows input transformations to be *state-specific* (no sharing), *model-specific* (single transformation shared by all states of a model) or *global* (single transformation shared by all states of all models). In fact, the system has been developed as an extension of the HTK package described in [117, 112] and allows arbitrary tying of input transformations in common with all of the other HMM parameters.

8.3.1 General theory

In a conventional HMM, the probability of an observation $\hat{\mathbf{o}}_t$ given a continuous mixture Gaussian output distribution is given by

$$b_j(\hat{\mathbf{o}}_t) = \sum_{m=1}^{M} c_{j,m} \frac{1}{(2\pi)^{D/2}|\boldsymbol{W}_{j,m}|^{1/2}} e^{-\frac{1}{2}(\hat{\mathbf{o}}_t - \boldsymbol{\mu}_{j,m})' \boldsymbol{W}_{j,m}^{-1}(\hat{\mathbf{o}}_t - \boldsymbol{\mu}_{j,m})} \tag{8.1}$$

where $\boldsymbol{\mu}_{j,m}$ is the mean vector and $\boldsymbol{W}_{j,m}$ is the covariance matrix for the m^{th} mixture component at HMM state j. In general, we can assume that the observation vector $\hat{\mathbf{o}}_t$ seen by the mixture distribution is the output of a state-specific transformation function $\mathcal{T}$ applied to the output from the acoustic pre-processor $\boldsymbol{o}_t$[2], e.g.

$$\hat{\mathbf{o}}_t = \mathcal{T}(\boldsymbol{o}_t)$$

A generic input transformation is defined as a sequence of matrix multiplications with the intermediate resulting vectors mapped using pre-defined output functions

$$\mathcal{T}(\boldsymbol{o}) = f_N(\boldsymbol{U}'_N f_{N-1}(\boldsymbol{U}'_{N-1} \dots f_1(\boldsymbol{U}'_1 \boldsymbol{o}) \dots)) \tag{8.2}$$

The above transformation can be viewed as an N-layer neural network where the units in layer l are set to compute $f_l(.)$ and the weights of layer l are the values in $\boldsymbol{U}_l$. The functions $f_l(.)$ are chosen to be differentiable vector valued functions. Two such functions are

1. *Linear*

$$\boldsymbol{y}_l = f_l(\boldsymbol{x}_l) \qquad \text{where} \qquad \boldsymbol{y}_l = \boldsymbol{x}_l \tag{8.3}$$

[2]Although the framework allows transformations to be state-specific, in most of the derivations in this chapter a single global input transformation is assumed. However, for the derivations of the re-estimation formulae, the notation $\hat{\mathbf{o}}_{t,j}$ will be used to denote the observation vector at time t, produced by the transformation associated with state j.

2. *Sigmoid* $\{-1.0; +1.0\}$

$$\boldsymbol{y}_l = f_l(\boldsymbol{x}_l) \qquad \text{where} \qquad y_{l,i} = \frac{2}{1+e^{-2x_{l,i}}} - 1 \tag{8.4}$$

In general, we shall use $\boldsymbol{x}_l$ to denote the vector of total input to the nodes at layer l and $\boldsymbol{y}_l$ will denote the output from the nodes at l following the application of the output function. We have

$$\boldsymbol{x}_l = \boldsymbol{U}_l' \, \boldsymbol{y}_{l-1} \qquad \text{and} \qquad \boldsymbol{y}_l = f_l(\boldsymbol{x}_l) \tag{8.5}$$

where $\boldsymbol{y}_1 = \boldsymbol{o}_t$. In the above notation, bias vectors are assumed to be implicitly incorporated into the weight matrices $\boldsymbol{U}_l$. In the context of neural networks, equation 8.2 defines the forward pass for a feed-forward network. A feed-forward network is one where the topology is such that values of $\boldsymbol{x}_l$ and $\boldsymbol{y}_l$ can be calculated in an order that does not lead to recursion. The following section will describe the structure of a recurrent input transformation.

8.3.2 Recurrent transformations

A major limitation of the hidden Markov modelling approach to automatic speech recognition tasks is the so called *observation independence assumption*

$$P(\boldsymbol{o}_t|\theta_1^T, \boldsymbol{o}_1^T) = P(\boldsymbol{o}_t|\theta_t) \tag{8.6}$$

This states that the probability of an acoustic observation $\boldsymbol{o}_t$ at time t depends only on the output distribution associated with the present state θ_t of the Markov chain but not on the other observations. The problem with systems of this kind is that slowly varying articulatory processes introduce significantly larger amounts of long-term correlation which cannot be modelled adequately by the state transition probabilities alone. As discussed in previous chapters, modelling of acoustic signal dynamics can be improved by adding new dimensions to the observation vectors. Unfortunately, increasing the number of features in the observation vectors introduces more parameters in the output distributions of the HMMs which, in turn, will require more training data and the associated computational effort will be increased. A partial solution to this problem is offered by the dimensionality reduction methods discussed in chapter 6. A more general approach is to enhance the transformation by incorporating a recurrent mechanism which allows the present output $\hat{\mathbf{o}}_t$ to incorporate knowledge of past and future observations. Due to the complex nature of the mapping and the lack of knowledge as to which types of correlations are important, the parameters of such transformation cannot be determined directly. In this section we introduce the concept of adaptive recurrent input transformations in the HMM framework. The transformations will be initialised to performing non-recurrent mappings, however, during training the discriminative training algorithm will be used to establish the recurrent mechanism in the transformation.

Recurrent processing structures were extensively studied by Robinson [97] in an attempt to model dynamic patterns by a sequential information processing system. The essential quality of a dynamic network is that its behaviour is determined both by the external

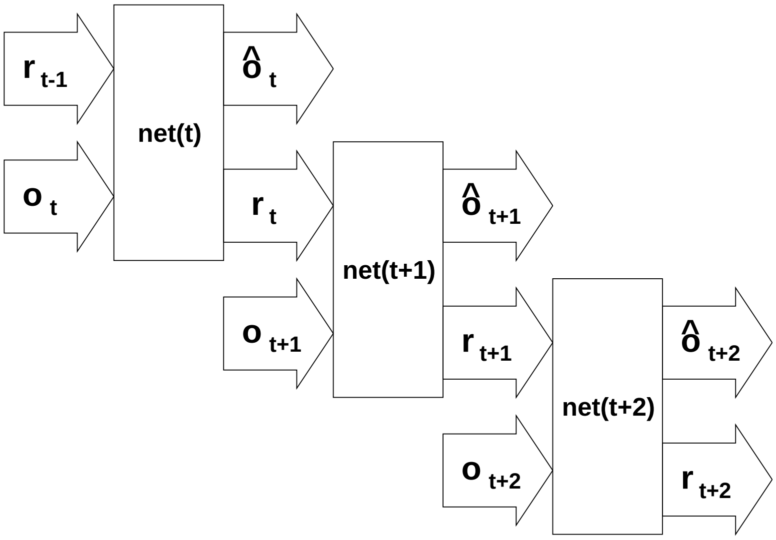

Figure 8.1: A dynamic network structure expanded in time where $\boldsymbol{o}_t$ is the input pattern to the network at time t, $\hat{\mathbf{o}}_t$ is the corresponding output and $\boldsymbol{r}_t$ is the output of recurrent state units.

input to the network and also by its internal state, which holds information about previous mappings. The state units form part of the current output of the network and also present themselves as part of the input to another copy of the network in the following time period. Another interpretation of the state units is that they link multiple copies of the network over time to form a dynamic sequential structure, see figure 8.1.

Using the above, a recurrent mechanism can be incorporated into 8.2 by allowing the argument of f_l to carry information about past states of the transform. In the following sections we consider single layer recurrent transformations with state units whose structure is shown in figure 8.2. The output of the transformation can be expressed as

$$\hat{\mathbf{o}}_t = f_O(\boldsymbol{U}'_{I,O}\boldsymbol{o}_t + \boldsymbol{U}'_{R,O}\boldsymbol{r}_{t-1}) \qquad (8.7)$$

$$\boldsymbol{r}_t = f_R(\boldsymbol{U}'_{I,R}\boldsymbol{o}_t + \boldsymbol{U}'_{R,R}\boldsymbol{r}_{t-1}) \qquad (8.8)$$

where $\boldsymbol{r}$ is the output vector from the state units, t is the time index and $\boldsymbol{U}_{I,O}$, $\boldsymbol{U}_{I,R}$, $\boldsymbol{U}_{R,R}$, $\boldsymbol{U}_{R,O}$ are the matrices describing the *input-to-output*, *input-to-state*, *state-to-state* and *state-to-output* connections respectively. The functions f_O and f_R are chosen to compute symmetric sigmoids (equation 8.4).

8.4 Initialisation

Several different approaches can be taken to initialising the input transformations. ANNs are commonly initialised with random weights, however, with random initialisation the training of the system will be rather slow since the parameters of the HMMs depend on the

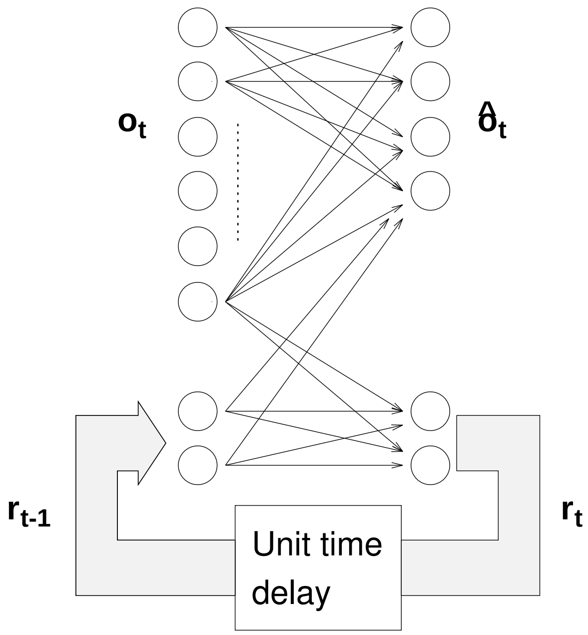

Figure 8.2: General structure of a recurrent input transformation where $\boldsymbol{o}$ is the output of the preprocessor, $\hat{\mathbf{o}}$ is the output of the transform, $\boldsymbol{r}$ is the output of the state units and t is the time index. (note: not all connections are drawn for clarity of presentation)

outputs of the transformations. Another possibility is to initialise the parameters of the transformation deterministically [15]. Such initialisation is highly desirable for the following reasons:

1. The parameter estimation does not depend on random initial conditions. Kolen et al. [63] have shown how different initial conditions can give rise to different neural networks with rather different performances.

2. Deterministic initialisation will allow input transformations to be integrated into an existing HMM system without any significant degradation in performance.

The HMM framework discussed so far offers two possible ways for deterministic initialisation of input transformation: 1) Initialisation based on rotations/scaling of the the observation feature space, and 2) Initialisation based on differential parameter computation. The following two sections will describe these two methods in detail.

8.4.1 Rotation/scaling of the feature space

In this approach, the input transformations can be initialised using a linear transformation matrix as provided by PCA, LDA or the method of principal discriminants [22]. Since we attempt to provide a better acoustic model with improved discrimination it seems plausible to initialise the transformations to perform linear discriminant analysis LDA on the feature vectors. As discussed in chapter 6, LDA makes use of two covariance matrices - the between-class covariance matrix $\boldsymbol{B}$ and the *average* within-class covariance matrix $\boldsymbol{W}$. The discussed HMM/IT framework allows input transformation to be state-specific. Hence, LDA should be performed at a level which reflects the specificity of the desired transformations. For example, if state-specific input transformations are desired, LDA will be performed separately for each state where $\boldsymbol{B}$ is still the overall between-class covariance matrix and $\boldsymbol{W}_j$ is the average covariance matrix from the distribution at state j. The between class covariance matrix is computed by

$$\boldsymbol{B} = \frac{1}{n}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_g)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_g)' \tag{8.9}$$

where n is the number of classes, $\boldsymbol{\mu}_i$ is the mean vector of the distribution associated with class i and $\boldsymbol{\mu}_g$ is the mean vector of the data. The use of sample average instead of proper expectation provides immunity to biasing $\boldsymbol{B}$ towards classes with more training data. The within-class covariance matrices $\boldsymbol{W}_j$ can be calculated as a by-product of the Baum-Welch algorithm hence, we proceed to build a full covariance HMM system where covariance matrices are tied to match the desired level of tying for the subsequently introduced input transformations. For example, if the eventual goal is to produce an HMM system with a single global input transformation, we set off to train a grand covariance HMM system. At the other end of the extreme, we can train a full covariance HMM system where each state distribution has its own distinct covariance matrix. The latter scheme will enable us to construct a final system with state-specific input transformations.

Assuming single Gaussian distributions with individual covariance matrices $\boldsymbol{W}_j$ and a grand covariance matrix $\boldsymbol{B}$, the required transformation matrix $\boldsymbol{S}_j$ is computed such that

$$\boldsymbol{S}_j'\boldsymbol{W}_j\boldsymbol{S}_j = \boldsymbol{I} \qquad \text{and} \qquad \boldsymbol{S}_j'\boldsymbol{B}\boldsymbol{S}_j = \boldsymbol{\Gamma} \tag{8.10}$$

where $\boldsymbol{\Gamma}$ is diagonal (see section 6.5.2). Solving the general eigen problem, $\boldsymbol{S}_j$ can be expressed as

$$\boldsymbol{S}_j = \boldsymbol{R}_1\boldsymbol{\Lambda}^{-1/2}\boldsymbol{R}_2 \tag{8.11}$$

where $\boldsymbol{R}_1$ is the matrix of eigenvectors of $\boldsymbol{W}_j$, $\boldsymbol{\Lambda}$ is the diagonal matrix of corresponding eigenvalues and $\boldsymbol{R}_2$ is the matrix of eigenvectors of $(\boldsymbol{R}_1\boldsymbol{\Lambda}^{-1/2})'\boldsymbol{B}(\boldsymbol{R}_1\boldsymbol{\Lambda}^{-1/2})$. Improved discrimination in a subspace of the original observation space can be achieved by discarding dimensions corresponding to the smallest elements of $\boldsymbol{\Gamma}$.

8.4.2 Stacked input

The observation vectors in most current HMM-based speech recognition systems are based on log-power spectrum representations together with their first and second derivatives. The earliest attempt to use differential information was discussed in [79] by researchers at IBM. Differential information was incorporated by concatenating adjacent frames together, much like the windowed input presented to ANN based speech classifiers. This dramatically increased the numbers of parameters and also caused estimation problems. The solution was to use principal components analysis to reduce the observation vector size. More recently, Brown [22] used linear discriminants to reduce the size of stacked observation vectors for an

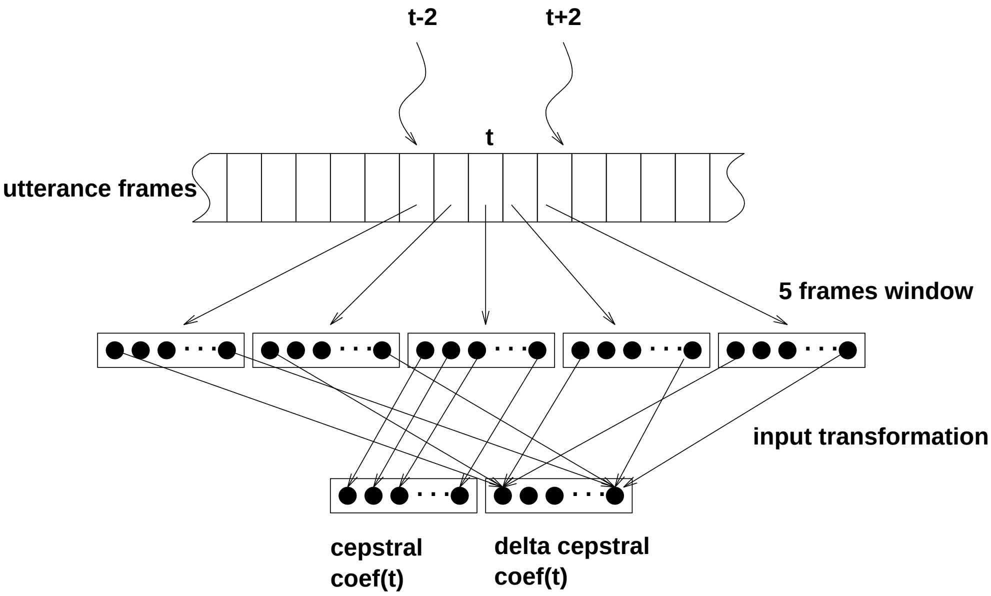

Figure 8.3: Input transformation initialised to calculate differential cepstral parameters over using an input window of 5 frames.

E-set recognition task. One problem with differential parameters as used in most current speech recognition systems is the assumption that the cepstral slope is the only useful feature. The stacking of input frames does not make such an assumption, hence, with an appropriate feature selection algorithm it may result in a more compact and informative set of parameters. The combined HMM/input transformation architecture provides a powerful mechanism for combining heterogeneous sets of features and extracting a small number of parameters according to a discriminative objective function. A deterministic initialisation of such input transformations is offered by the formulae used to compute the first derivative parameters

$$\Delta C_i(t) = \frac{\sum_{\tau=1}^{V} \tau(C_i(t+\tau) - C_i(t-\tau))}{2\sum_{\tau=1}^{V} \tau^2} \tag{8.12}$$

where V is typically set to 2. The above equation can be expressed as a transformation matrix $\boldsymbol{A}$ applied to an input vector formed by concatenating five adjacent observation frames from the utterance, see figure 8.3.

8.4.3 Non-linearity and 2-layer transformations

Assuming that we have the transformation matrices $\boldsymbol{A}$ and $\boldsymbol{S}_j$ derived using the above schemes, we can proceed to initialise the weight matrices of the input transformations. Based on the properties of the input transformation, we consider the following cases

1. *Single-layer, linear.* The derived transformation matrix $\boldsymbol{S}_j$ becomes the transformation matrix $\boldsymbol{U}_1$.

2. *Single-layer, non-linear.* Following Bengio [15], the connections of the transformation can be initialised with $\boldsymbol{U} = \epsilon \boldsymbol{S}_j$ where ϵ is a small positive number. Consequently, the total input to the output units of the network will be small and the sigmoid functions will operate within a linear range.

3. *Two-layer, non-linear (1).* The derived matrix $\boldsymbol{S}_j$ can be decomposed into the product of two matrices e.g. $\boldsymbol{S}_j = \boldsymbol{P}\boldsymbol{Q}$, using LU decomposition [90]. This decomposition can be used to initialise a two layer network with connection matrices $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ describing the input-to-hidden and hidden-to-output connections respectively. Then, $\boldsymbol{U}_1 = \epsilon \boldsymbol{P}$ and $\boldsymbol{U}_2 = \boldsymbol{Q}$.

4. *Two-layer, non-linear (2).* The transformation matrix $\boldsymbol{A}$ corresponding to the differential parameter computation over a fixed window of short duration is used to initialise the first layer in the transformation, e.g. $\boldsymbol{U}_1 = \epsilon \boldsymbol{A}$. The second layer of the transformation is initialised by setting $\boldsymbol{U}_2 = \boldsymbol{S}_j$.

5. *Recurrent, non-linear.* Similarly to case 2, the input-to-output connection matrix can be initialised with $\boldsymbol{U}_{I,O} = \epsilon \boldsymbol{S}_j$. The *state-to-state* and *state-to-output* matrices $\boldsymbol{U}_{R,R}$ and $\boldsymbol{U}_{R,O}$ respectively, are initialised with small random numbers and the elements of the *input-to-state* connection matrix $\boldsymbol{U}_{I,R}$ are set to zero.

In all cases, the elements of the bias vectors are set to zero. The values of the scaling coefficient ϵ and output range of the random number generator are determined empirically, so that the performance of the existing HMM system is partially or fully preserved.

8.5 Back-propagation

In chapter 4 we presented the derivatives of all HMM parameters which allowed us to optimise any likelihood-based objective function using a gradient search technique. Although using only first derivative information, the *QuickProp* algorithm was shown to provide fast convergence when applied to the optimisation of the MMIE and SMMIE objective functions. In this section we will extend the training algorithm to allow adaptation of the parameters of the input transformations. Since an input transformation can be considered as a special kind of neural network, the derivatives of the parameters of the transformations with respect to the objective function will be computed using the well-known back-propagation algorithm. Incidently, the back-propagation algorithm [100], was one of the main reasons for the resurgence of interest in ANN's.

The back propagation algorithm is an application of the chain rule to ANN's. It is not an optimisation technique, however it can be used to calculated the derivatives of the chosen objective function with respect to each weight in the transformation. The algorithm is well documented elsewhere [100], so only a short description follows. In general, let node i in the network receives input from all other nodes in the network given by x_i. The output, or *activation* of the node, y_i is given by

$$y_i = f_i(x_i)$$

where f_i is the node function. For nodes in the input layer y_i is set to the input pattern. For nodes not in the input layer x_j is given by

$$x_j = \sum_i y_i w_{i,j}$$

where $w_{i,j}$ is the weight between nodes i and j. The back-propagation algorithm provides a mechanism for calculating $\frac{\partial E}{\partial w_{i,j}}$ for each weight in the network, where E is the objective function. By applying the chain rule we can calculate the derivatives in the following order. First, the effect that changing the weight $w_{i,j}$ has on the input to node j is given by

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial x_j}\frac{\partial x_j}{\partial w_{i,j}} = \frac{\partial E}{\partial x_j} y_i$$

An expression for $\frac{\partial E}{\partial x_i}$ in terms of the node activations y_i is given by

$$\frac{\partial E}{\partial x_i} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial x_i} = \frac{\partial E}{\partial y_i}\frac{\partial}{\partial x_i} f_i(x_i)$$

Finally, for nodes which are not in the final layer, $\frac{\partial E}{\partial y_i}$ is given by

$$\frac{\partial E}{\partial y_i} = \sum_j w_{i,j}\frac{\partial E}{\partial x_j}$$

The above derivations are equivalent to propagating the derivatives $\frac{\partial E}{\partial y_i}$ backwards through the layers of the network, hence, the name "back-propagation". The algorithm is also applicable to recurrent networks as described in the preceding sections. In this case, the partial derivatives are computed by unfolding the network in time. The algorithm requires the same amount of computation as for a standard feed-forward network, however, the state unit activation should be stored to avoid redundant computation.

Next, the above algorithm will be used to calculate the derivatives of the MMIE objective function with respect to the parameters in the transformation. Analogous to appendix B we proceed to define the derivatives of the *log*-likelihood of the training set $\boldsymbol{\mathcal{O}}$ given a model $\mathcal{M}$. For the objective function we have

$$\frac{\partial}{\partial \boldsymbol{U}_{j,l}} f_{MMI}(\lambda) = \frac{\partial}{\partial \boldsymbol{U}_{j,l}} \mathcal{L}_\lambda(\boldsymbol{\mathcal{O}}|\mathcal{M}_{corr}) - \frac{\partial}{\partial \boldsymbol{U}_{j,l}} \mathcal{L}_\lambda(\boldsymbol{\mathcal{O}}|\mathcal{M}_{rec}) \tag{8.13}$$

where $\boldsymbol{U}_{j,l}$ is the connection matrix for layer l of the input transformation at state j. Using the chain rule and from the definition of $\mathcal{L}_\lambda(\boldsymbol{\mathcal{O}}|\mathcal{M})$ (equation B.1)

$$\frac{\partial}{\partial \boldsymbol{U}_{j,l}} \mathcal{L}_\lambda(\boldsymbol{\mathcal{O}}|\mathcal{M}) = \frac{1}{R}\sum_{r=1}^{R}\frac{1}{P_\lambda(\mathcal{O}_r|\mathcal{M})}\sum_{t=1}^{T_r}\frac{\partial}{\partial \boldsymbol{U}_{j,l}} b_j(\hat{\mathbf{o}}_{r,t,j})\frac{\partial}{\partial b_j(\hat{\mathbf{o}}_{r,t,j})} P_\lambda(\mathcal{O}_r|\mathcal{M}) \tag{8.14}$$

Where $\hat{\mathbf{o}}_{r,t,j}$ is the output of the transformation at state j, time frame t, utterance r. For the derivative of $b_j(\hat{\mathbf{o}}_{r,t,j})$ we have

$$\frac{\partial}{\partial \boldsymbol{U}_{j,l}} b_j(\hat{\mathbf{o}}_{r,t,j}) = \frac{\partial}{\partial \boldsymbol{U}_{j,l}}\hat{\mathbf{o}}_{r,t,j}\frac{\partial}{\partial \hat{\mathbf{o}}_{r,t,j}} b_j(\hat{\mathbf{o}}_{r,t,j}) \tag{8.15}$$

From appendix B

$$\frac{\partial}{\partial b_j(\hat{\mathbf{o}}_{r,t,j})} P_\lambda(\mathcal{O}_r|\mathcal{M}) = \frac{\alpha_j(t)\beta_j(t)}{b_j(\hat{\mathbf{o}}_{r,t,j})} \tag{8.16}$$

and

$$\frac{\partial}{\partial \hat{\mathbf{o}}_{r,t,j}} b_j(\hat{\mathbf{o}}_{r,t,j}) = \sum_{m=1}^{M} c_{j,m} b_{j,m}(\boldsymbol{o}_{r,t,j}) \boldsymbol{W}_{j,m}^{-1}(\hat{\mathbf{o}}_{r,t,j} - \boldsymbol{\mu}_{j,m}) \tag{8.17}$$

Using the back-propagation algorithm, the above derivative can be propagated backwards through the layers of the transformation which will enable one to compute the derivatives of the objective function with respect to each connection matrix (equation 8.15).

8.6 Implementation

The MMIE training of the HMM/IT architecture was incorporated within the MMIE training framework discussed in section 4.5. Following the basic design principles of HTK [117], the tying of input transformations was implemented in the internal representation of each HMM/IT using structure sharing. Storage for each shared transformation is allocated once, and all higher level structures use pointers to refer to the shared objects. In particular, shared objects can be whole transformations or individual layers. The syntax of the language used to define the transformations is outlined in appendix D. To eliminate redundant computation when shared objects are used, caching of observation vectors and intermediate node activations is implemented for each distinct transformation. In general, all necessary observation vectors from each transformation are computed and stored on the forward (α) pass. The pre-computed vectors are then used during the calculation of the backward (β) pass. In the case of recurrent transformations, the cache is automatically extended to accommodate the activation vector of the state units for each time frame. A final pass through the frames of the utterance is made in order to accumulate the derivatives of the likelihood function with respect to each parameter in the system. In the case of recurrent transformations, the recurrent structure is automatically unfolded in time for the length of the current utterance by updating the derivative accumulators whilst scanning the observation sequence backwards. All derivative accumulators are attached directly to the corresponding objects, hence any sharing of parameters is completely transparent to the the re-estimation procedure.

8.7 Experimental evaluation

This section presents the results from incorporating a variety of adaptive input transformations into existing HMM systems on the TIMIT database. In the experiments reported here, all transformations were initialised to perform non-linear mappings. All results quoted are on the TIMIT core set (192 utterances). The results are also supplemented by the number of parameters each system uses excluding transition probabilities. Table 8.1 contains baseline MLE and MMIE results which will be used for performance/system size comparisons. In order to allow for reasonable development and evaluation time, the transformations were used

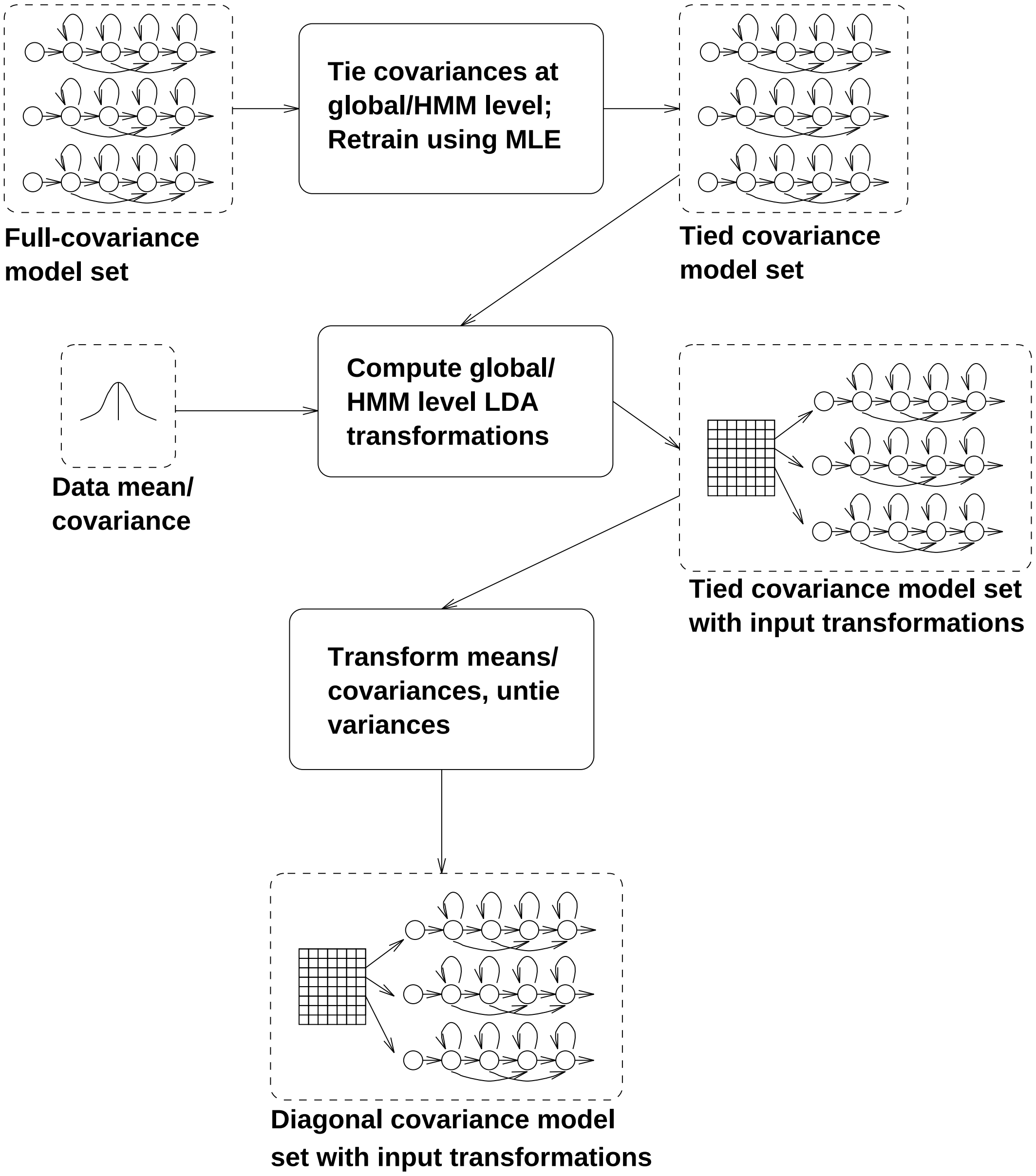

Figure 8.4: Derivation of input transformations

HMM type	MLE		MMIE		Param.
	%Corr	%Acc	%Corr	%Acc	
1/Diag, MFCC_E_D_A	62.94	55.58	67.79	61.00	11232
4/Diag, MFCC_E_D_A	68.29	61.59	71.61	65.23	45504
1/Diag, MFCC_E_D	59.78	54.79	65.85	60.47	7488
2/Diag, MFCC_E_D	62.99	58.14	68.09	63.20	15264

Table 8.1: Baseline MLE and MMIE results on the TIMIT database

in conjunction with single mixture diagonal covariance models. Training was performed on the full TIMIT training set (3696 utterances) and bigram language model with scale factor of 2.0 was incorporated into the looped phonetic model used in the MMIE training.

8.7.1 Training

So far, in MLE and MMIE training the parameters of the HMMs were updated after a complete pass through the training data. This type of method is often referred to as *off-line* optimisation. Since the MLE procedure requires a small number of iterations to converge, this is the most suitable training method. Neural networks typically rely on slower gradient based optimisation techniques, and in many cases parameter update after every single training pattern results in faster convergence. Such optimisation is often called *on-line* learning. On-line methods therefore cannot optimise a global objective function and such optimisation may result in oscillations. During some preliminary experiments, optimising the parameters of the input transformations only, was found to be slower than the MMIE training of the HMM parameters. In order to speed up the convergence of the optimisation algorithm a hybrid *on/off-line* optimisation scheme was adopted. During the first 6 iterations parameter updates took place after a gradually increasing number of training utterances. For the remaining part of the training, the optimisation was performed in *off-line* mode. This hybrid scheme resulted in 25% fewer interations than the corresponding fully *off-line* case.

8.7.2 Global input transformations

This section describes the experiments carried out in order to evaluate the performance of a global adaptive input transformation. The transformation was incorporated into a single mixture diagonal covariance model set using MFCC_E_D_A parametrisation. The baseline MLE and MMIE performance of this system is given in table 8.1. The input transformation was derived in a sequence of steps as shown in figure 8.4. Initially, a single mixture grand covariance system was built by tying all covariance matrices in the full covariance model set given in table 5.9. After tying, the parameters of the grand-covariance system were re-estimated using 4 iterations of MLE training. Using linear discriminant analysis, the grand-covariance matrix was transformed into the identity matrix and the between-class covariance matrix (equation 8.9) was diagonalised. Dimensionality reduction was performed according

Dimen.	Baseline		MMIE t-form		MMIE joint		Param.
	%Corr	%Acc	%Corr	%Acc	%Corr	%Acc	
G(39)	61.58	56.56	62.04	56.84	68.62	61.93	12792
G(34)	60.49	55.47	60.86	55.50	67.91	61.40	11152
G(30)	58.93	53.90	60.32	54.40	67.12	61.01	9840
G(26)	57.71	53.33	58.12	53.83	66.25	60.96	8528
G(22)	56.55	52.57	56.91	53.07	65.00	59.93	7216
G(18)	55.41	51.35	56.12	52.13	63.34	57.26	5904
G(14)	52.99	49.06	53.69	50.20	60.26	55.86	4592
G(10)	49.62	45.92	51.03	48.18	57.71	52.52	3280
G(6)	41.84	39.13	44.16	42.11	50.67	47.11	1968

Table 8.2: Results for single mixture models with a global input transformation.

to the eigenvalues from the second diagonalisation stage in LDA. In each case, the derived transformation matrix was used to transform the mean vectors of the HMMs. The grand-covariance matrix in the resulting system was replaced by individual unity-variance vectors for each distribution. Finally, the transformation matrix was used to initialise a single layer input transformation as described in section 8.4.3. The parameters of each model set were further optimised using 2 iterations of MLE training. The baseline performance of all model sets is given in table 8.2. The labels in the first column will be used to identify each individual system e.g. `G(14)` is the model set in which the output of the transformation consists of 14 features.

The first observation is that when all features are retained (39) the system's performance (56.56% accuracy/61.58% correct) is roughly equivalent to the performance of the original single mixture diagonal covariance system in table 8.1 (55.58%/62.94%). The somewhat disappointing result is that, with 26 features retained, the LDA transformation performs worse than the single mixture diagonal covariance system using `MFCC_E_D` parametrisation (table 8.1) e.g. 53.33%/57.71% vs. 54.79%/59.78%. The latter result is probably due to the assumptions made by LDA which do not hold in practice.

In the second set of experiments, the parameters of the transformation in each system were optimised according to the MMIE criterion. The results from these experiments are given in table 8.2/"MMIE t-form". The optimisation of the transformation has provided a small but consistent improvement in the recognition performance of all model sets. In absolute terms, it appears that the discriminative optimisation is more effective for the smaller systems e.g. ($\approx$ 3%) with 6 features, than the larger ones e.g. ($\approx$ 0.3%) with the full set of 39 features.

In the final set of experiments, the MMIE criterion was used to optimise all parameters in each system using further 15 iterations of the *QuickProp* algorithm. From the results in table 8.2/"MMIE joint", the global optimisation has provided a reasonable improvement in performance across all model sets. For comparison, with all 39 features retained, the jointly

Dimen.	Baseline		MMIE joint		Param.
	%Corr	%Acc	%Corr	%Acc	
M(39)	53.75	48.09	71.56	65.41	86112
M(34)	56.19	49.76	71.03	65.02	75072
M(30)	56.70	49.76	70.41	64.53	66240
M(26)	57.48	50.82	69.62	63.95	57408
M(22)	58.06	51.49	68.73	63.26	48576
M(18)	57.89	51.59	67.71	62.36	39744
M(14)	57.31	51.71	66.43	61.14	30912
M(10)	55.19	50.40	65.11	59.92	22080
M(6)	50.21	47.25	62.47	57.77	13248

Table 8.3: Results for single mixture models with model-specific input transformations.

optimised model set G(39) achieves better performance figures of 61.93%/68.62% than the baseline single mixture diagonal covariance system with performance of 61.00%/67.79% accuracy/correct respectively. Similarly to previous MMIE experiments, the the joint optimisation of HMM/IT parameters has been more effective for the "compact" systems than for the ones which use more parameters.

8.7.3 Model-specific input transformations

The aim of these experiments was to investigate the usefulness of model-specific input transformations. The rationale behind this approach is that a discriminative feature set appropriate for one class is not necessarily optimal in a discriminative sense for another class. As before, the systems used in the experiments were constructed from a full covariance model set where the same covariance matrix was shared amongst all states within each model. Linear discriminant analysis was used for each model to derive a model-specific input transformation with reduced dimensionality. One unfortunate result from the specificity of the transformations, is the loss[3] of the model-specific covariance weighting provided by the constant term in the Gaussian distributions. This term is given by

$$gconst = \frac{1}{(2\pi)^{D/2}|\boldsymbol{W}_{j,m}|^{1/2}} \tag{8.18}$$

There are two possible solutions to this problem:

- Compute the correct *gconst* for each model in the reduced subspace as suggested by Woodland et al. [109]. Fix the variance vectors in each distribution to unity and during training update only the mean parameters of the distributions.
- Calculate *gconst* in each distribution using $\boldsymbol{W}_{j,m} = \boldsymbol{I}$ and allow the training algorithm to optimise both the mean and variance parameters.

[3] The loss is incurred when the model-specific covariance matrix is transformed into the identity matrix by the corresponding input transformation.

System/Topology Inp×Out×Rec	Baseline		MMIE joint		Param.
	%Corr	%Acc	%Corr	%Acc	
R(39), $39 \times 39 \times 32$	61.58	56.56	70.12	64.56	16344
R(26), $39 \times 26 \times 32$	57.71	53.33	68.36	62.41	11664
R(14), $39 \times 14 \times 32$	52.99	49.06	64.71	59.82	7344
R(6), $39 \times 6 \times 32$	41.84	39.13	58.67	53.60	4464

Table 8.4: Results for a global recurrent input transformations using single mixture models.

Fixing the variance parameter was thought to be over-restrictive, hence, in the experiments presented here the latter approach was adopted. After derivation, each transformation, was used to rotate/scale the mean vectors of the distributions in the corresponding model. After this operation, the variances in each distribution were set to unity, and the parameters of each model set were re-estimated using 2 iterations of the Baum-Welch algorithm. The baseline performance of all newly derived systems is given in table 8.3. The effects of losing the correct Gaussian normalisation term are immediately visible from the significantly degraded recognition performance. As more dimensions are discarded the performance gradually improves, however, even with 22 features the performance of the system (51.49/58.06) is still significantly worse than the performance of the baseline diagonal covariance system (55.58%/62.94%).

In the following experiments, joint optimisation according to the MMIE objective function was carried out. The results are given under the heading "MMIE joint" in table 8.3. The results show improved recognition performance provided by the discriminative training algorithm. In figure 8.5, the performance of the model-specific transformation systems is plotted against the performance of the corresponding global transformation systems. In general, the decline in recognition accuracy is more gradual for the model-specific systems than for the ones that use global transformations. The relative reduction in error rate provided by M(6) over G(6) accounts for 20% whilst with all features kept the reduction in error rate is only 9%. As mentioned above, the poor performance of the initial systems was due to the inability to retain the model-specific normalisation terms, hence a comparison with other systems which use similar numbers of parameters is thought to be more meaningful. In this respect, the gain in recognition performance appears to be associated with a dramatic increase in the number of free parameters. For example, system G(39) achieve performance figures of 61.93%/68.62%, performs better than system M(6) (57.77%/62.47%) and uses 3% fewer parameters. A similar argument applies when comparing model sets M(22), M(26), M(30), M(34) against the 4-mixture diagonal covariance system in table 8.1. In general, although more flexible than the global transformation systems, the model-specific transformations do not appear to be very efficient in terms of the number of parameters used. Furthermore, the discriminative training of these systems is too computationally expensive compared to a conventional HMM architecture to justify its use in practice.

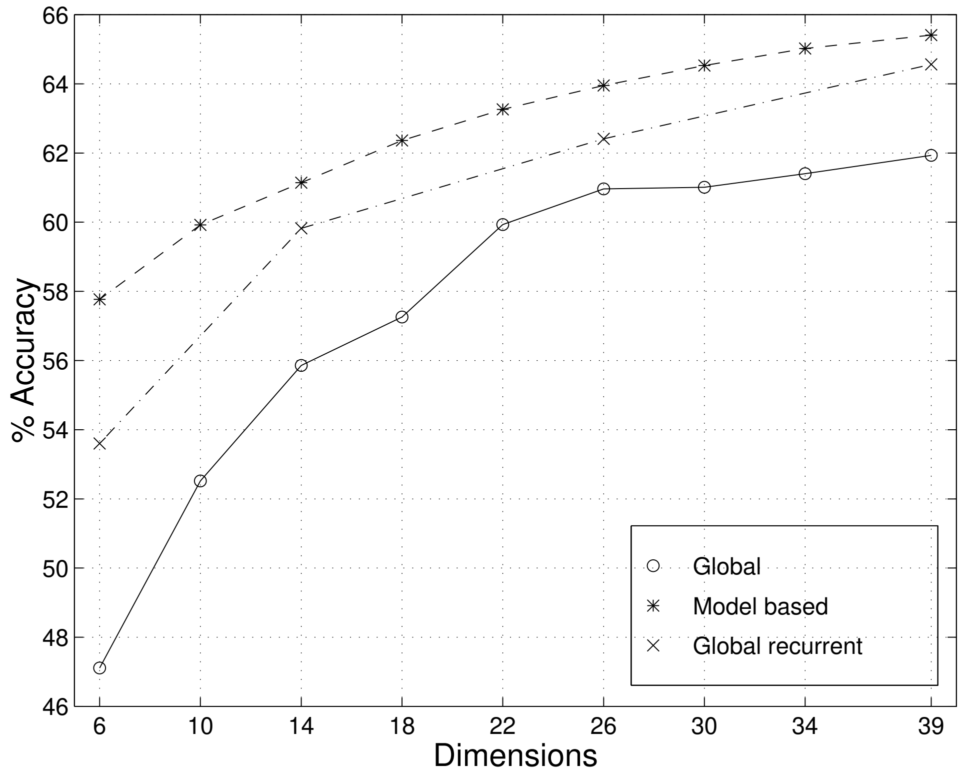

Figure 8.5: A comparison of different input transformations

8.7.4 Recurrent input transformations

This section investigates the use of recurrent input transformations within the HMM framework. Although it is possible to construct a system which uses model-specific recurrent input transformations, in this study we have considered only a single global input transformation for the following reasons:

- The pruning mechanism used in the Forward-Backward algorithm was shown to be crucial for the tractable implementation of MMIE training. Unfortunately, model activation/deactivation introduces discontinuities in the sequence of observation vectors which causes difficulties in the processing carried out by recurrent structure.

- The model-specific transformations investigated in the previous section were shown to be inefficient with regard to the number of parameters used.

Furthermore, due to the increased amount of processing required to compute the derivatives of the state unit connections only a limited number of systems were evaluated. The recurrent input transformation was derived in the same way that was used to derive the global input transformations in section 8.7.2. In all cases, the transformations had 32 state

System/Transformation	Baseline		MMIE joint		Param.
	%Corr	%Acc	%Corr	%Acc	
`P1`, [1] recurrent $26 \times 26 \times 32$/LDA	60.46	55.93	69.76	63.67	10910
`P2`, [1] LDA/global	60.46	55.93	67.32	61.43	8190
`P3`, [1] LDA/model	53.32	48.29	70.84	65.18	41184
`P4`, [1] Δ-comp/global	59.17	54.21	68.47	63.33	9204
`P5`, [1] Δ-comp/global, [2] LDA/global	60.87	56.41	69.16	64.23	9906
`P6`, [1] Δ-comp/global, [2] LDA/model	52.86	47.53	71.07	65.92	42900

Table 8.5: Results for various input transformations using single mixture diagonal covariance models.

units. No attempt was made to optimise this number, however, results in [98] suggest that improved recognition can be achieved as more state units are added. In each system, the derived transformation was optimised jointly with the HMMs' parameters using 25 iterations of MMIE training. The performance of these systems is given in table 8.4/"MMIE joint" and the corresponding accuracy plot is shown in figure 8.5. In terms of recognition performance, the systems perform better than the corresponding global non-recurrent transformation systems. For example, system `R(26)` achieves 62.41%/68.36% whilst `G(26)` yields 60.96%/66.25%. In terms of performance with roughly equal numbers of parameters, `R(6)` is worse than `G(14)` and systems `R(14)/R(26)` are somewhat similar to systems `G(22)/G(34)` respectively.

8.7.5 Other transformations

This section describes experiments with a collection of one and two layer transformations applied to the single mixture diagonal covariance HMM system using `MFCC_E_D` parametrisation (see table 8.1). The latter feature set was chosen, since it allows the delta parameter calculation to be mapped onto a single layer input transformation over an input window of five frames. After initialisation, joint optimisation of all HMM/IT parameters was carried out for each system using 25 iterations of MMIE training. The performance of these systems is shown in table 8.5. The transformation type is given in terms of layer number, the initialisation procedure used and the level of sharing. For example, {[1] Δ-comp/global, [2] LDA/model} describes a two-layer input transformation where the first layer is initialised to perform first differential parameters computation and is shared by all states in the system; the second layer of the transformation is model-specific and is initialised to perform LDA on the components of the `MFCC_E_D` feature vector.

The first system presented (`P1`) makes use of a global recurrent transformation with 32 state units and no dimensionality reduction. The system uses nearly 30% fewer parameters than the baseline two mixture diagonal covariance system shown in table 8.1 and at the same achieves better recognition performance. The next system (`P2`) uses a single layer global input transformation with no dimensionality reduction. Compared to the corresponding single mixture diagonal covariance system in table 8.1, the input transformation has resulted in an average improvement in performance of 1% (absolute). The model-specific version of this transformation (system P3) achieves recognition performance of 65.18% accuracy/70.84% correct at the expense of 41184 parameters. Incidently, this system performs better than system M(30) in table 8.3 which uses 60% more parameters. In system P4, the transformation was initialised to perform calculation of first order cepstral coefficients from an input window of 5 frames. The system exhibits higher recognition accuracy than the P2 system which uses a global LDA-initialised transformation. The addition of a second layer (system P5) does provide improvements in performance which is very similar to the performance of the recurrent input transformation system P1. Finally, the best performing system P6 uses two-layer input transformations, which are derived as a combination of the global transformation used in system P4 and the model-specific transformations used in system P3. In fact, system P6 achieves similar performance and uses fewer parameters than the four-mixture diagonal covariance system shown in table 8.1.

8.8 Summary

This chapter has presented the theory and evaluation of adaptive input transformations in the HMM framework. The conventional HMM framework was extended to accommodate input transformations which transform the observation vectors before they enter the output distributions of the HMMs. The system was implemented to allow arbitrary sharing of input transformations in common with all other HMM parameters. A variety of deterministic initialisation procedures were described, which allow transformations to be introduced into existing HMM systems without any significant degradation in performance. Recurrent input transformations were proposed in an attempt to tackle the observation independence assumption in the HMM framework. An adaptive feature extraction procedure was also discussed whereby an input transformation is initialised to perform the calculation of first order dynamic coefficients.

In all experiments presented, the joint optimisation of HMM/IT parameters according to the MMIE criterion has resulted in improved recognition performance. In the initial set of experiments, performance gains were visible from optimising only the parameters of a global input transformation. However, the improvements from such optimisation were found to be small. Similarly to the experiments presented in chapter 5, the discriminative optimisation of the HMM/IT parameters has proven more effective for the model sets with smaller observation vector size. Compared to the baseline results of 55.58%/62.94% accuracy/correct for the MFCC_E_D_A single mixture diagonal covariance system trained using MLE, the global discriminative optimisation of HMM/IT parameters have provided the following improvements. The use of a global input transformation allowed us to reduce the observation vector size to 18 features and at the same time it has improved the recognition performance to 57.26%/63.34%. Using model-specific input transformations, the performance of the system rose to 57.77%/62.47% with only 6 features. Finally, when using a global recurrent input transformation, the performance of the system was 59.82%/64.71%

with a feature set of 14 components.

In general, global input transformations, were found to be more efficient in their use of parameters than the corresponding model-specific transformations. In many cases the latter type achieved better recognition performance, however, the same results could be obtained by applying MMIE training to conventional HMM systems with multiple mixture components. For example, with all 39 features retained the system with model-specific transformations achieved performance figures of 65.41%/71.56% accuracy/correct which is roughly equivalent to the performance of the 4-mixture diagonal covariance system (65.23%/71.61%) which uses nearly 50% fewer parameters.

Several input transformations were evaluated in conjunction with HMMs using the `MFCC_E_D` feature set. The best performance figures were achieved using model-specific input transformations with single and 2-layer topologies. The recurrent input transformation and the two-layer transformation initialised to compute delta coefficients followed by LDA also provided good performance with very efficient use of parameters. In general, the performance of the recurrent transformation (63.67%/69.76%) was very similar to the performance of the 2-layer global transformation (64.23%/69.16%). This was closely followed by the global input transformation initialised to perform the calculation of first order differential parameters (63.33%/68.47%), which, in turn, was better than the performance of the single global transformation initialised to perform LDA (61.43%/67.32%).

Chapter 9

Conclusions

There are many ways of improving the performance of HMM-based speech recognition systems. Improved recognition performance can be achieved by optimising the front-end of the system, utilising output distributions with more mixture components, introducing context-dependent models, increasing the complexity of the language model, etc. In this thesis, emphasis was placed on improving the discriminative abilities of the acoustic models through parameter estimation according to the MMIE objective function. The study also investigated ways of improving the acoustic representation with the aim of providing compact and informative feature vectors. The latter goal was pursued through the introduction of discriminative feature selection and adaptive feature extraction techniques. The following sections summarise the work carried out and also give suggestions for future research.

9.1 Summary of work and general conclusions

Chapter 3 outlined the rationale behind two very different optimisation criteria - Maximum Likelihood estimation (MLE) and Maximum Mutual Information estimation (MMIE). MMIE was shown to be more appealing since it does not rely on the model correctness assumptions, which form the basis of MLE. On the other hand, MMIE is more difficult to implement, requires more computation and when this work began the empirical evidence of its success was very limited. One point to note is that although MMIE is discriminative in nature, a marginal increase in the value of the objective function does not guarantee improved recognition accuracy even for the training set. This is a consequence of the fact that in MMIE multiple incorrect paths are considered together, hence, decreasing their total likelihood does not guarantee recovering the dominance of the correct path. Other discriminative training techniques such as MCE claim a more direct relationship to the empirical error rate of the recogniser, whereby discrimination is achieved against the most likely incorrect path. In the spirit of MCE, an extension to MMIE was discussed where a sigmoid weighting function is applied to each individual training utterance to re-adjust the corresponding derivative contributions (SMMIE).

The effectiveness of discriminative training depends to a large extent on the availability of a good training algorithm which provides a reasonable improvement in the value of the objective function over a small number of iterations. With this in mind, chapter 4 outlined a variety of techniques which can be used to optimise the HMM parameters according to a discriminative objective function. Second order techniques were considered but found impractical to implement due to the substantial computational requirements involved in evaluating and inverting the Hessian matrix of second derivatives. Instead, our study moved to investigate the application of local optimisation techniques from the field of neural networks where separate learning parameters are employed for each individual HMM parameter. Following Kapadia [61], a discriminative training framework based on the Forward-Backward algorithm was outlined where the calculation of parameter derivatives is carried out in two passes: one for the numerator of the MMIE objective function using the correct acoustic model constructed from the transcription of the utterance, and a second pass where the training utterances are aligned against the recognition model. The update of model parameters was carried out in a post-processing stage in which, parameter derivatives accumulated in the two preliminary passes are combined together. Chapter 4 also presented the expressions required to calculate the derivatives of any likelihood based objective function with respect to each HMM parameter. Stochastic and positive constraints were automatically satisfied within the re-estimation procedure by performing suitable mappings on the corresponding parameters.

In chapter 5 MMIE and SMMIE were empirically compared to MLE on three speech databases. The *QuickProp* algorithm was found to be very effective for optimising the HMM parameters according to the MMIE criterion. On average, the algorithm required 12-18 iterations to achieve a reasonable improvement in the value of the objective function. In all experiments, MMIE demonstrated improved recognition performance over the corresponding MLE-trained model sets. On all three databases, discriminative training was found to be more effective for the smaller model sets which used fewer mixture components per state. For the BTL E-set the SMMIE training algorithm demonstrated further improvements in recognition performance. However, on the ISOLET database the improvements were only marginal. This was explained by the larger number of speakers in the training set which was thought to be more representative of the task. Modelling cepstral parameter correlations using full covariance distributions was shown to be crucial for achieving state of the art results on the two isolated tasks. The use of second differential parameters was shown to yield gains in performance on the TIMIT continuous phone recognition task, in particular, when used in conjunction with "good" acoustic models. Further improvements in phone recognition accuracy were obtained through the use of a fourgram back-off language model, used in conjunction with the best performing MLE and MMIE trained models.

Chapter 7 outlined a feature selection algorithm based on the mutual information criterion. It was argued that since the mutual information measure relates to the quantities used during recognition, it would be an appropriate criterion to rank-order the components of the feature vector. The MMIC algorithm was used to rank-order the components of the 39 dimensional `MFCC_E_D_A` feature vector used in the TIMIT continuous phone recognition experiments. The algorithm was shown to provide an intuitively plausible feature order. It also fulfilled the objective of allowing us to reduce the size of the observation vectors with a gradual increase in the recognition error rate. Unfortunately, the MMIC algorithm did not provide any improvements in performance over a number of manually selected feature sets. The latter observation was attributed to the various approximations made within the algorithm and in particular, the lack of modelling of the correlations between the components of the feature vector.

In chapter 8 the traditional HMM framework was extended to accommodate adaptive input transformations whose parameters are optimised jointly with the HMM parameters according to MMIE objective function. A variety of input transformations were discussed together with methods for their deterministic initialisation. The combined HMM/IT architecture was evaluated on the TIMIT phone recognition task. In all experiments, the jointly optimised systems achieved improved recognition accuracy over the corresponding MMIE-trained HMM systems. The HMM/IT framework was implemented to allow sharing of input transformations in common with all other HMM parameters. Unfortunately, in the context of single Gaussian HMMs, the model-specific input transformations were shown to be inefficient in their use of parameters. In general, the optimal balance between performance and model complexity was achieved through the use of recurrent input transformations and one/two-layer input transformations initialised to calculate first differential cepstral parameters in conjunction with LDA. The results from these experiments were encouraging and the application of adaptive input transformations is to be investigated further in the larger application domain.

9.2 Some suggestions for future work

There are several aspects of the work presented in this thesis that will require further investigation both in terms of different application domains and modifications to the existing techniques.

9.2.1 Discriminative training and LVCSR

The MMIE results presented in this thesis are very promising. Although there are a number of tasks with small but useful vocabularies, current research in speech recognition is concentrated on improving the performance of large vocabulary systems. By current standards, the tasks considered in this study were based on very small vocabularies e.g. 48 in the case of continuous phone recognition. In the continuous phone recognition experiments, this was an important issue since it allowed us to use the somewhat simple looped phonetic model when calculating the denominator of the MMIE objective function. Indeed, improvements in recognition performance were observed in all phone recognition experiments. However, it would not be a surprise if the word recognition error rate of the same systems was not improved at all. This follows from the fact that many of the discriminations at phone level achieved by the MMIE training algorithm are unnecessary when applying the higher level word constraints. The implication of this argument is that in order to achieve meaningful discriminations the recognition grammar should be fully incorporated into the training procedure. Unfortunately, incorporating higher level language constraints requires a substantial search effort in a potentially huge HMM state space. One possible way of achieving this is to use N-best sentence hypotheses as an approximation to the denominator of the MMIE objective function. However, as the baseline performance of our conventional systems improves, it is very likely that the top N hypothesis will differ from one to another by a very small number of words. Consequently, N will have to be chosen sufficiently large in order to allow for a reasonable number of substantially different alternatives to be considered. With research moving onto very large vocabulary systems (in excess of $20,000$ words) a different approach to hypothesis representation has become more viable. Instead of generating an N-best list, a lattice of word choices is produced. The lattice consists of nodes connected by arcs, where each node represents a time instance and each arc a word which is hypothesised as occurring between two time frames. Since, the lattice can be considered as a compact representation of a large number of sentence hypotheses, it can be used to approximate the denominator of the MMIE objective function. Furthermore, with recent advances in search techniques [3], the lattice generation can be speeded up through the use of multiple forward-backward passes.

9.2.2 Language model weight

For the E-set and Alphabet recognition tasks the language model scores were assumed to be uniform and their role during training and recognition was not important. Virtually, all phone recognition experiments made use of a phone level bigram language model whose scores were raised to the power of 2.0 prior to being combined with the acoustic scores both during training and recognition. The language model weight was determined empirically in a preliminary set of experiments using MLE/MMIE trained models. Consequently, it is very possible that the language model weight was appropriate for some experiments and wrong for others. As Normandin [84] points out, the language model weight is very similar to the code-book exponents used in the semi-continuous modelling framework and both types of parameters arise from the incorrect modelling assumptions in the Markov models. In the same way as used in [84] the language model weight can be adapted during training according to the MMIE criterion.

9.2.3 Discriminative feature selection

Based on the mutual information criterion, the MMIC training algorithm was shown to provide a reasonable ordering of the cepstral coefficients and their derivatives with gradual and modest degradation in performance. As mentioned before, one limitation of dynamic parameters is that the slope/curvature of the features are thought to be the only useful features. This assumption can be put to the test by applying the MMIC feature selection algorithm to a feature vector of a large dimensionality which apart from the standard `MFCC_E_D_A` parameters also includes basic cepstral parameters from neighbouring frames.

The failure of the algorithm to provide improved recognition performance over the manually selected feature sets was attributed to the assumption that all components of the feature vector are independent. Inspection of the correlation matrix of the TIMIT training data showed that this is not the case. Hence, another desirable extension is to incorporate covariance modelling within the MMIC feature selection algorithm. The basic idea is to build a full covariance system and using the covariance matrices to attempt to derive a model which approximates the change in the mutual information measure when removing each possible pair of features.

9.2.4 Adaptive input transformations

The area of adaptive input transformations is very new and the set of experiments presented in this thesis barely scratches the surface of the multitude of possible topologies. The application of adaptive transformations in LVCSR systems will depend to a large extent on the successful implementation of the MMIE training algorithm and the ability to exploit the full grammar constraints within the training process. Another observation is the fact that with more complex systems (e.g. large number of mixture components) the parameters employed by the input transformations will be only a small proportion of the total number of parameters used by the system. In these cases, model-specific or even state-specific input transformations may prove a viable option.

Recurrent input transformations offer great flexibility by allowing one to enhance their structure through the use of more state units. With more state units, such transformations are expected to model longer contextual effects. However, it is also expected that with the higher level word constraints in place, the difference in performance between the recurrent input transformations and the windowed input transformations will narrow, due to the limited range of context modelled by the state units in the recurrent transformations.

9.3 Summary

This chapter has summarised the conclusions from this study and identified areas for future research. The main conclusion is that discriminative training should be considered in any small vocabulary task or real-time speech recognition system. The use of adaptive input transformations in the HMM framework has demonstrated very promising results and they merit further research. Future work in discriminative methods should aim at applying them to larger scale tasks such as continuous speech dictation systems. Furthermore, with the ever increasing power of computer technology such investigations are rapidly becoming possible.

Appendix A

Databases

This appendix describes the speech databases and the basic HMM sets used in the various experiments described in this thesis.

A.1 Speech coding

The speech coding procedure described below was used to parametrise the ISOLET and TIMIT databases. Parametrisation was carried out using the `HCode` tool from the HTK toolkit [117]. For the processing, each utterance was split into frames of 25ms duration with frame shift at 10ms. Pre-emphasis by a factor of $k = 0.97$ was applied to the speech waveform, defined by

$$s'_n = s_n - ks_{n-1}$$

Each frame was then transformed using a Hamming function defined by

$$s'_n = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) s_n$$

where N is the number of samples in the frame and s_n is the n^{th} sample. Each frame was coded into 12 Mel-Frequency Cepstral Coefficients (MFCCs), plus a log-energy component. The MFCC coefficients are computed from the output of a mel-scaled triangular filter bank according to

$$C_i = \sum_{j=1}^{P} m_j \cos\left(\frac{\pi i}{P}(j-0.5)\right) \qquad for \qquad i = 1, \ldots M$$

where $P = 24$ is the number of filters, and M is the desired number of MFCC parameters. This was followed by cepstral liftering using a factor of $L = 22$

$$C'_i = \left(1 + \frac{L}{2}\sin\left(\frac{\pi n}{L}\right)\right) C_i$$

The final observation vectors were formed by coercing the MFCCs with the energy component and with their first and second differentials. Three commonly used feature sets

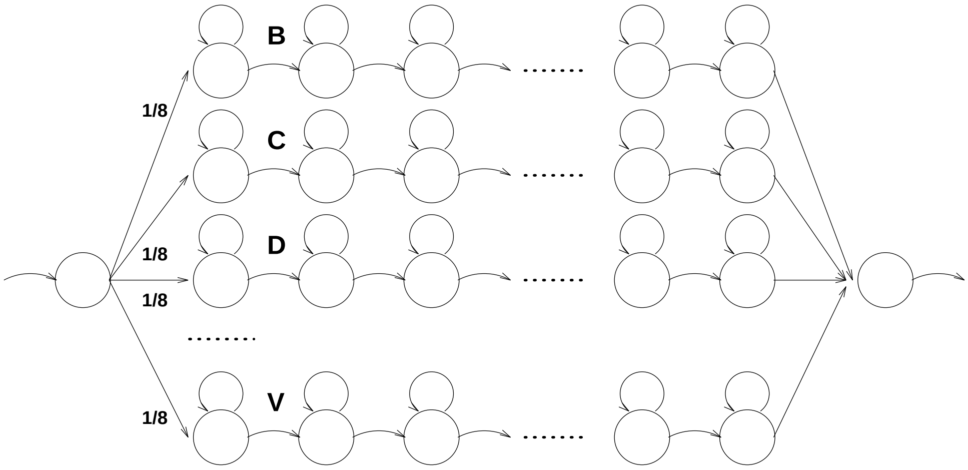

Figure A.1: E-set recognition network with a uniform language model.

are

$$\begin{aligned}\texttt{MFCC\_E} &= \{C_1, \ldots, C_{12}, E\} && \text{(A.1)}\\ \texttt{MFCC\_E\_D} &= \{\texttt{MFCC\_E}, \Delta C_1, \ldots, \Delta C_{12}, \Delta E\} && \text{(A.2)}\\ \texttt{MFCC\_E\_D\_A} &= \{\texttt{MFCC\_E\_D}, \Delta^2 C_1, \ldots, \Delta^2 C_{12}, \Delta^2 E\} && \text{(A.3)}\end{aligned}$$

First derivatives of the basic `MFCC_E` coefficients are computed using the regression

$$\Delta C_i(t) = \frac{\sum_{\tau=1}^{V} \tau(C_i(t+\tau) - C_i(t-\tau))}{2\sum_{\tau=1}^{V} \tau^2}$$

For the second derivatives, the summation in the numerator is taken over the differences between the first derivative parameters in the feature vector.

A.2 The BTL E-set database

The British English E-set is defined as the set {"B", "C", "D", "E", "G", "P", "T" & "V"}. E-set recognition is considered to be a particularly difficult task due to the high level of confusability between the different classes in the set. The BTL E-set database was collected and distributed by British Telecom Laboratories (BTL) as part of the CONNEX project and forms a subset of a larger alphabet database. The data was recorded in a noise-free environment and was originally sampled at 20KHz. Utterance boundaries were marked by a semi-automatic process. Each member of the E-set is represented by three utterances from each of the 104 different speakers (54 males, 50 females). The speakers were split into two halves to form a training set of 1239 utterances and a test set of 1219 utterances. The acoustic preprocessor used the output of a 27 channel filter-bank (SRU-Bank) followed by a Discrete Cosine Transform to produce 12 Mel Frequency cepstral coefficients (MFCCs) and their first differentials. The twelve coefficients include the zeroth coefficient which is the

letter	*phones*	*letter*	*phones*
A	ey	N	eh n
B	b ee	O	oh
C	c ee	**P**	p ee
D	d ee	Q	k y ooh
E	ee	R	ah r
F	eh f	S	eh s
G	j ee	**T**	t ee
H	ey ch	U	y uh
I	ay	**V**	v ee
J	j ey	W	d uh b ul y ooh
K	k ey	X	eh k s
L	eh l	Y	w ay
M	eh m	**Z**	z ee

Table A.1: Alphabet pronunciations

average value of the log power spectrum. Principal Component Analysis was then used to transform the covariance matrix of the data to the identity matrix. The preprocessor and the partitioning of the data are identical to the ones used by Woodland and Cole in [109].

Each member of the E-set was modelled by a left-to-right HMM with 15 emitting states and no skip transitions. The network used in the recognition experiments is shown in figure A.1.

The BTL E-set database was used by McCulloch [75] to compare the performance of standard ML training vs a discriminative training technique accomplished within the Alphanet framework [18]. Using 8 state left-to-right models with a shared variance vector and parametrisation of 8 MFCC coefficients he achieved 68.58% accuracy for the ML training and a peak performance of 74.09% accuracy after several thousand iterations of discriminative training.

Woodland and Cole [109], performed experiments on the BTL E-set database in order to evaluate a discriminative feature extraction scheme based on state specific input transformation. The best result achieved using MLE trained models was 91.1% accuracy using 24 parameters (12 MFCC + 12 delta), 15 state left-to-right models and full-covariance output distributions. Using state specific input transformations with reduced parameter vectors of 16 features the result improved to 92.1% accuracy.

The same database was also used by Ayer [4] to evaluate the performance of a global Whole-word Adaptive LDA (WALDA) transformation. The system used DTW templates equivalent to a set of left-to-right HMMs allowing loop and skip transitions without penalty. The application of the WALDA transformation improved the accuracy to 96.0% using 19 features. This is the best result published on this database to date.

A.3 The ISOLET database

The ISOLET database is an isolated speech, alphabet database collected and distributed by the Oregon Graduate Institute. The database was collected as part of the development of the EAR system [42, 28, 41]. The database consists of two tokens of each letter produced by 150 American English speakers, 75 male and 75 female. As used by the developers, the data is divided into five sections of 30 speakers each. The first four sections are used for training (120 speakers) and the last portion is used for testing (30 speakers).

The letters were recorded one at a time in random order in a single session lasting approximately 30 minutes. The speech was recorded using a Sennheiser HMD 224 noise cancelling microphone, low-pass filtered at 7.6KHz and sampled at 16KHz with 16 bit resolution. During the recording session, each digitised utterance was played back to ensure that the entire letter was captured into the recording buffer. In addition, each utterance in the database was manually verified by listening to it and simultaneously viewing the waveform. For the experiments described in this thesis, the speech data was parametrised as described in section A.1. Each word in the alphabet was modelled by a left-to-right HMM with 15 emitting states and no skip transitions.

For several years, English alphabet recognition has been a popular task in the study of speech recognition. Early approaches [29] were template-based and achieved speaker dependent recognition performance of 60% to 80%. The FEATURE system [30] was speaker independent and demonstrated substantially improved recognition accuracy (89%) by combining knowledge-based features and multivariate classifiers. Recently [39], improved recognition accuracy of 93% has been obtained using hidden Markov models. As mentioned earlier, the ISOLET database emerged from the development of the EAR system. The initial performance of the EAR system on the speaker independent letter classification task was 86%. This was later improved to 96% by optimising the feature sets and the overall classification strategy. In the final version of the system [27], letter classification was performed using a fully connected neural network with 617 inputs, 52 hidden units and 26 outputs. Recognition within the E-set was further refined using a specialised E-set recognition network.

A.4 The TIMIT database

The TIMIT corpus of read speech has been designed to provide speech data for the acquisition of acoustic-phonetic knowledge and for the development and evaluation of automatic speech recognition systems [65]. TIMIT contains a total of 6300 sentences, 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States. The text material in the TIMIT prompts consists of dialect sentences (SA), phonetically-compact sentences (SX) and phonetically diverse sentences (SI). The dialect sentences were meant to expose the dialectal variants of the speakers and were considered unsuitable for training speaker independent phone recognisers. Thus, the full training set consists of 3696 utterances.

In the latest release, the speech material has been subdivided into portions for training and testing. The test data has a core portion containing 24 speakers, 2 male and 1 female

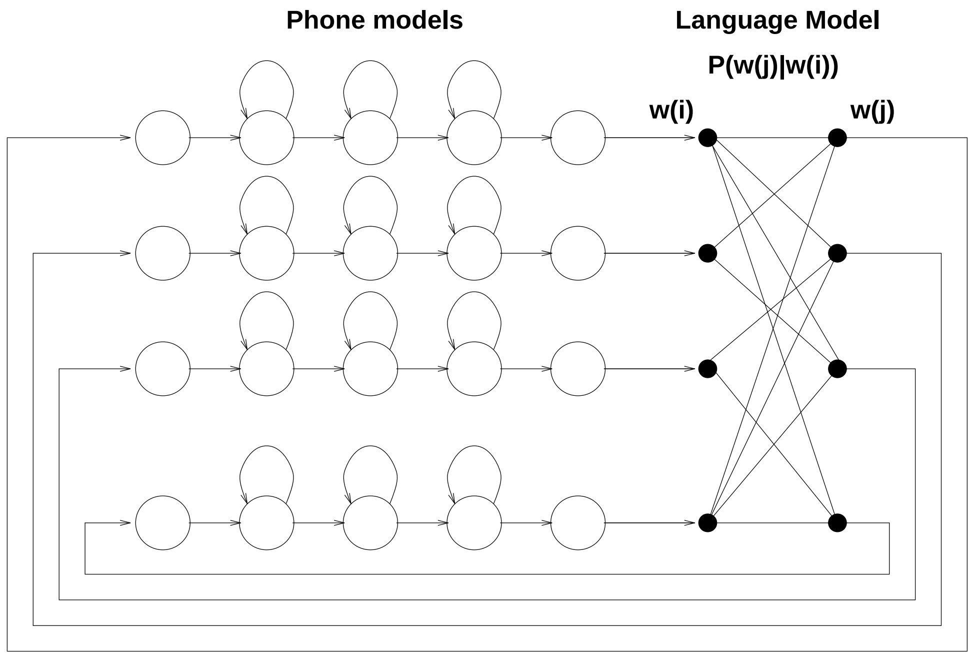

Figure A.2: Looped phonetic model for continuous phone recognition using bigram language model.

from each dialect region. Each speaker read a different set of SX sentences. Thus, the core test material contains 192 sentences, 5 SX and 3 SI for each speaker, each having a distinct text prompt. No speaker appears in both the training and test partitions. The texts corresponding to the test data were also checked to ensure that they included at least one occurrence of each phoneme.

The experimental conditions are similar to those used by Lee and Hon in their baseline TIMIT experiments [67]. The 61 phone TIMIT label set was mapped down to a 48 phone set. The reduced phone set is given in table A.2. For scoring purposes only, the 48 phone is folded into a reduced 39 phone set as it was done in [67]. All results are stated in terms of percentage correct and percentage accuracy defined as

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

and

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

where H is the number of correct labels, I is the number of insertions and N is the total number of labels in the recognition file. The number of correct labels is defined as

$$H = N - (S + D)$$

where S and D are the number of substituted and deleted labels respectively as computed by the dynamic programming algorithm by matching the recognised transcription against the correct label file.

Phone	Example	Allophones	Phone	Example	Allophones
/iy/	me**a**t		/en/	lesso**n**	
/ih/	h**i**t		/ng/	si**ng**	/eng/
/eh/	s**e**t		/ch/	**ch**urch	
/ae/	h**a**t		/jh/	**j**udge	/j/
/ix/	ros**e**s		/dh/	**th**ey	
/ax/	th**e**	/ax-h/	/b/	**b**ob	
/ah/	b**u**tt		/d/	**d**ad	
/uw/	r**oo**t	/ux/	/dx/	Sea**tt**le	
/uh/	b**oo**k		/g/	**g**ag	
/ao/	th**ou**ght		/p/	**p**ope	
/aa/	c**o**t		/t/	**t**ot	
/ey/	m**ai**n		/k/	**k**ick	
/ay/	b**i**te		/z/	**z**ap	
/oy/	t**oy**		/zh/	trea**s**ure	
/aw/	b**ough**		/v/	**v**ery	
/ow/	m**oa**t		/f/	**f**ield	
/l/	**l**ead		/th/	**th**ief	
/el/	logica**l**		/s/	**s**ize	
/r/	**r**ed		/sh/	**sh**ed	
/er/	yea**r**	/axr/	/hh/	**h**ay	/hv/
/y/	**y**et		/cl/	(voiced cl.)	/p/, /t/, /k/, /q/-cl
/w/	**w**et		/vcl/	(unvoiced cl.)	/b/, /d/, /g/-cl
/m/	**m**ake	/em/	/epi/	(epin. cl.)	
/n/	**n**on	/nx/	/sil/	(silence)	/h#/, /#h/, /pau/

Table A.2: 48 phone TIMIT set

For all recognition experiments, except when stated otherwise, a phone bigram language model was used. The language model was computed using the `HLStats` tool from HTK. The language model scores were raised to the power of 2.0 before being combined with the acoustic scores in the Viterbi recognition. Recognition was performed using a looped phonetic model with structure shown in figure A.2.

The TIMIT database has become a benchmark test for scoring continuous phone recognition systems. The first HMM results on this task were provided with the phone recognition version of the SPHINX system [67]. The authors achieved performance of 66.1% accuracy/73.8% correct using 1450 right context-dependent discrete phone models with multiple code-books. Young [112] reported results of 59.9%/73.7% using 807 generalised triphone models with continuous mixture density output distributions. Robinson [98] achieved 75.0%/78.6% using a recurrent error propagation network. Using a slightly different phone set Ljolje [73] obtained 69.4%/74.8% using single mixture continuous density, "quasi-triphonic" models with durational constraints and a trigram language model. Most recently, Young and Woodland [116] achieved 72.3%/76.7% using 1176 state clustered right context dependent bi-phone HMMs assembled from a library of 1142 different states with 9 mixture components per state.

A.5 E-set spectrograms

This section contains example spectrograms for the members of the American-English E-set. The examples were taken from speaker "fcmc0" in the ISOLET database. The spectrograms aim to show that accurate classifications requires making fine phonetic distinction over the short initial section of each utterance. Due to the relatively short part of the utterance which corresponds to the consonant sound, it is also necessary to precisely model each segment's duration. This is usually achieved by using word-level HMMs with large number of states.

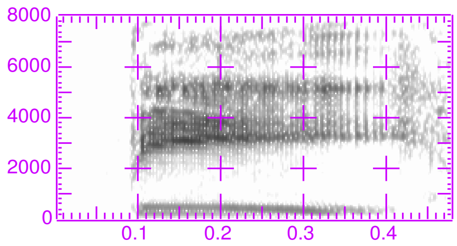

Figure A.3: Spectrogram fcmc0/B

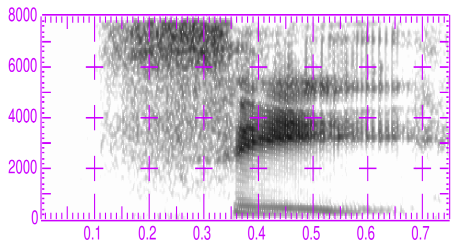

Figure A.4: Spectrogram fcmc0/C

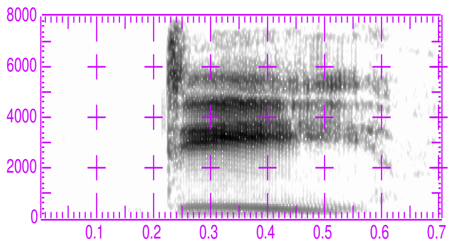

Figure A.5: Spectrogram fcmc0/D

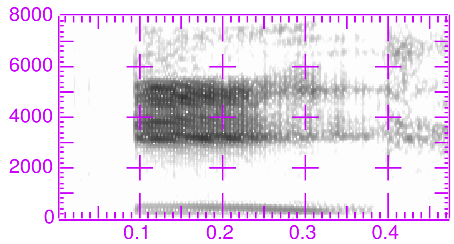

Figure A.6: Spectrogram fcmc0/E

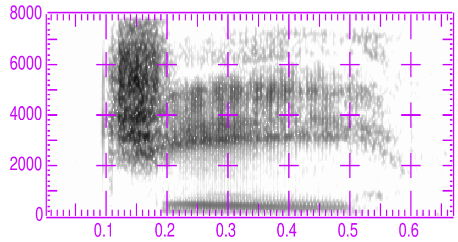

Figure A.7: Spectrogram fcmc0/G

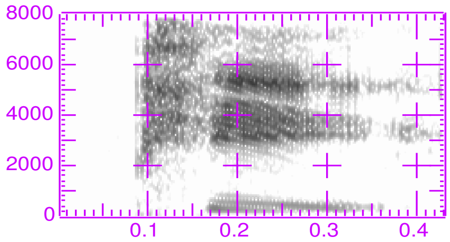

Figure A.8: Spectrogram fcmc0/P

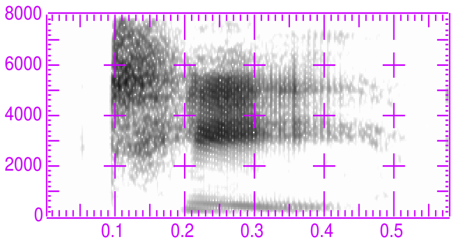

Figure A.9: Spectrogram fcmc0/T

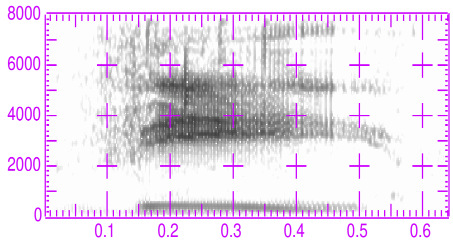

Figure A.10: Spectrogram fcmc0/V

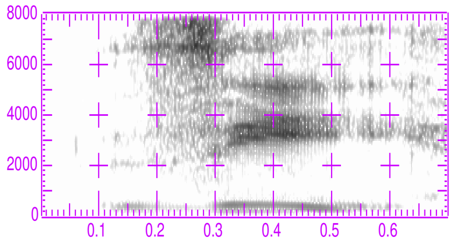

Figure A.11: Spectrogram fcmc0/Z

Appendix B

HMM Parameter Derivatives

This appendix presents the derivatives of the likelihood function with respect to the various HMM parameters.

B.1 Likelihood differentiation

The log likelihood of the training set $\boldsymbol{\mathcal{O}}$ given a model $\mathcal{M}$ is

$$\mathcal{L}_\lambda(\boldsymbol{\mathcal{O}}|\mathcal{M}) = \frac{1}{R}\sum_{r=1}^{R} \log P_\lambda(\mathcal{O}_r|\mathcal{M}) \tag{B.1}$$

For a given model $\mathcal{M}$ we proceed to derive the partial derivatives of $\mathcal{L}_\lambda(\mathcal{O}_r|\mathcal{M})$ with respect to each individual HMM parameter. The computation of these derivatives is facilitated by the chain rule and the differentiation rules given in appendix C. Differentiating equation B.1 gives

$$\frac{\partial}{\partial\lambda}\mathcal{L}_\lambda(\boldsymbol{\mathcal{O}}|\mathcal{M}) = \frac{1}{R}\sum_{r=1}^{R}\frac{1}{P_\lambda(\mathcal{O}_r|\mathcal{M})}\frac{\partial}{\partial\lambda}P_\lambda(\mathcal{O}_r|\mathcal{M}) \tag{B.2}$$

The HMM parameter set λ is partitioned into transition parameters, mixture weights, Gaussian mean vectors and variance vectors/covariances matrices. Using these parameters, $P_\lambda(\mathcal{O}_r|\mathcal{M})$ can be computed as

$$P_\lambda(\mathcal{O}_r|\mathcal{M}) = \sum_{t=1}^{T_r}\sum_{j=1}^{N}\alpha_j(t)\beta_j(t) \tag{B.3}$$

where $\alpha_j(t)$ and $\beta_j(t)$ are the forward and backward probabilities of state j, time t (see section 2.2.3). In order to expose the individual model parameters, the above expression can be reformulated as

$$P_\lambda(\mathcal{O}_r|\mathcal{M}) = \sum_{t=1}^{T_r}\sum_{j=1}^{N}\left\{\sum_{i=1}^{N}\alpha_i(t-1)a_{i,j}\right\}b_j(\boldsymbol{o}_{r,t})\beta_j(t) \tag{B.4}$$

For the transition parameters at state i we have

$$\frac{\partial}{\partial a_{i,j}}P_\lambda(\mathcal{O}_r|\mathcal{M}) = \sum_{t=1}^{T_r}\alpha_i(t-1)b_j(\boldsymbol{o}_{r,t})\beta_j(t) \tag{B.5}$$

The probabilistic interpretation of $a_{i,j}$ is maintained by the following two constraints

$$0 \leq a_{i,j} \leq 1 \qquad \text{and} \qquad \sum_{j=1}^{N} a_{i,j} = 1$$

After updating at each iteration, a simple way to enforce these constraints is to set any negative transitions to zero and then re-normalise to ensure the sum-to-one constraint. An obvious problem occurs if all transitions have been reduced to zero prior to normalisation. A more robust method of enforcing the stochastic constraints is to transform the constrained set $\{a_{i,j}\}$ into an unconstrained set $\{h_{i,j}\}$, via a mapping of the form

$$a_{i,j} = \frac{f_a(h_{i,j})}{\sum_k f_a(h_{i,k})} \tag{B.6}$$

where $f_a(x)$ is a monotonically increasing function. A possible choice is to use $f_a(x) = e^x$, as discussed in [19], and optimise the objective function with respect to $\{h_{i,k}\}$. Differentiating equation B.6 produces

$$\frac{\partial}{\partial h_{i,k}} a_{i,j} = a_{i,j}(\delta_{k,j} - a_{i,k}) \tag{B.7}$$

where $\delta_{k,j}$ is the Kronecker delta. Using the chain rule

$$\frac{\partial}{\partial h_{i,k}} P_\lambda(\mathcal{O}_r|\mathcal{M}) = \sum_j \frac{\partial}{\partial a_{i,j}} P_\lambda(\mathcal{O}_r|\mathcal{M}) \frac{\partial}{\partial h_{i,k}} a_{i,j} \tag{B.8}$$

Substituting equations B.5 and B.7 into equation B.8 yields

$$\frac{\partial}{\partial h_{i,k}} P_\lambda(\mathcal{O}_r|\mathcal{M}) = \sum_{t=1}^{T_r} \sum_{j=1}^{N} \alpha_i(t-1) a_{i,j}(\delta_{k,j} - a_{i,k}) b_j(\boldsymbol{o}_{r,t}) \beta_j(t) \tag{B.9}$$

For the output distribution $b_j(\boldsymbol{o}_{r,t})$ we have

$$\frac{\partial}{\partial b_j(\boldsymbol{o}_{r,t})} P_\lambda(\boldsymbol{\mathcal{O}}|\mathcal{M}) = \sum_{t=1}^{T_r} \sum_{j=1}^{N} \left\{ \sum_{i=1}^{N} \alpha_i(t-1) a_{i,j} \right\} \beta_j(t) \tag{B.10}$$

For continuous mixture densities (section 2.2.2)

$$b_j(\boldsymbol{o}_{r,t}) = \sum_{m=1}^{M} c_{j,m} b_{j,m}(\boldsymbol{o}_{r,t}) \tag{B.11}$$

where

$$b_{j,m}(\boldsymbol{o}_{r,t}) = \frac{1}{(2\pi)^{D/2} |\boldsymbol{W}_{j,m}|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{o}_{r,t} - \boldsymbol{\mu}_{j,m})' \boldsymbol{W}_{j,m}^{-1} (\boldsymbol{o}_{r,t} - \boldsymbol{\mu}_{j,m})} \tag{B.12}$$

where $\boldsymbol{\mu}_{j,m}$, $\boldsymbol{W}_{j,m}$ and $c_{j,m}$ are the mean vector, covariance matrix and mixture weight respectively for the m^{th} mixture components at state j. Differentiating equation B.11 gives

$$\frac{\partial}{\partial b_{j,m}(\boldsymbol{o}_{r,t})} b_j(\boldsymbol{o}_{r,t}) = c_{j,m} \tag{B.13}$$

Differentiating equation B.12 with respect to the mean vector $\boldsymbol{\mu}_{j,m}$

$$\frac{\partial}{\partial \boldsymbol{\mu}_{j,m}} b_{j,m}(\boldsymbol{o}_{r,t}) = b_{j,m}(\boldsymbol{o}_{r,t}) \boldsymbol{W}_{j,m}^{-1}(\boldsymbol{o}_{r,t} - \boldsymbol{\mu}_{j,m}) \tag{B.14}$$

When $\boldsymbol{W}_{j,m}$ is diagonal, e.g.

$$\boldsymbol{W}_{j,m} = \begin{pmatrix} \sigma^2_{j,m,1} & 0 & 0 & \vdots & 0 \\ 0 & \sigma^2_{j,m,2} & 0 & \vdots & 0 \\ 0 & 0 & \sigma^2_{j,m,3} & \vdots & 0 \\ \dots & \dots & \dots & \ddots & \dots \\ 0 & 0 & 0 & \vdots & \sigma^2_{j,m,D} \end{pmatrix} \tag{B.15}$$

we can reformulate equation B.14 as

$$\frac{\partial}{\partial \mu_{j,m,d}} b_{j,m}(\boldsymbol{o}_{r,t}) = b_{j,m}(\boldsymbol{o}_{r,t}) \left\{ \frac{(o_{r,t,d} - \mu_{j,m,d})}{\sigma^2_{j,m,d}} \right\} \tag{B.16}$$

where $\sigma_{j,m,d}$ is the standard deviation of the d^{th} component in the observation vector. Similarly, differentiating equation B.12 with respect to $\boldsymbol{W}_{j,m}^{-1}$ yields

$$\begin{aligned} &\frac{\partial}{\partial \boldsymbol{W}_{j,m}^{-1}} b_{j,m}(\boldsymbol{o}_{r,t}) = \\ &= \frac{1}{2} b_{j,m}(\boldsymbol{o}_{r,t}) \boldsymbol{W}_{j,m} - \frac{1}{2} b_{j,m}(\boldsymbol{o}_{r,t}) \left[(\boldsymbol{o}_{r,t} - \boldsymbol{\mu}_{j,m})(\boldsymbol{o}_{r,t} - \boldsymbol{\mu}_{j,m})' \right] \\ &= b_{j,m}(\boldsymbol{o}_{r,t}) \frac{1}{2} (\boldsymbol{W}_{j,m} - (\boldsymbol{o}_{r,t} - \boldsymbol{\mu}_{j,m})(\boldsymbol{o}_{r,t} - \boldsymbol{\mu}_{j,m})') \end{aligned} \tag{B.17}$$

The covariance matrix $\boldsymbol{W}$ is positive definite. In order to ensure this, we use Choleski decomposition

$$\boldsymbol{W}_{j,m}^{-1} = \boldsymbol{L}_{j,m} \boldsymbol{L}'_{j,m}$$

and perform updates on the lower Choleski factor $\boldsymbol{L}_{j,m}$. Hence the derivative

$$\frac{\partial}{\partial \boldsymbol{L}_{j,m}} b_{j,m}(\boldsymbol{o}_{r,t}) = b_{j,m}(\boldsymbol{o}_{r,t}) \left\{ (\boldsymbol{L}_{j,m}^{-1})' - (\boldsymbol{o}_{r,t} - \boldsymbol{\mu}_{j,m})(\boldsymbol{o}_{r,t} - \boldsymbol{\mu}_{j,m})' \boldsymbol{L}_{j,m} \right\} \tag{B.18}$$

Since $\boldsymbol{L}_{j,m}$ is lower triangular, the elements above the leading diagonal in $(\boldsymbol{L}_{j,m}^{-1})'$ are not needed, hence

$$(\boldsymbol{L}_{j,m}^{-1})' = diag\left\{ \frac{1}{L_{11}}, \frac{1}{L_{22}}, \dots, \frac{1}{L_{dd}}, \dots, \frac{1}{L_{DD}} \right\}$$

where L_{dd} are the diagonal elements of $\boldsymbol{L}_{j,m}$. When $\boldsymbol{W}$ is diagonal (equation B.15), we have

$$\frac{\partial}{\partial \sigma^2_{j,m,d}} b_{j,m}(\boldsymbol{o}_{r,t}) = b_{j,m}(\boldsymbol{o}_{r,t}) \frac{1}{2} \left\{ \frac{(o_{r,t,d} - \mu_{j,m,d})^2}{(\sigma^2_{j,m,d})^2} - \frac{1}{\sigma^2_{j,m,d}} \right\} \tag{B.19}$$

In the diagonal case we have to enforce the constraint that all variances are positive. In the case of transition probabilities we used a mapping which transformed the constrained set into an unconstrained set. Similarly, for the variance parameters we will apply a transformation function

$$\sigma^2_{j,m,d} = f_{\sigma^2}(z_{j,m,d}) \tag{B.20}$$

Young [111] suggests using $f_{\sigma^2}(x) = x^2$ and an alternative choice is $f_{\sigma^2}(x) = e^x$.The latter is preferable since it results in a simpler expression. Differentiating equation B.20 and combining with equation B.19 yields

$$\begin{aligned}
\frac{\partial}{\partial z_{j,m,d}} b_{j,m}(\boldsymbol{o}_{r,t}) &= \frac{\partial}{\partial \sigma^2_{j,m,d}} b_{j,m}(\boldsymbol{o}_{r,t}) \frac{\partial}{\partial z_{j,m,d}} \sigma^2_{j,m,d} \\
&= b_{j,m}(\boldsymbol{o}_{r,t}) \frac{1}{2} \left\{ \frac{(o_{r,t,d} - \mu_{j,m,d})^2}{(\sigma^2_{j,m,d})^2} - \frac{1}{\sigma^2_{j,m,d}} \right\} \sigma^2_{j,m,d} \\
&= b_{j,m}(\boldsymbol{o}_{r,t}) \frac{1}{2} \left\{ \frac{(o_{r,t,d} - \mu_{j,m,d})^2}{\sigma^2_{j,m,d}} - 1 \right\} \qquad \text{(B.21)}
\end{aligned}$$

For the mixture weights $c_{j,m}$ we have

$$\frac{\partial}{\partial c_{j,m}} b_j(\boldsymbol{o}_{r,t}) = b_{j,m}(\boldsymbol{o}_{r,t})$$

Subject to constraints

$$0 \le c_{j,m} \le 1 \qquad \text{and} \qquad \sum_{m=1}^{M} c_{j,m} = 1$$

Similar to the transition parameters, we define a mapping

$$c_{j,m} = \frac{f_c(u_{j,m})}{\sum_k f_c(u_{j,k})} \tag{B.22}$$

where $f_c(x) = e^x$. Differentiating equation B.22

$$\frac{\partial}{\partial u_{j,k}} c_{j,m} = c_{j,m}(\delta_{k,m} - c_{j,k}) \tag{B.23}$$

Hence

$$\begin{aligned}
\frac{\partial}{\partial u_{j,k}} b_j(\boldsymbol{o}_{r,t}) &= \sum_{m=1}^{M} \frac{\partial}{\partial c_{j,m}} b_j(\boldsymbol{o}_{r,t}) \frac{\partial}{\partial u_{jk}} c_{j,m} \\
&= \sum_{m=1}^{M} b_{j,m}(\boldsymbol{o}_{r,t}) c_{j,m}(\delta_{k,m} - c_{j,k}) \qquad \text{(B.24)}
\end{aligned}$$

The final derivative expression for different HMM parameters and multiple training utterances are given in section 4.4.1.

Appendix C

Differentiation Rules

This appendix presents a list of differentiation rules used to calculate the derivatives of the likelihood function (appendix B) with respect to the HMM parameters.

C.1 Notation

$\boldsymbol{W}$	a symmetric positive definite matrix with elements W_{ij}.
$(\boldsymbol{W}^{-1})$	the inverse of $\boldsymbol{W}$ with elements $(\boldsymbol{W}^{-1})_{ij}$.
$\boldsymbol{W'}$	the transpose of $\boldsymbol{W}$ with elements W'_{ij}.
$\lvert\boldsymbol{W}\rvert$	the determinant of $\boldsymbol{W}$.
$cof_{ij}(\boldsymbol{W})$	the cofactor of W_{ij} in $\boldsymbol{W}$.
$\boldsymbol{L}$	the lower Choleski factor of $\boldsymbol{W}$, e.g. $\boldsymbol{W} = \boldsymbol{LL'}$, with elemnts L_{ij}.
$\boldsymbol{A}$	a matrix with elements A_{ij}.
$\boldsymbol{u}, \boldsymbol{v}$	column vectors with elements u_i and v_i respectively.

C.2 Rules

1. Let $f(\boldsymbol{W}) = |\boldsymbol{W}|$, find $\partial f(\boldsymbol{W})/\partial \boldsymbol{W}$.

$$\begin{aligned}
|\boldsymbol{W}| &= \sum_{ij} cof_{ij}(\boldsymbol{W}) W_{ij} \\
(\boldsymbol{W}^{-1})_{ij} &= \frac{cof_{ji}(\boldsymbol{W})}{|\boldsymbol{W}|} \\
\frac{\partial}{\partial W_{ij}} f(\boldsymbol{W}) &= cof_{ij}(\boldsymbol{W}) = |\boldsymbol{W}|(\boldsymbol{W}^{-1})'_{ij} \\
\frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W}) &= |\boldsymbol{W}|(\boldsymbol{W}^{-1})' = |\boldsymbol{W}|(\boldsymbol{W}^{-1})
\end{aligned}$$

2. Let $f(\boldsymbol{L}) = |\boldsymbol{L}\boldsymbol{L}'|$, find $\partial f(\boldsymbol{L})/\partial \boldsymbol{L}$.

$$\begin{aligned} f(\boldsymbol{L}) &= |\boldsymbol{L}||\boldsymbol{L}'| = |\boldsymbol{L}|^2 \\ \frac{\partial}{\partial \boldsymbol{L}}|\boldsymbol{L}| &= |\boldsymbol{L}|(\boldsymbol{L}^{-1})' \\ \frac{\partial}{\partial \boldsymbol{L}} f(\boldsymbol{L}) &= 2|\boldsymbol{W}|(\boldsymbol{L}^{-1})' \end{aligned}$$

3. Let $f(\boldsymbol{v}) = \boldsymbol{v}'\boldsymbol{W}\boldsymbol{v}$, find $\partial f(\boldsymbol{v})/\partial \boldsymbol{v}$.

$$\begin{aligned} \frac{\partial}{\partial v_k} f(\boldsymbol{v}) &= 2\sum_{i,j} \frac{\partial}{\partial v_k} v_i W_{ij} v_j = 2\sum_j W_{k,j} v_j \\ \frac{\partial}{\partial \boldsymbol{v}} f(\boldsymbol{v}) &= 2\boldsymbol{W}\boldsymbol{v} \end{aligned}$$

4. Let $f(\boldsymbol{W}) = \boldsymbol{u}'\boldsymbol{W}\boldsymbol{v}$, find $\partial f(\boldsymbol{W})/\partial \boldsymbol{W}$.

$$\begin{aligned} f(\boldsymbol{W}) &= \sum_{i,j} u_i W_{ij} v_j \\ \frac{\partial}{\partial W_{ij}} f(\boldsymbol{W}) &= u_i v_j \\ \frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W}) &= \boldsymbol{u}\boldsymbol{v}' \end{aligned}$$

5. Let $f(\boldsymbol{L}) = \boldsymbol{u}'\boldsymbol{L}\boldsymbol{L}'\boldsymbol{v}$, find $\partial f(\boldsymbol{L})/\partial \boldsymbol{L}$.

$$\begin{aligned} \boldsymbol{u}'\boldsymbol{L}\boldsymbol{L}'\boldsymbol{v} &= \sum_j \left[\sum_i u_i L_{ij}\right]\left[\sum_i v_i L_{ij}\right] \\ \frac{\partial}{\partial L_{ik}} f(\boldsymbol{L}) &= u_i \sum_j v_j L_{jk} + v_i \sum_j u_j L_{jk} \\ \frac{\partial}{\partial \boldsymbol{L}} f(\boldsymbol{L}) &= \boldsymbol{u}\boldsymbol{v}'\boldsymbol{L} + \boldsymbol{v}\boldsymbol{u}'\boldsymbol{L} \end{aligned}$$

6. Let $f(\boldsymbol{v}) = (\boldsymbol{A}\boldsymbol{v} + \boldsymbol{u})'\boldsymbol{W}(\boldsymbol{A}\boldsymbol{v} + \boldsymbol{u})$, find $\partial f(\boldsymbol{v})/\partial \boldsymbol{v}$.

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{v}_k} f(\boldsymbol{v}) &= 2\sum_{ij} \frac{\partial}{\partial v_k}(\boldsymbol{A}\boldsymbol{v} + \boldsymbol{u})_i W_{ij} (\boldsymbol{A}\boldsymbol{v} + \boldsymbol{u})_j \\ &= 2\sum_{ij} A_{ik} W_{ij} (\boldsymbol{A}\boldsymbol{v} + \boldsymbol{u})_j \\ &= 2\sum_{ij} A'_{ki} W_{ij} (\boldsymbol{A}\boldsymbol{v} + \boldsymbol{u})_j \\ \frac{\partial}{\partial \boldsymbol{v}} f(\boldsymbol{v}) &= 2\boldsymbol{A}'\boldsymbol{W}(\boldsymbol{A}\boldsymbol{v} + \boldsymbol{u}) \end{aligned}$$

Appendix D

Transformation Definitions

In HTK [117], HMM definitions reside in external files. The HMM structure and parameters are described using a definition language. Each HMM can reside in a separate file. Alternatively, all HMMs used in an application can be stored together in a single Master Model File (MMF). Parameters shared amongst several HMMs are kept in a separate "macro" file. Processing of HMM files and associated macro files is carried out using the `HModel` library module from HTK. This appendix describes the extended HMM definition language which is used to define the input transformations.

D.1 New features

The HTK support for hidden Markov models was enhanced to incorporate the following additional features

- Each emitting state in an HMM can have an associated input transformation which transforms the input vector before it enters the output distribution for that state.
- Each transformation consists of a single or multiple layers which computed pre-selected output functions.
- Transformations can incorporate recurrent connections implementing unit time delay.
- The output of the transformation can be used to directly provide the probability of the input vector.
- Transformations can be shared amongst states and layers can be shared between multiple transformations.

D.2 Definition language

This section describes the added syntax to the HMM definition language for the definition of the input transformations. Following conventions in the HTK [117], the syntax is described

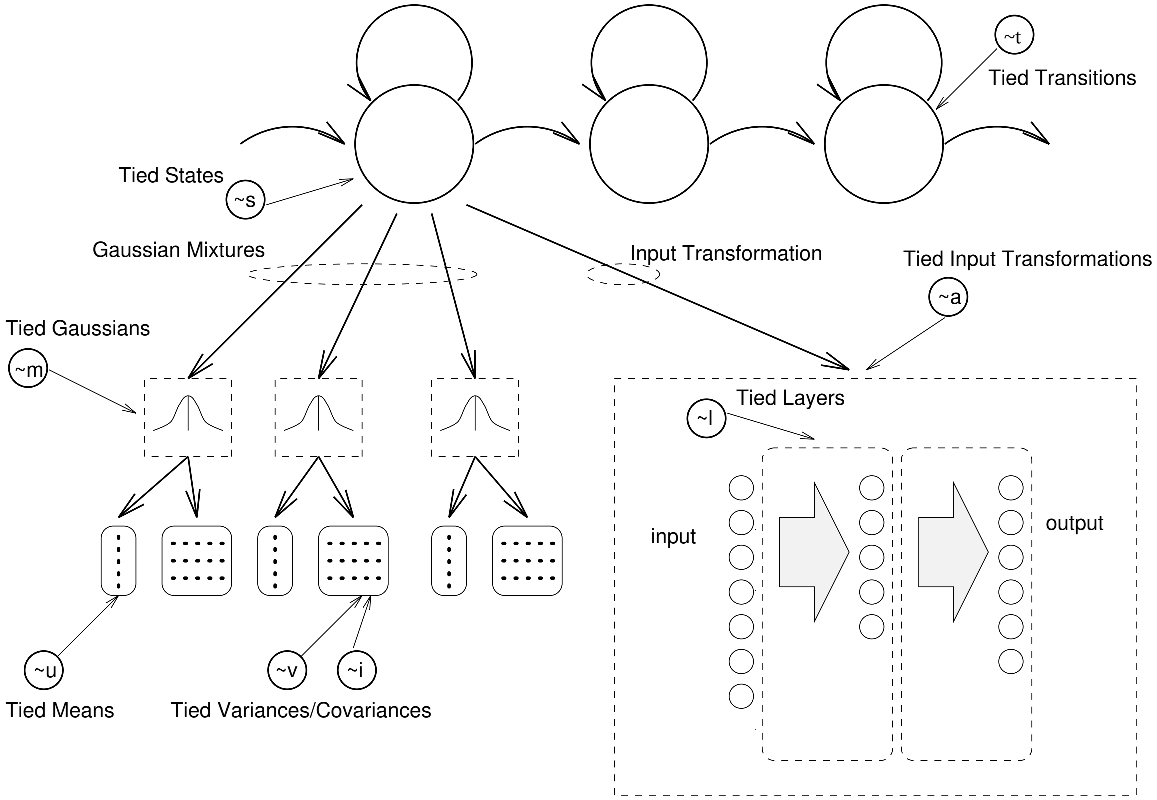

Figure D.1: Enhanced HMM parameter hierarchy and potential sharing points

using an extended BNF notation in which alternatives are separated by a vertical bar |, brackets [ ] denote options, and braces { } denote zero or more repetitions. The top level syntax of an HMM definition is given by the following rule

```
hmmdef =    <BeginHMM>
               [<Use> "macrofilename"]
               { option } { mlp-option }
               state {state}
               transP
               [duration]
            <EndHMM>
```

The extra options provided by `{ mlp-option }` describe the utility of the input transformations for the given HMM according to the following rule

```
mlp-option =    <mlpEnable> | <mlpDirect>
```

The presence of `<mlpEnable>` enables the transformations and `<mlpDirect>` allows the output of the transform to bypass the corresponding output distribution and become the emission probability for the parent HMM state. In the absence of the latter option, the output from the transform will be fed to the output distribution of the corresponding state which, in turn, will provide the probability of generating the observation vector.

Each HMM state has its own transformation definition. The definition of a state is given by

```
stateinfo =    ~s macro | [mixes] [weights]
                stream {stream} [duration] [mlpdef]
```

The structure of the transformation and its parameters are described by the syntax

```
mlpdef =    ~a macro | <BeginMLP>
                          {mlp-struct}
                          layerdef {layerdef}
                        <EndMLP}
```

The second part of the rule is the usual definition of an input transformation. However, as an alternative, a `~a macro` can be used to select a transformation macro definition from the loaded macro file.

The MLP structure is described in terms of the number of layers, the number of input units, and and the number of recurrent connections if any.

```
mlp-option =    <numLayers> int | <numInUnits> int | <numRecurrent>  int
```

If these parameters are not specified, the number of input units will be set to the observation vector size, the number of layers[1] will be 2 and no recurrent connections will be setup. If non-zero, the number of recurrent units will be used to setup unit time delay connections between the last n units of the output layer of the and the last n units of the first layer. The definition for each layer consists of the number of units, function type, the connection matrix between the current layer and the previous one, and the bias vector. This is given by the syntax

```
layerdef =    ~l macro | <Layer>
                            <numUnits> int int functype
                            <weights> matrix
                            <biases> vector
```

Three self-explanatory function types are supported

```
functype =    <Linear> | <Quadratic> | <Sigmoid>
```

D.3 Example definition

The following is an example definition of a single layer recurrent input transformation with 13 inputs, 4 outputs and 4 recurrent units.

[1] All transformations must have at least 2 layer. The first layer of the transformation is implicitly defined as the input observation vector.

```
~a "theMLP"
<BeginMLP>
  <numLayers> 2 <numInUnits> 13 <numRecurrent> 4
  <Layer> 2
    <numUnits> 8 <LINEAR>
    <Weights>
       1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
       0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0
       0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0
       0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0
       0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
       0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
       0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
       0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
       0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
       0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
       0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
       0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
       0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
       1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
       0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
       0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
       0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
    <Biases>
       0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<EndMLP>
```

The transformation is initialised to output the first four components from $(\boldsymbol{o}_t + \boldsymbol{o}_{t-1})$.

Bibliography

[1] F.S. Acton. *Numerical Methods that Work*. Harper & Row, Publishers, 1969.

[2] F. Alleva, H.-W. Hon, X. Huang, M. Hwang, R. Rosenfield, and R. Weide. Applying SPHINX-II to the DARPA Wall Street Journal CSR Task. In *Proc. of DARPA Speech and Natural Language Workshop*. DARPA, February 1992.

[3] S. Austin, R. Schwartz, and P. Placeway. The Forward-Backward Search Algorithm. In *Proc. ICASSP'91*, volume 1, pages 697–700. IEEE, May 1991.

[4] C.M. Ayer. *A Discriminatively Derived Linear Transform Capable of Improving Speech Recognition Accuracy*. PhD thesis, University of London, 1992.

[5] C.M. Ayer, M.J. Hunt, and D.M. Brookes. A Discriminatively Derived Linear Transform for Improved Speech Recognition. In *Proc. Eurospeech*, Berlin, September 1993.

[6] L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer. Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition. In *Proc. ICASSP'86*. IEEE, 1986.

[7] L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer. A New Algorithm for the Estimation of Hidden Markov Model Parameters. In *Proc. ICASSP'88*, April 1988.

[8] J.K. Baker. The DRAGON System - An Overview. *IEEE transactions on Acoustics, Speech and Signal Processing*, ASSP-23(1):24–29, February 1975.

[9] D.C. Bateman and M.J. Hunt. Visualisation of an IMELDA Representation. In *Proc. ESCA Workshop on "Comparing Speech Signal Representations"*, pages 91–95, Sheffield, UK, 1992.

[10] L.E. Baum. An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes. *Inequalities*, 3(1):1–8, 1972.

[11] L.E. Baum and J.A. Eagon. An Inequality with Applications to Statistical Estimation for Probabilistic Functions of Markov Processes and to a Model for Ecology. *Bull. Amer. Math. Soc.*, 73:360–363, May 1967.

[12] L.E. Baum, T. Petrie, G. Soules, and N. Weiss. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Ann. Math. Stat.*, 41(1):164–171, 1970.

[13] J.R. Bellegarda and D. Nahamoo. Tied Mixture Continuous Parameter Modeling for Speech Recognition. *Transactions on Acoustics, Speech and Signal Processing*, 38:2033–2045, 1990.

[14] Y. Bengio, R. Cardin, R. De Mori, and Y. Normandin. A Hybrid Coder for Hidden Markov Models using a Recurrent Neural Network. In *Proc. ICASSP'90*, pages 537–540, April 1990.

[15] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe. Global Optimisation of a Neural Network - Hidden Markov Model Hybrid. Technical Report TR-SOCS-90.22, McGill University School of Computer Science, 3480 University street, H3A2A7, Montreal, Qc., Canada, December 1990.

[16] E.L. Bocchieri and G.R. Doddington. Frame-Specific Statistical Features for Speaker Independent Speech Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-34(4):755–764, August 1986.

[17] E.L. Bocchieri and J.G. Wilpon. Discriminative Feature Selection for Speech Recognition. *Computer Speech and Language*, 1993.

[18] J.S. Bridle. Alpha-Nets: A Recurrent "Neural" Network Architecture with a Hidden Markov Model Interpretation. Technical report, Speech Research Unit, RSRE, St. Andrew's Road, Great Malvern, UK, October 1989.

[19] J.S. Bridle. Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationship to Statistical Pattern Recognition. In F. Ed Fougleman-Soulie and J. Herault, editors, *Neuro-computing: algorithms, architectures and applications.* Springer-Verlag, 1989.

[20] J.S. Bridle and L. Dodd. An Alphanet Approach to Optimising Input Transformations for Continuous Speech Recognition. In *Proc. ICASSP'91*, volume 1, pages 277–280, May 1991.

[21] J.S. Bridle, P. Nowell, and L. Dodd. An Investigation into Discriminative Training of Input Transformations for Continuous Speech Recognition. In *Proc. I.O.A.*, volume 14, pages 523–528, 1992.

[22] P.F. Brown. The Acoustic-Modelling Problem in Automatic Speech Recognition. Technical report, IBM Thomas J. Watson Research Centre, August 1987.

[23] R. Cardin, Y. Normandin, and R. De Mori. High Performance Connected Digit Recognition using Maximum Mutual Information Estimation. In *Proc. ICASSP'91*, volume 1, pages 533–536. IEEE, May 1991.

[24] R. Cardin, Y. Normandin, and R. De Mori. High Performance Connected Digit Recognition using Codebook Exponents. In *Proc. ICASSP'92*, volume 1, pages 505–508. IEEE, 1992.

[25] W. Chou, B.H. Juang, and C.H. Lee. Segmental GPD Training of HMM based Speech Recognizer. In *Proc. ICASSP'92*, volume 1, pages 473–476. IEEE, 1992.

[26] Y.L. Chow. Maximum Mutual Information Estimation of HMM Parameters for Continuous Speech Recognition using The N-Best Algorithm. In *Proc. ICASSP'90*. IEEE, 1990.

[27] R. Cole and M. Fanty. Spoken Letter Recognition. In *Proc. Third DARPA Speech and Natural Language Workshop*, Somerset, PA, June 1990.

[28] R. Cole, M. Fanty, Y. Muthusamy, and M. Gopalakrishnan. Speaker-Independent Recognition of Spoken English Letters. In *Proc. of the International Joint Conference on Neural Networks*, San Diego, CA, June 1990.

[29] R.A. Cole, R.M. Stern, and M.J. Lasry. Performing Fine Phonetic Distinctions: Templates vs. Features. In J. Perkell and D. Klatt, editors, *Invariance and Variability of Speech Processes*. Lawrence Erlbaum, New York, 1984.

[30] R.A. Cole, R.M. Stern, M.S. Phillips, S.M. Brill, A.P. Pilant, and P. Specker. Feature-Based Speaker-Independent Recognition of Isolated English Letters. In *Proc. ICASSP'83*, pages 731–734. IEEE, April 1983.

[31] A.P. Dempster, N.M. Larid, and D.B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[32] G.R. Doddington. Phonetically Sensitive Discriminants for Improved Speech Recognition. In *Proc. ICASSP'89*, volume 1, pages 556–559. IEEE, 1989.

[33] G.R. Doddington. Frame-Specific Variance Weighting for Improved Speech Recognition. In *Proc. ICASSP'90*. IEEE, 1990.

[34] Y. Ephraim, A. Dembo, and L.R. Rabiner. A Minimum Discrimination Information Approach for Hidden Markov Modeling. In *Proc. ICASSP'87*, April 1987.

[35] Y. Ephraim, A. Dembo, and L.R. Rabiner. A Minimum Discrimination Information Approach for Hidden Markov Modeling. *IEEE Transactions on Information Theory*, 35(5), September 1989.

[36] Y. Ephraim and L.R. Rabiner. On the Relations Between Modeling Approaches for Information Sources. In *Proc. ICASSP'88*, April 1988.

[37] Y. Ephraim and L.R. Rabiner. On the Relations Between Modeling Approaches for Speech Recognition. *IEEE Transactions on Information Theory*, 36(2), March 1990.

[38] S. Euler and J. Zinke. Experiments on the Use of the Generalized Probabilistic Descent Method in Speech Recognition. In *Proc. ICSLP'92*, volume 1, pages 157–160, Banff, Canada, October 1992.

[39] S.A. Euler, B.H. Juang, C.H. Lee, and F.K. Soong. Statistical Segmentation and Word Modeling Techniques in Isolated Word Recognition. In *Proc. ICASSP'90*. IEEE, 1990.

[40] S.E. Fahlman. An Empirical Study of Learning Speed in Back-Propagation Networks. Technical report, CMU, September 1988.

[41] M. Fanty and R. Cole. Speaker-Independent English Alphabet Recognition: Experiments with the E-Set. In *Proc. of the 1990 International Conference on Spoken Language Processing*, Kobe, Japan, November 1990.

[42] M. Fanty and R. Cole. Spoken Letter Recognition. In *Proc. of the Neural Information Processing System Conference*, November 1990.

[43] H. Franco and A. Serralheiro. Training HMMs using a Minimum Error Approach. In *Proc. ICASSP'91*. IEEE, 1990.

[44] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, 1972.

[45] S. Furui. Speaker-Independent Isolated Word Recognition using Dynamic Features of Speech Spectrum. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-34(1), February 1986.

[46] P.S. Gopalakrishnan, D. Kanevsky, A. Nádas, and D. Nahamoo. An Inequality for Rational Functions with Applications to Some Statistical Estimation Problems. *IEEE Transactions on Information Theory*, 37(1), January 1991.

[47] P.S. Gopalakrishnan, D. Kanevsky, A. Nádas, D. Nahamoo, and M.A. Picheny. Decoder Selection Based on Cross-Entropies. In *Proc. ICASSP'88*, 1988.

[48] M.M. Hochberg, G.D. Cook, S.J. Renals, A.J. Robinson, and R.S. Schechtman. The 1994 ABBOT Hybrid Connectionist-HMM Large-Vocabulary Recognition System. In *Proc. of ARPA Spoken Language Technology Workshop*, Austin, Texas, January 1995. ARPA.

[49] M.M. Hochberg, L.T. Niles, J.T. Foote, and H.F. Silverman. Hidden Markov Model/Neural Network Training Techniques for Connected Alphadigit Speech Recognition. In *Proc ICASSP'91*, pages 109–112, 1991.

[50] X.D. Huang and M.A. Jack. Semi-Continuous Hidden Markov Models for Speech Signals. *Computer Speech and Language*, 3, July 1989.

[51] X.D. Huang, K.-F. Lee, H.-W. Hon, and M.Y. Hwang. Improved Acoustic Modelling with the SPHINX Speech Recognition System. In *Proc ICASSP'91*, volume 1, pages 345–348. IEEE, 1991.

[52] M.J. Hunt and C. Lefebvre. Speaker Dependent and Independent Speech Recognition Experiments with an Auditory Model. In *Proc. ICASSP'88*, volume 1, pages 215–218. IEEE, 1988.

[53] M.J. Hunt and C. Lefebvre. A Comparison of Several Acoustic Representations for Speech Recognition with Degraded and Undegraded Speech. In *Proc ICASSP'89*, volume 1, pages 262–265. IEEE, 1989.

[54] M.J. Hunt, S.M. Richardson, D.C. Bateman, and A. Piau. An Investigation of PLP and IMELDA Acoustic Representations and of their Potential for Combination. In *Proc. ICASSP'91*, volume 2, pages 881–884. IEEE, 1991.

[55] R.A. Jacobs. Increased Rate of Convergence Through Learning Rate Adaptation. *Neural Networks*, 1:295–307, 1988.

[56] F. Jelinek. Continuous Speech Recognition by Statistical Methods. *Proc. IEEE*, 64(4), April 1976.

[57] F.T. Johansen and M.H. Johnsen. Non-Linear Input Transformations for Discriminative HMMs. In *Proc. ICASSP'94*, volume 1, pages 225–228. IEEE, 1994.

[58] B.-H. Juang. Maximum-Likelihood Estimation for Mixture Multivariate Stochastic Observations of Markov Chains. *AT&T Technical Journal*, 64(6):1235–1249, July-August 1985.

[59] B.-H. Juang and S. Katagiri. Discriminative Training. *Journal of the Acoustical Society of Japan*, 13(6):333–339, 1992.

[60] B.-H. Juang and L.R. Rabiner. Mixture Autoregressive Hidden Markov Models for Automatic Speech Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-33(6):1404–1413, December 1985.

[61] S. Kapadia. On the organisation of discriminative training procedures. Personal communication, 1991.

[62] S.M. Katz. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35(3):400–401, March 1987.

[63] J.F. Kolen and J.B. Pollack. Backpropagation is Sensitive to Initial Conditions. *Complex Systems*, 4(3):269–280, 1990.

[64] F. Kubala, S. Austin, C. Barry, J. Makhoul, P. Placeway, and R. Schwartz. BYBLOS Speech Recognition Benchmark Results. In *Proc. of the DARPA Speech Recognition Workshop*. DARPA, February 1991.

[65] L.F. Lamel, R.H. Kasel, and S. Seneff. Speech Database Development: Design and Analysis of the Acoustic-Phonetic Corpus. In *Proc. of the DARPA Speech Recognition Workshop*, pages 26–32, 1987.

[66] K.-F. Lee. *Automatic Speech Recognition, The Development of the SPHINX System*. Kluwer Academic Publishers, 1989.

[67] K.-F. Lee and H.-W. Hon. Speaker-Independent Phone Recognition using Hidden Markov Models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(11):1641–1648, 1989.

[68] K.-F. Lee, H.-W. Hon, M.Y. Hwang, S. Mahajan, and R. Reddy. The SPHINX Speech Recognition System. In *Proc. ICASSP'89*. IEEE, April 1989.

[69] K.-F. Lee, H.-W. Hon, and R. Reddy. An Overview of the SPHINX Speech Recognition System. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, January 1990.

[70] K.-F. Lee and S. Mahajan. Corrective and Reinforcement Learning for Speaker-Independent Continuous Speech Recognition. *Computer Speech and Language*, 4, 1990.

[71] S.E. Levinson, L.R. Rabiner, and M.M. Sondhi. An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition. *The Bell Systems Technical Journal*, 62(4), April 1983.

[72] L.A. Liporace. Maximum Likelihood Estimation for Multivariate Observations of Markov Sources. *IEEE Transactions on Information Theory*, 28(5):729–734, September 1982.

[73] A. Ljolje. New Developments in Phone Recognition using an Ergodic Hidden Markov Model. Technical Report TM-11222-910829-12, AT & T, Bell Laboratories, 1991.

[74] A. Ljolje, Y. Ephraim, and L.R. Rabiner. Estimation of Hidden Markov Model Parameters by Minimizing Empirical Error Rate. In *Proc. ICASSP'90*, April 1990.

[75] N.A. McCulloch. *Neural Network Approaches to Speech Recognition & Synthesis*. PhD thesis, Department of Communication and Neuroscience, Keele University, June 1990.

[76] B. Merialdo. Phonetic Recognition using Hidden Markov Models and Maximum Mutual Information Training. In *Proc. ICASSP'88*, April 1988.

[77] N. Morgan and H. Bourlard. Continuous Speech Recognition using Multilayer Perceptrons with Hidden Markov Models. In *Proc. ICASSP'90*, pages 413–416, April 1990.

[78] A. Nádas. A Decision Theoretic Formulation of a Training Problem in Speech Recognition and a Comparison of Training by Unconditional Versus Conditional Maximum Likelihood. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 31(4), August 1983.

[79] A. Nádas, R. Mercer, L. Bahl, R. Bakis, P.S. Cohen, A.G. Cole, F. Jelinek, and B.L. Lewis. Continuous Speech Recognition with Automatically Selected Acoustic Prototypes Obtained by Either Bootstrapping or Clustering. In *Proc. ICASSP'81*. IEEE, 1991.

[80] A. Nádas, D. Nahamoo, and M.A. Picheny. On a Model-Robust Training Method for Speech Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(9), September 1988.

[81] R. Naeb-Umbach and H. Ney. Linear Discriminant Analysis for Improved Large Vocabulary Continuous Speech Recognition. In *Proc. ICASSP'92*, volume 1, pages 13–16. IEEE, March 1992.

[82] H. Ney, U. Essen, and R. Kneser. On Structuring Probabilistic Dependences in Stochastic Language Modelling. *Computer Speech and Language*, 8:1–38, 1994.

[83] L.T. Niles and H.F. Silverman. Combining Hidden Markov Model and Neural Network Classifiers. In *Proc ICASSP'90*, pages 417–420, 1990.

[84] Y. Normandin. *Hidden Markov Models, Maximum Mutual Information Estimation, and the Speech Recognition Problem*. PhD thesis, Department of Electrical Engineering, McGill University, Montreal, March 1991.

[85] Y. Normandin and S.D. Morgera. An Improved MMIE Training Algorithm for Speaker-Independent, Small Vocabulary, Continuous Speech Recognition. In *Proc. ICASSP'91*, volume 1, pages 537–540. IEEE, May 1991.

[86] J.J. Odell, V. Valtchev, P.C. Woodland, and S.J. Young. A One Pass Decoder Design For Large Vocabulary Recognition. In *Proc. ARPA Human Language Technology Workshop*. ARPA, March 1994.

[87] K.K. Paliwal. Dimensionality Reduction of the Enhanced Feature Set for the HMM-Based Speech Recognizer. *Digital Signal Processing*, 2:157–173, 1992.

[88] T. Parsons. *Voice and Speech Processing*. McGraw-Hill, Inc., 1987.

[89] D.B. Paul. The Lincoln Robust Continuous Speech Recogniser. In *Proc. ICASSP'89*, volume 1, pages 449–452. IEEE, 1989.

[90] W.H. Press, W.T. Vetterling, S.A. Teukolsky, and B.P. Flannery. *Numerical Recipies in C, The Art of Scientific Computing*. Cambridge University Press, 1992.

[91] L.R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. of the IEEE*, 77(2), February 1989.

[92] L.R. Rabiner, S.E. Levinson, and M.M. Sondhi. On the Application of Vector Quantization and Hidden Markov Models to Speaker-Independent Isolated Word Recognition. *The Bell Systems Technical Journal*, 62(4), April 1983.

[93] L.R. Rabiner, M.M. Sondhi, and S.E. Levinson. A Vector Quantizer Incorporating both LPC Shapes and Energy. In *Proc. ICASSP'84*, pages 556–569. IEEE, 1984.

[94] D. Rainton and S. Sagayama. Appropriate Error Criterion Selection for Continuous Speech HMM Minimum Error Training. In *Proc. ICSLP'92*, volume 1, pages 233–236, Banff, Canada, October 1992.

[95] S. Renals, N. Morgan, and H. Bourlard. Probability Estimation by Feed-forward Networks in Continuous Speech Recognition. In *IEEE Workshop on Neural Networks for Signal Processing*, October 1991.

[96] A.G. Richter. Modelling of Continuous Speech Observations. In *Conf. on Advances in Speech processing*, IBM Europe Institute, Oberlech, Austria, July 1986.

[97] A.J. Robinson. *Dynamic Error Propagation Networks*. PhD thesis, Cambridge University Engineering Department, Trumpington Street, Cambridge, CB2, June 1989.

[98] A.J. Robinson. Several Improvements to a Recurrent Error Propagation Network Phone Recognition System. Technical Report CUED/F-INFENG/TR.82, Engineering Department, Cambridge University, September 1991.

[99] A.J. Robinson and F. Fallside. Phoneme Recognition from the TIMIT Database using Recurrent Error Propagation Networks. Technical Report CUED/F-INFENG/TR.42, Engineering Department, Cambridge University, March 1990.

[100] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning Internal Representation by Error Propagation. In *Parallel Distributed Processing*, volume 1, pages 318–362. MIT Press, 1986.

[101] D.E. Rumelhart and J.L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. Bradford Books/MIT Press, Cambridge MA, 1986.

[102] W. Schiffmann, M. Joost, and R. Werner. Optimization of the Backpropagation Algorithm for Training Multilayer Perceptrons. Technical report, University of Koblenz, Institute of Physics, Rheinau 3-4, W-5400 Koblentz, Germany, 1992.

[103] R. Schwartz and S. Austin. A Comparison of Several Approximate Algorithms for Finding Multiple (N-Best) Sentence Hypotheses. In *Proc. ICASSP'91*, volume 1, pages 701–704. IEEE, May 1991.

[104] C.E. Shannon. A Mathematical Theory of Communication. *Bell Systems Technical Journal*, 27:379–423, 623–656, 1948.

[105] R.S. Sutton. Two Problems with Backpropagation and Other Steepest-Descent Learning Procedures for Networks. In *Proc. The Eighth Annual Conference of the Cognitive Science Society*, pages 823–831, 1986.

[106] V. Valtchev, J.J. Odell, P.C. Woodland, and S.J. Young. A Dynamic Network Decoder Design for Large Vocabulary Speech Recognition. In *Proc. ICSLP'94*. Acoustical Society of Japan, September 1994.

[107] A.J. Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–269, 1967.

[108] A. Waibel, T. Hanazawa, and K. Shikano. Phoneme Recognition: Neural Networks vs. Hidden Markov Models. In *Proc ICASSP'88*, pages 107–110. IEEE, 1988.

[109] P.C. Woodland and D.R. Cole. Optimising Hidden Markov Models using Discriminative Output Distributions. In *Proc. ICASSP'91*. IEEE, April 1991.

[110] P.C. Woodland, C.J. Leggetter, J.J. Odell, V. Valtchev, and S.J. Young. The 1994 HTK Large Vocabulary Speech Recognition System. In *Proc. ICASSP'95*, Detroit, 1995. IEEE. To appear.

[111] S.J. Young. Competitive Training: A Connectionist Approach to the Discriminative Training of Hidden Markov Models. Technical Report CUED/F-INFENG/TR.41, Cambridge University Engineering Department, March 1990.

[112] S.J. Young. The General Use of Tying in Phoneme-Based HMM Speech Recognisers. In *Proc. ICASSP'92*. IEEE, March 1992.

[113] S.J. Young. The HTK Hidden Markov Model Toolkit: Design and Philosophy. Technical Report CUED/F-INFENG/TR.152, Cambridge University Engineering Department, 1993.

[114] S.J. Young, J.J Odell, and P.C. Woodland. Tree-Based State Tying for High Accuracy Acoustic Modelling. In *Proc. ARPA Human Language Technology Workshop*. Morgan Kaufmann, March 1994.

[115] S.J. Young, N.H. Russel, and J.H.S. Thornton. Token Passing: a Simple Conceptual Model for Connected Speech Recognition Systems. Technical Report CUED/F-INFENG/TR.38, Cambridge University Engineering Department, July 1989.

[116] S.J. Young and P.C. Woodland. State Clustering in Hidden Markov Model-based Continuous Speech Recognition. *Computer Speech and Language*, 8:369–383, 1994.

[117] S.J. Young, P.C. Woodland, and W.J. Byrne. *HTK: Hidden Markov Model Toolkit V1.5 - Reference Manual*. Entropic Research Laboratories Inc., September 1993.