



UNIVERSITEIT • STELLENBOSCH • UNIVERSITY
jou kennisvenoot • your knowledge partner

Tree-based Gaussian Mixture Models for Speaker Verification

by

Francois Dirk Cilliers

*Thesis presented at the University of Stellenbosch in partial
fulfilment of the requirements for the degree of*

Master of Science in Electronic Engineering

Department of Electrical and Electronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Study leader: Prof J.A. du Preez

December 2005

Copyright © 2005 University of Stellenbosch
All rights reserved.



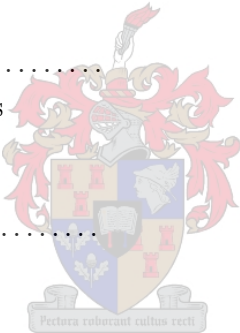
Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature:

F.D. Cilliers

Date:



Abstract

Tree-based Gaussian Mixture Models for Speaker Verification

F.D. Cilliers

Department of Electrical and Electronic Engineering

University of Stellenbosch

Private Bag X1, 7602 Matieland, South Africa

Thesis: M.Sc.Eng. (Electronic with Computer Science)

December 2005

The Gaussian mixture model (GMM) performs very effectively in applications such as speech and speaker recognition. However, evaluation speed is greatly reduced when the GMM has a large number of mixture components. Various techniques improve the evaluation speed by reducing the number of required Gaussian evaluations. These techniques are based on the observation that only a few mixture components contribute significantly to the final model likelihood.

One technique that shows promise beyond the basic speed improvement, is the use of tree-based GMMs. The mixture components are used as leaf nodes in a tree structure. Groups of leaf nodes are represented by approximating Gaussian density functions. This approximation is repeated for each layer of the tree, up-to the root node. The group of mixture components that contribute the most to the model likelihood can then be found quickly by searching through the approximating nodes that also have the highest contributions.

We introduce a new version of the model, called a tree-based adaptive GMM, and apply it to the task of speaker verification. This model not only searches for the components with the highest likelihood contributions, but it produces a virtual GMM using the approximating nodes for the rest of the model. No significant extra computation is required, because those approximating nodes are already evaluated during the search procedure. This new model can also be used to improve the speed of model training. We show that performance similar to the regular GMM can be obtained with an execution time that is logarithmically related to that of the regular GMM.

Opsomming

Boom-gebaseerde Gaussiese Mengselmodelle vir Sprekerverifikasie

F.D. Cilliers

Departement Elektriese en Elektroniese Ingenieurswese

Universiteit van Stellenbosch

Privaatsak X1, 7602 Matieland, Suid Afrika

Tesis: M.Sc.Ing. (Elektronies met Rekenaarwetenskap)

Desember 2005

Die Gaussiese mengselmodel (GMM) lewer hoë werkverrigting vir toepassings soos spraak- en sprekerherkenning. Maar, berekeningspoed word drasties verminder vir 'n GMM met baie mengselkomponente. Verskeie tegnieke verbeter die berekeningspoed deur die hoeveelheid benodigde Gaussiese berekeninge te verminder. Die tegnieke berus op die waarneming dat slegs 'n klein aantal mengselkomponente beduidend bydra tot die finale modelwaarskynlikheid.

'n Tegniek wat belowend lyk vir meer as blote spoedverbetering, is die gebruik van boom-gebaseerde GMM'e. Die mengselkomponente word as blaarnodusse in 'n boom gebruik. Groepe blaarnodusse word verteenwoordig deur benaderende Gaussiese digtheidsfunksies. Die benadering word herhaal vir elke vlak van die boom, tot-en-met die wortelnodus. Die groep mengselkomponente wat die meeste bydra tot die modelwaarskynlikheid, word vinnig gevind met 'n soektog deur daardie benaderende nodusse wat ook die grootste bydraes toon.

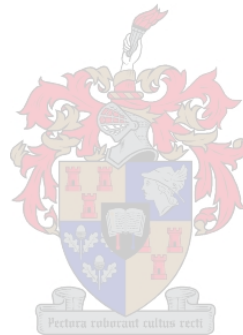
Ons stel 'n nuwe model bekend, die boom-gebaseerde aanpasbare GMM, en gebruik dit vir sprekerverifikasie. Die model soek nie net vir die komponente met die hoogste bydraes nie, maar skep 'n virtuele GMM deur gebruik te maak van die benaderende nodusse. Geen beduidende bykomende berekeninge word benodig nie, omdat die benaderende nodusse alreeds bereken word tydens die soektog. Hierdie nuwe model kan ook afrigspoed verbeter. Ons wys dat werkverrigting soortgelyk aan die van 'n gewone GMM behaal word, maar met 'n uitvoertyd wat logaritmes verwant is aan die van 'n gewone GMM.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations:

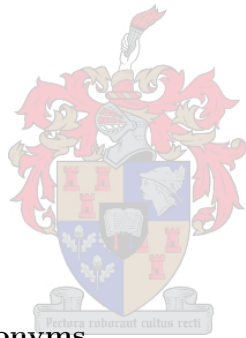
- God, for creating this wonderful world with all sorts of fascinating things with which we mere humans can keep our thoughts occupied;
- Prof J.A. du Preez, for being an excellent and pleasant study leader, an unbounded source of knowledge, and being responsible for many of the ideas that were included in this thesis;
- my parents, for their financial and general support, without which the task of producing this thesis would have been much more difficult;
- the National Research Fund (NRF), and the University of Stellenbosch for their much needed financial aid;
- Ludwig Schwardt, for all his effort at making very useful notes on pattern recognition, and for a few Matlab scripts that were used in this thesis;
- all the people who made contributions to the PatrecII software suite, which saved me a lot of effort;
- all the developers and maintainers that put their effort into the Debian Linux operating system and related software packages;
- all the people that put their effort into the development of the LyX software package as well as the L^AT_EX packages and software that were used for the generation of this thesis;
- Danie Els, for his USthesis L^AT_EX style sheet, Pieter Rautenbach for his contributions to the packaging and Stéfan van der Walt for creating the corresponding LyX layouts;
- Niko Brümmer from DataVoice, for allowing the use of his results in the 2004 NIST SRE as reference;

- Prof C.H. Lee, for kindly answering my questions regarding the SMAP formulae;
- Herman Engelbrecht, for providing insightful comments and being generally interested in my work;
- Jaco Müller, for helping out with many of the administrative tasks that were required to complete this thesis;
- (in alphabetical order) Andries du Toit, Ferdl Graser, Elana Kok, De Villiers Malan, Francois Malan, André Muller, Jaco Müller, Pieter Rautenbach, Sonja Slabber, Carike Visagie, Stéfan van der Walt and Neels van Greunen for being great friends, throwing outstanding parties, having long conversations, showing interest in my work, and all their attempts at motivating me.



Contents

Declaration	ii
Abstract	iii
Opsomming	iv
Acknowledgements	v
Contents	vii
List of Figures	xii
List of Tables	xvii
List of Algorithms	xviii
List of Abbreviations/Acronyms	xix
List of Symbols	xxi
1 Introduction	1
1.1 Motivation	1
1.2 Concepts Relating to Modelling in Speaker Verification	2
1.3 Previous Work Regarding Tree-based GMMs	3
1.4 Research Objectives	5
1.5 Contributions	5
1.6 Overview of this work	6
2 Literature Study	8
2.1 Introduction	8
2.2 Speaker Verification	9
2.2.1 Front-end Processing	9
2.2.2 Evaluation	9

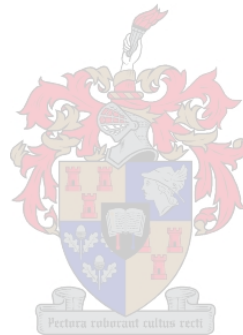


2.3	Speaker Modelling	11
2.4	Tree-based Approaches	12
2.5	Performance Differences	14
2.5.1	GMM versus VQ	14
2.5.2	GMM versus HMM	15
2.5.3	HGMM versus GMM	16
2.5.4	SGMM/SBM versus GMM	16
2.6	Summary	17
3	Modelling Acoustic Properties of Speaker Identity	20
3.1	Introduction	20
3.2	Speaker Verification Overview	20
3.3	Speaker Modelling Overview	22
3.4	Theory of Gaussian Mixture Modelling	23
3.5	Model Training	25
3.5.1	Maximum Likelihood Estimation	25
3.5.2	Expectation-Maximisation	27
3.5.3	Model Initialisation	30
3.5.4	Maximum a Posteriori (MAP) Adaptation	30
3.6	Summary	33
4	Tree-based Adaptive Gaussian Mixture Models	34
4.1	Introduction	34
4.1.1	The Speed Problem	34
4.1.2	Popular Approach to Speed Improvement	35
4.1.3	Tree-based Approach to Speed Improvement	35
4.1.4	The Tree-Based Adaptive GMM	37
4.2	Theory	37
4.3	Training	42
4.4	Summary	44
5	The Speaker Verification System	46
5.1	Introduction	46
5.2	Feature Extraction	46
5.2.1	Front-end Signal Processing	46
5.2.2	Mel-Frequency Cepstral Coefficients	49
5.2.3	Channel Compensation	49
5.2.4	Dynamic Features	51
5.2.5	Feature Normalisation	51

5.3	Evaluation	52
5.3.1	Bayesian Decision Theory	52
5.3.2	Score Normalisation (T-Norm)	55
5.3.3	Threshold Selection	56
5.4	Distributed Processing for the Verification Task	57
5.5	Summary	58
6	Implementation Issues	59
6.1	Introduction	59
6.2	Methodology	60
6.3	Working with data	61
6.3.1	Applying data to the system	61
6.3.2	Speech databases	63
6.3.3	Selection of training and evaluation data	64
6.4	T-BAGMM algorithms	67
6.4.1	Evaluation	67
6.4.2	Training	69
6.5	The baseline system configuration	71
6.5.1	Introduction	71
6.5.2	Front-end processing	71
6.5.3	UBM	72
6.5.4	Speaker models	72
6.5.5	Evaluation	73
6.6	Process distribution	73
6.6.1	First generation	73
6.6.2	Second generation	73
6.7	Summary	74
7	Experimental Investigation	76
7.1	Introduction	76
7.2	The Detection Error Trade-off (DET) Curve	76
7.3	Significance Testing (McNemar's Test)	77
7.4	General System Evaluation: 2004 NIST SRE	80
7.4.1	Overview	80
7.4.2	Brief Description of the Reference System	81
7.4.3	Procedure for Full Performance Evaluation	83
7.5	The Baseline System Performance	83
7.5.1	Motivation	83
7.5.2	Setup and Execution	83

7.5.3	Results	84
7.5.4	Interpretation	86
7.6	Computational Characteristics of the T-BAGMM	86
7.6.1	Motivation	86
7.6.2	Setup and Execution	87
7.6.3	Results	89
7.6.4	Interpretation	94
7.7	Influence of Mixture Component Count	97
7.7.1	Motivation	97
7.7.2	Setup and Execution	97
7.7.3	Results	97
7.7.4	Observations	101
7.8	Influence of Front-end Processing	101
7.8.1	Motivation	101
7.8.2	Setup and Execution	102
7.8.3	Results	103
7.8.4	Interpretation	106
7.9	Influence of the T-Norm Impostor Set Size	107
7.9.1	Motivation	107
7.9.2	Setup and Execution	107
7.9.3	Results	108
7.9.4	Interpretation	110
7.10	Influence of the UBM Data Set Size	111
7.10.1	Motivation	111
7.10.2	Setup and Execution	111
7.10.3	Results	112
7.10.4	Interpretation	114
7.11	MAP Adaptation for both Means and Covariance Matrices	114
7.11.1	Motivation	114
7.11.2	Setup and Execution	115
7.11.3	Results	115
7.11.4	Interpretation	115
7.12	Summary	116
8	Conclusions and Recommendations	118
8.1	Tree-based Adaptive Gaussian Mixture Model	118
8.2	The Acoustic Speaker Verification System	120
	Bibliography	122

A	Derivations and Justifications	A-1
A.1	The Likelihood Ratio and Test Normalisation	A-1
A.2	Intuitive Derivation of MAP Formulas	A-3
A.3	On-line Re-estimation Formulas	A-6



List of Figures

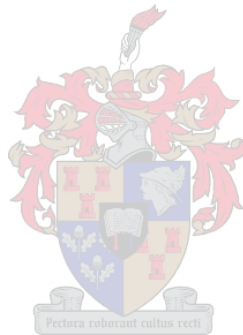
3.1	Simplified block diagram of the speaker verification system.	22
3.2	Block diagram representing a GMM. Each of the Gaussian PDF components p_k are evaluated for the test feature vector \mathbf{x} . Their resultant likelihoods are then weighed and summed to produce a model likelihood $p(\mathbf{x} \lambda)$	24
4.1	Tree-based GMM representation. Each node has a node weight g_i and a Gaussian PDF $p_i = p(\mathbf{x} \lambda_i)$	37
4.2	Example of a virtual GMM. (a) shows the original tree. Nodes with solid outlines were selected for the virtual GMM. (b) shows the virtual GMM that was constructed from the tree in (a). Note that the resulting mixture weights do add up to one.	39
4.3	Very simple example of T-BAGMM evaluation. (a) and (b) show the regular GMM that will be approximated by a T-BAGMM. The grey dots in the scatter plot (a) are two-dimensional feature points. The circles in (a) indicate the standard deviation curves of each component PDF. (c) and (d) represent the comparison between the two nodes in the first tree level. Here node 2 has a higher likelihood contribution. (e) and (f) show the final selection of nodes for the resulting 3-component virtual GMM. The grey curves in (c) and (e) are shown for reference.	41
5.1	Block diagram of feature extraction.	47
7.1	DET curve showing the difference between the performance of the baseline and the reference systems.	84
7.2	DET curve showing the difference between the performance of the baseline GMM and the baseline T-BAGMM system.	85

7.3	DET curve showing the statistical significance of differences between the performance of the baseline GMM and the baseline T-BAGMM system. A significance level of 5 % ($\alpha = 0.05$) was used for this figure. Areas with thick lines show significant difference and areas with thin lines indicate where differences are doubtful.	85
7.4	Training times for the T-BAGMM UBM relative to that of GMM UBM. Lower values are better. Where the ratio is unity, the training times for both models are the same.	89
7.5	Relative difference between training data log-likelihoods of T-BAGMM UBM and that of GMM UBM. Zero values indicate identical performance. Positive values represent a better fit of the training data by the T-BAGMM.	89
7.6	Evaluation time for the T-BAGMM relative to that of the corresponding GMM, shown with respect to the mixture component count. Curves for different test beam widths are shown. The theoretical lower boundary is also shown.	90
7.7	Evaluation time for the GMM relative to that of the corresponding T-BAGMM (base-2 logarithmic scale), shown with respect to the mixture component count. Curves for different test beam widths are shown. This figure shows more clearly how the measured speed increase exceeds the theoretical upper boundary.	91
7.8	EER for the T-BAGMM relative to that of the corresponding GMM, shown with respect to the mixture component count. Lower values are better. This figure shows that the accuracy of a T-BAGMM system differs very little from that of a regular GMM system.	92
7.9	FRR at 1% FAR for the T-BAGMM relative to that of the corresponding GMM, shown with respect to the mixture component count. Lower values are better. This figure shows that the accuracy of a T-BAGMM system using a low beam width is consistently better in this region than that of a regular GMM system.	92
7.10	Absolute evaluation time given as the number of processor clock cycles (CPU ticks). This figure shows how the evaluation times differ in reality between systems using different component counts. These differences are not visible on the figures that show relative times.	93

7.11	Absolute evaluation time given as the number of processor clock cycles (CPU ticks). This is a close-up of the previous figure. It can be consulted to avoid a speed penalty when more mixture components are needed (i.e. higher accuracy) for a particular application.	93
7.12	Evaluation time of the T-BAGMM relative to that of the corresponding GMM, shown with respect to the testing node score beam width. Curves for different mixture component counts are shown. This figure gives an alternative perspective to the speed-accuracy trade-off.	94
7.13	DET curve showing the influence of the number of GMM components.	97
7.14	DET curve close-up showing the influence of the number of GMM components in the low FAR region.	98
7.15	DET curve close-up showing the influence of the number of GMM mixture components at the EER.	98
7.16	DET curve showing the influence of the number of T-BAGMM leaf nodes.	99
7.17	DET curve close-up showing the influence of the number of T-BAGMM leaf nodes in the low FAR region.	99
7.18	DET curve close-up showing the influence of the number of T-BAGMM leaf nodes at the EER.	100
7.19	Comparison of the EER between the GMM and T-BAGMM, shown with respect to the number of mixture components (leaf nodes).	100
7.20	Comparison of the FRR between the GMM and T-BAGMM where the FAR=1%, shown with respect to the number of mixture components (leaf nodes).	101
7.21	DET curve showing the influence of CMS.	103
7.22	DET curve showing the statistical significance of differences between the performance of the baseline T-BAGMM system with and without CMS applied. A significance level of 5 % ($\alpha = 0.05$) was used for this figure. Areas with thick lines show significant difference and areas with thin lines indicate were differences are doubtful.	103
7.23	DET curve showing the influence of Δ -features without CMS applied. Curves are shown for calculation of Δ -features with a 2-point window.	104

7.24	DET curve showing the influence of Δ -features, with CMS applied. Curves are shown for calculation of Δ -features with a 2-point window as well as with a 5-point window.	104
7.25	DET curve showing the influence of $\Delta\Delta$ -features. Only 5-point windows were used here.	105
7.26	DET curve showing the influence of alternative cepstral feature selection (13 MFCCs minus the zeroth coefficient).	105
7.27	DET curve showing the statistical significance of differences between the performance of the T-BAGMM system using basic or alternative cepstral feature selection. A significance level of 5 % ($\alpha = 0.05$) was used for this figure. Areas with thick lines show significant difference and areas with thin lines indicate were differences are doubtful.	106
7.28	DET curve showing the influence of the number of T-Norm impostors using CMS and Δ -features.	108
7.29	DET curve showing the influence of the number of T-Norm impostors using CMS and both Δ - and $\Delta\Delta$ -features.	109
7.30	DET curve close-up showing the influence of the number of T-Norm impostors at low FAR.	109
7.31	DET curve close-up showing the influence of the number of T-Norm impostors at the EER.	110
7.32	DET curve showing the difference between a system using 200 UBM speakers and one using 400 UBM speakers. Here, 2048 mixture components are used with 20 T-Norm impostors.	112
7.33	DET curve showing the difference between a system using 200 UBM speakers and one using 400 UBM speakers. Here, 512 mixture components are used with 20 T-Norm impostors.	112
7.34	DET curve showing the statistical significance of differences between the performance of the T-BAGMM system using 200 or 400 UBM speakers with 512 mixture components. A significance level of 5 % ($\alpha = 0.05$) was used for this figure. Areas with thick lines show significant difference and areas with thin lines indicate were differences are doubtful.	113
7.35	DET curve showing the difference between a system using 200 UBM speakers and one using 400 UBM speakers. Here, 512 mixture components are used with 100 T-Norm impostors.	113

7.36 DET curve showing the difference between a system using mean-only MAP adaptation and one using MAP adaptation with regard to both the mean and the covariance matrix. The second system was tested with a relevance factor of 16 and again with a relevance factor of 72, as shown in parentheses. 115



List of Tables

2.1	Speaker Identification Accuracies for VQ and GMM	15
2.2	False acceptance rates for HMM and GMM where the false rejection rate is 0%.	15
2.3	Speaker verification EER for GMM and HGMM	16
2.4	Speaker recognition results for GMM and HGMM using NIST 2002 Multi-modal data	16
2.5	EER and computational reduction factor (F) for SGMM	17
2.6	EER for GMM and SGMM augmented with MLP	17
4.1	Theoretical trade-offs for different balanced tree structures. These figures were chosen so that there would be enough leaf nodes to accommodate 2048 mixture components.	36
4.2	Comparative summary between different model evaluation techniques	45
6.1	Technical specifications of the Switchboard-2 Phase III speech corpus	63
6.2	Technical specifications of the Mixer speech corpus	64
7.1	Comparison of error rates between the baseline GMM, the baseline T-BAGMM and the reference system	84
7.2	EER measurements (values in %)	91
7.3	FRR measurements where FAR=1% (values in %)	91
7.4	Speed improvement factors for the case where a GMM has 2048 mixture components and the corresponding T-BAGMM has a depth of either 9, 10 or 11 levels.	95

List of Algorithms

1	Binary-split model initialisation	31
2	Recursive evaluation of a T-BAGMM	40
3	Training of a T-BAGMM	43
4	Recursive Evaluation of a T-BAGMM, making use of a node score threshold	70



List of Abbreviations/Acronyms

ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
BPF	Band-pass Filter
CMS	Cepstral Mean Subtraction
CPU	Central Processing Unit
DCF	Detection Cost Function
DCT	Discrete Cosine Transform
DET	Detection Error Tradeoff
DFT	Discrete Fourier Transform
DSP	Digital Signal Processing
EER	Equal Error Rate
GMM	Gaussian Mixture Model
EM	Expectation Maximisation
FA	False Acceptance / False Alarm
FAR	False Acceptance Rate
FR	False Rejection
FRR	False Rejection Rate
HGMM	Hierarchical GMM

HMM	Hidden Markov Model
HPF	High-pass Filter
IID	Independent and Identically Distributed
LP	Linear Prediction
LPCC	Linear Prediction Cepstral Coefficient
LPF	Low-pass Filter
MAP	Maximum <i>a Posteriori</i>
MFCC	Mel-Frequency Cepstral Coefficient
ML	Maximum Likelihood
MLP	Multi-level Perceptron
NIST	National Institute for Standards and Technology
PDF	Probability Density Function
PR	Pattern Recognition
SBM	Structural Background Model
SGMM	Structural GMM
SHFS	Shell File System
SID	Speaker Identification
SMAP	Structural MAP
SRE	Speaker Recognition Evaluation
SSH	Secure Shell
T-BAGMM	Tree-based Adaptive Gaussian Mixture Model
T-Norm	Test Normalisation
UBM	Universal Background Model
VQ	Vector Quantisation

List of Symbols

Constants:

\mathbf{I}	Identity matrix
$\pi =$	3.141 592 653 589 793 238 462 643 383 279 5
$e =$	2.718 281 828 459 045 235 360 287 471 352 6

Variables, Vectors and Matrices:

A	Scalar variable representing a constant value
X	Random variable (roman type)
x	Scalar variable
\mathbf{x}	Column vector variable (bold-face)
\mathbf{x}_t	Observation vector at discrete time t
\mathcal{X}	Sequence of random variables $\{X_t : t = 1, 2, \dots, T\}$
\mathbf{X}	Matrix or sequence of observation vectors $\{\mathbf{x}_t : t = 1, 2, \dots, T\}$
$\boldsymbol{\mu}$	Statistical mean vector
μ	Statistical scalar mean
$\boldsymbol{\Sigma}$	Statistical covariance matrix
σ, σ^2	Statistical standard deviation and variance
w_i	Mixture weight
g_i	Node weight
g'_i	Node level weight
λ	Model parameters
Λ	Log-likelihood ratio (score)
θ	Decision threshold
s_n	Node score
δ	Node score beam width
ϕ	Node score threshold

Functions:

$\exp(x)$	The exponential function e^x
$P(x)$	The probability of the random variable X having the value x (i.e. $P(X = x)$)
$p(x)$	The probability density function (PDF) with respect to the variable x
$P(x A)$	The conditional probability of the variable x given the knowledge A
$p(x A)$	The conditional PDF with respect to the variable x given the knowledge A
$\mathcal{N}(\mathbf{x} \boldsymbol{\mu}, \Sigma)$	Multivariate Normal (Gaussian) PDF with mean $\boldsymbol{\mu}$ and covariance matrix Σ
$L_T(k)$	The tree level of node p_k



Chapter 1

Introduction

1.1 Motivation

In the modern age of digital computers, a vast amount of useful functions have already been developed to process signals in digital form. Speaker recognition is a technique of digital signal processing (DSP) regarding the correct identification of people by analysing their speech. It has a wide variety of useful applications in the broad areas of access control, transaction authentication, law enforcement and more [1]. Some of these applications, in combination with other technologies, might someday help to narrow the digital divide by providing computer-illiterate people easy access to technology.

In the past few decades, important contributions have been made in the field of speaker recognition. Among these contributions, the Gaussian mixture model (GMM) has proved extremely efficient for characterising speaker identity at the acoustic level [2]. One big disadvantage of using GMMs, is that the computation time of model evaluations can become very expensive. This is especially true in research and development where large numbers of trials must be executed to determine the performance of a verification system.

This thesis examines current, advanced techniques surrounding Gaussian mixture modelling in the focus area of acoustic text-independent speaker verification. These techniques transform the GMM into a tree structure that can improve the evaluation speed considerably. By building on these techniques, a new tree-based version of the GMM is introduced. Theoretical prediction indicate that this new model can achieve nearly a hundred-fold improvement in evaluation speed. This comes at a cost of requiring only double the amount of memory. The new model is not restricted to application in speaker verification, but can also replace the regular GMM in other tasks such as speech recognition.

1.2 Concepts Relating to Modelling in Speaker Verification

Speaker recognition falls under the general concept of pattern recognition (PR) or pattern classification [3, 4]. It is sub-divided into two main areas: Speaker identification (SID) and speaker verification [1]. The latter is also called speaker detection. For the SID task, it must be determined which person in a known, fixed-size set is doing the talking. For the speaker verification task, on the other hand, it must be determined whether a person is whom he or she is claimed to be. This requires discrimination between the target speaker and a possibly infinite number of impostor speakers.

For any PR task, features are needed that allow discrimination between the different classes in the data. For example: the number of wrinkles on a person's face might be a good feature for age discrimination. Normally, discrimination can be enhanced by taking a number of different features into account. This gives extra dimension (more viewpoints) to the data being studied. In mathematical terms, it thus makes sense to work with features as multi-dimensional vectors. Each dimension represents a different feature measurement that is extracted from the data.

Speech data come in the form of time-varying signals. A speech data set is a collection of sample speech signals. A number of schemes have already been produced for extracting useful information (to be used as features) from speech signals. In recent years, cepstral coefficients (of which a few different types exist) have become the features of choice for speech.

Models in PR are mathematical descriptions of classes, of which the parameters are estimated from feature vectors. On the acoustic level, a speaker model tries to represent all the possible sounds that the voice of the corresponding person can make. It is assumed that a person's voice is unique and therefore make unique sounds. But, in reality, a given sound might be produced by more than one person. This obviously makes the modelling process very difficult, because speaker models might overlap to quite an extent. Also, training data for any given person are always limited and do not necessarily contain all the possible sounds that the person can make. Statistical methods that use probability density functions (PDFs) to model classes are commonly used for PR tasks. This thesis employs a tree-based version of the Gaussian mixture model (GMM), which is a statistical model.

The verification task produces a score that indicates how well a given test sample (or utterance) matches the model of the claimed identity. A threshold

must be chosen to decide, based on the score, whether or not the test sample was actually produced by the claimed identity. If the score is above the threshold, the decision is affirmative, otherwise not.

Two types of errors occur during verification. A false acceptance (FA) (or false alarm) is the error made when the verification system accepts the claim, although the claimed identity is not present in the sample. A false rejection (FR) (or miss) is the error made when the system fails to detect the claimed identity, although it is present in the sample. The performance of a system is normally characterised by the equal-error rate (EER). This is the probability of an error occurring if the threshold is chosen so that the occurrence of both FA and FR errors are equally likely.

Text-dependent speaker recognition requires the speaker to speak a cue sentence from a fixed set. The number of ways in which these sentences can be produced by a single person are somewhat limited. This allows the system to know most of the sounds that can be expected from a test sample. Much higher accuracies can therefore be obtained than with text-independent speaker recognition where the speaker may say anything. With this latter method, the system must be able to deal with sounds that it may not have encountered during training.

To effectively construct any practical speaker recognition system, a large volume of training and testing data is required. Speaker models are created from training data and the system performance (accuracy) is measured from testing data. The latter are also used to choose appropriate thresholds. Many commercial speech databases are available to satisfy this need. A speech database is often also called a speech corpus.

1.3 Previous Work Regarding Tree-based GMMs

As used in modern speaker recognition systems, Gaussian mixture modelling has become a very popular technique since it was first introduced [2]. Apart from the fact that it gives very good performance, it has an intuitive connection to the physical properties of a person's voice. In short, a GMM is a sum of weighted component Gaussian PDFs. Each component can be thought of as representing a given group of sounds that a person can produce. The GMM as a whole can therefore represent all the sounds of a person's voice with relatively high accuracy.

Because of data scarcity, early systems used a maximum of 64 components for speaker models. Later, the technique of Bayesian speaker adaptation (originally used in speech recognition systems [5]) was applied to create speaker

models from a universal background model (UBM) [6]. The UBM is trained using data from a large population of speakers, thereby creating a model representing the general (or background) speaker. Because so much training data are available for the UBM, the number of Gaussian mixture components was increased to 2048. This allowed much more detailed modelling of the acoustic space. The drawback of having so many components, however, is that computation time is also increased considerably.

Recently, researchers have started to develop techniques to improve the speed of GMM computation by rearranging the components into a tree formation [7, 8, 9, 10]. The tree is usually constructed so that the nodes in each level serve as approximations for their respective descendant nodes. When determining the score of a test feature vector for the model, the tree structure ensures that a quick search can be made to find only those components that contribute significantly to the score. Chapter 2 describes these developments in more detail.

In existing techniques, the contributions of approximate nodes to the model score are used to determine the search path. Only the subtrees of those nodes with high enough contributions are evaluated further. Normally, the contributions of these approximate nodes are otherwise ignored, because they are not actual mixture components and merely exist to facilitate the search.

As a consequence of the research for this thesis, a new version of the tree-based GMM was produced, called the tree-based adaptive Gaussian mixture model (T-BAG mixture model, or T-BAGMM). It has the property of including the contributions of approximate nodes that have lower contributions to the score, but only when they are evaluated as part of the search procedure. These nodes then serve as a cost-efficient approximation to the clusters of mixture components that would otherwise have been discarded. This action costs nothing extra in computation time, but it is hoped to produce slightly better performance than other speed improvement techniques. It effectively provides a multi-resolution GMM that models different regions of feature space with different degrees of detail, depending on the location of the test feature vector. This gives the model its adaptive nature.

1.4 Research Objectives

The main concern of this thesis is the improvement of evaluation speed for the GMM. Although other techniques exist to perform this function, the solution presented here is believed to have greater potential. The primary aim of this thesis therefore consists of:

- introducing a new time-efficient version of the GMM, based on recent advances made by other researchers;
- providing detailed discussions about factors surrounding the construction and functionality of the model; and
- investigating the new model's effectiveness in the application of speaker verification.

The secondary aim is to fully describe the construction of a modern acoustic-level text-independent speaker verification system. To achieve this second objective, it is necessary to cover all the components that are used to construct the verification system. These are:

- front-end processing (signal analysis and feature extraction),
- speaker modelling,
- evaluation and decision making.

In a thesis such as this one, it is not practical to investigate all these system components in detail. However, detailed descriptions of specific techniques can be provided. Also, investigations should be performed to discover the influence of some of these techniques on the system performance. As a consequence of the primary objective, the speaker modelling component receives the greatest attention.

1.5 Contributions

Throughout the development process and experimental investigations, a few important contributions were made by this research:

- A detailed description of the new T-BAGMM was produced. Some additional insight regarding its implementation was given, and the possibility of adjusting the speed-accuracy trade-off was discovered and applied.

- It was discovered that the accuracy of the T-BAGMM is very similar to the GMM. The T-BAGMM also performs consistently better in the low false alarm region, although no proper explanation for this has yet been found. The T-BAGMM was used to perform multiple experiments successfully while experiencing up-to roughly a 100-fold improvement in evaluation speed. It was observed that the speed measurement for 2048-component models was even better than predicted by theory. The reason for this has also not been found, but inaccuracies in the measurements and prediction may have a small contribution. Training speed was also improved considerably.
- Characteristic curves were obtained of the T-BAGMM speed improvement. These can be used by developers as an indication of whether, and by how much, the T-BAGMM will benefit any particular application.
- Various techniques were investigated under the same challenging conditions of the 2004 National Institute of Standards and Technology (NIST) speaker recognition evaluation (SRE). These results present the unique perspective that shows how the different techniques improve performance relative to each other. In this way, it was seen that channel compensation plays a dominant role in high-performance systems.
- As an alternative to existing, but more complex software packages, a simple implementation of process distribution was developed using Linux shell scripts.
- Experimental work was performed using the PatrecII software suite that is being developed by the DSP group of the University of Stellenbosch. During this research work, software components were created for PatrecII to implement T-BAGMM functionality. Some other PatrecII components were also improved upon, in order to complete the verification system. Issues regarding the implementation of algorithms are discussed in this thesis, but no specifics about the actual software implementation are given.

1.6 Overview of this work

This section describes the organisation of this thesis. Chapter 2 reviews work done previously by other researchers on the topics of speech processing and specifically speaker verification in the acoustic domain. The various techniques used in this thesis are introduced. This covers feature extraction, speaker mod-

elling and evaluation methods. Special attention is given to modelling techniques regarding tree-based GMMs.

Chapter 3 introduces the concept of speaker modelling for the speaker verification task. It provides the theory of modelling acoustic space with GMMs. The structure of the GMM is described, as well as the practical methods for determining its parameters through maximum likelihood (ML) estimation and the expectation maximisation (EM) algorithm. The method of adapting speaker-specific models from a more general speaker-independent model, using maximum *a posteriori* (MAP) adaptation, is also covered.

Chapter 4 introduces the new T-BAGMM that was developed for this thesis. Thorough theoretical coverage of the model structure and method of evaluation is given. Model training is also discussed.

Chapter 5 covers the remaining parts of the speaker verification system. It shows how the different components work together to produce a verification result. Detailed descriptions of specific techniques for front-end signal processing, the feature extraction process and score calculation are given. The theoretical foundation for making decisions is discussed. The general idea for applying distributed processing to the verification task is also covered in this chapter.

Chapter 6 describes the various choices and algorithmic specialisations that were made for this thesis. It investigates the choices for various parameters of the underlying processes in the system. A discussion is given about the application of data, and the characteristics that are desired in the data are pointed out. The actual method of implementation for distributed processing is described here. Specific additions or alterations to the general algorithms (which are described in the theoretical sections of this thesis) are also discussed.

Chapter 7 shows the results of a few investigations into the effectiveness of the new T-BAGMM. Figure 7.2 and Table 7.1 show that the T-BAGMM performance matches closely that of the regular GMM. Specific speed improvement measurements are summarised for model training in Figure 7.4 and for evaluation in Figure 7.6. The influence of a few other techniques that were employed are also shown.

Finally, Chapter 8 makes a few conclusions from this research. Some recommendations are also given for possible further investigations.

Chapter 2

Literature Study

2.1 Introduction

The general verification (or detection) task requires that a system confirm the presence of a pattern in a sample of data. In speaker verification, it must be determined whether the speech of a given person is present in a speech sample. This task is therefore also referred to as speaker detection. Verification itself is a form of Pattern Recognition of which general concepts are discussed by appropriate literature [3, 4].

By 1975, digital computers have been in use for about 30 years or more. The basic structure for speaker verification systems in use at that time is outlined as follows [11]: From a test utterance of a specific cue sentence or phrase, features such as pitch period and intensity were calculated on a per-frame basis. These features produced parameter contours that represent the utterance. The contours were then time-aligned to the reference contours of the claimed identity and the distance between them compared to a threshold value, either accepting or rejecting the claim.

Many advances have been made in the following years. The modern structure of speaker verification systems [1] is not very different from the above description. New kinds of features are extracted from the test utterance on a per-frame basis during front-end processing. These features are then compared to the model of the target identity as well as to the models of possible impostors. The likelihood ratio is the ratio between how well the target model describes the features and how well the impostor models describe the features. This ratio is compared to a threshold value that decides whether to accept or reject the claim. This system structure is also adopted for this thesis. It must be noted that this system is based on statistical methods and therefore it is assumed that the reader is familiar with probability theory principles [12, 13].

2.2 Speaker Verification

2.2.1 Front-end Processing

Features in speech signals that can distinguish between speakers are not obvious. Currently, we do not yet have an exact understanding of speech production and perception, and especially how speaker identity is conveyed. However, it is known that the vocal tract is the main (but not the only) physiological factor that distinguishes different voices from each other. The speech spectrum [14, 15] (obtained from frequency analysis [16, 15]) contains information about a person's vocal tract structure. Consequently, the spectrum has been used very effectively for speaker recognition in the acoustic domain [17].

In the search for understanding the acoustic properties in speech, feature extraction processes have spawned from research that try to model speech with as few as possible parameters (i.e. compress it) with the intention of reconstructing it as well as possible. One of the early and still widely used methods of speech modelling is Linear Prediction (LP). This was eventually applied to speaker verification systems by using LP coefficients as features [11]. More recently, LP cepstral coefficients (LPCCs) (based on analysis of the LP speech spectrum) have been employed in speech processing applications [18, 19, 8]. A disadvantage of model-based approaches, is that they can be badly affected by noise [20].

The most popular features used in speaker verification systems today are Mel-frequency cepstral coefficients (MFCCs) [2, 6, 9, 10, 18, 21] that are obtained by cepstral analysis [22], using directly computed filter-bank energies rather than parameters of a speech production model. MFCCs are employed in this thesis and will be described in Section 5.2.

Also of great importance are the techniques that have been developed to make the system robust against noise and channel variations. Cepstral mean subtraction (CMS) [23] is a popular technique that aims to remove linear channel distortions. It was recently found that prior knowledge about language can improve the performance of CMS [24]. Although it is not covered by this thesis, short-time Gaussianisation [25] (or feature warping) has been shown to provide better channel compensation than CMS.

2.2.2 Evaluation

In general, speaker verification systems make decisions with a likelihood ratio test, which is based on Bayes decision theory [3]. A score representing how well the target speaker model matches the test utterance is compared to a score

representing how well a model for impostors matches the test utterance. When the ratio of the two scores is above a certain threshold value, the system accepts the claim that the test utterance was produced by the target speaker. The log-likelihood ratio is also referred to as the normalised log-likelihood score [26], because the target score is in effect normalised by the impostor score.

Two major approaches exist for modelling impostors [1, 27, 28]. In the one case, a world model, or universal background model (UBM) is used to represent impostors with a general (speaker independent) voice model. This has the advantage that only a single impostor model has to be trained and evaluated. The same UBM can also be used for speaker adaptation when training specific target speaker models. The other case uses a finite set of non-target models, called cohort models, to compute an impostor score. The simplest method for using a set of cohort models is to sum their likelihoods for the test utterance and divide by the number of models in the cohort set. This gives a single impostor score that can be used in the likelihood ratio. The models in the cohort set can be selected in a number of ways [27].

The output scores of the system have a certain statistical distribution that is dependent on both the target speaker model and the test observation utterance. To obtain a single global threshold that can be used for all target speakers, the score distribution must be scaled (or normalised). Related to the cohort approach for impostor modelling, the Test normalisation (T-Norm) scaling technique [28, 29] is used in this thesis. T-Norm scales target scores in such a way that the distribution of all impostor trial scores have zero mean and unity variance for a given test utterance. Because all impostor trial scores are not available when scaling must be performed, a subset of impostor model scores is used to estimate the scaling parameters. T-Norm is able to calculate the scaling parameters for each test utterance and therefore compensate for differences in the acoustic environment.

For research purposes, it is necessary to compare the performance (effectiveness) of different system configurations. In recent years, the Detection Error Tradeoff (DET) curve [30] has become the dominant tool for making such comparisons. For a given speaker verification system, performance results for system evaluation may be different for one set of data than for another set. It is important to take this variability and uncertainty into account when comparing results of different system configurations. McNemar's test [31] provides a good framework for determining whether the difference in the performance of two systems is statistically significant.

2.3 Speaker Modelling

Over the years, various different methods have been applied to model speaker identity for speaker recognition tasks. Some of the more well-known methods include vector quantisation (VQ) [32], hidden Markov models (HMMs) [33, 34, 35, 36] and Gaussian mixture models (GMMs) [2].

VQ was originally used to encode (or compress) speech. Any given utterance is translated into a sequence of source vectors that can be used to reconstruct the original signal. Each source vector is encoded (or approximated) by a representative vector, called a code-word. The sound is therefore altered, but remains very close to the original. A collection of code-words is called a code-book. The utterance can be stored much more efficiently as a sequence of indices to these code-words rather than storing all the source vectors.

To apply VQ to speaker recognition, each speaker is represented by a separate code-book. The code-words in a code-book are calculated from training speech that is representative of the corresponding speaker. This is normally done by performing some kind of clustering on the source vectors of the training speech. It is always aimed to minimise the error (or distortion) between the code-book and the training speech. VQ can also be used to model speaker voices with feature vectors extracted from the training speech. Speaker recognition is done by calculating which code-book shows the smallest error for a given test utterance.

In brief terms, an HMM consists of a sequence of connected state probability density functions (PDFs). Early implementations of HMMs for speech processing tasks made use of Gaussian PDFs as the state PDFs, but modern applications normally use GMMs. The state PDFs are connected to each other by transition probabilities. The latter indicates how likely it is for the model to change from one state to another. Applied to words, for example, each state would represent the acoustic properties of a given sound in the word. The transition probabilities would then determine how likely it is that a speech sample of the given word will contain a sound of one state followed by a sound of another state.

Speaker recognition applications that use HMMs are typically text-dependent. The HMM aims to model the sounds and transitions between sounds for specific words spoken by a speaker. Speaker recognition is performed by determining how likely it is that the test utterance was generated by the corresponding HMM for each speaker. The HMM approach produced much better performance than the previous template-based approaches.

The GMM is widely used in modern, top-performing speaker recognition systems. Simply put, the GMM is a PDF of which the result is the sum of a set of weighted Gaussian PDFs, called mixture components. Many researchers prefer the GMM for acoustic speaker recognition, because it is a reasonably simple, but very effective model and has an intuitive connection to the distribution of the data. It also fits well into the likelihood ratio test because it is a stochastic (statistical) model.

The mixture components are thought to model underlying acoustic classes in a speaker's voice. No long-term transition characteristics (i.e. for sequences of more than two sounds) are modelled as with the HMM. Speaker recognition determines how likely it is that the sounds present in a test utterance were generated by the GMM of a given speaker. The work presented in this thesis is based on the GMM.

Because training data for each speaker are quite scarce, this thesis adopts the technique of Bayesian speaker adaptation [5, 6]. A good speaker model can be trained by using its limited amount of training data to adjust the parameters of a general speaker model, also called a UBM. The UBM itself is trained with a large amount of data from many different speakers and serves to represent speakers in general.

2.4 Tree-based Approaches

Similar to many other speech processing techniques, the application of tree-structured PDFs to the speech recognition task [7] predates its application to the speaker recognition task. Speech recognition systems typically work with models of words, phonemes or syllables in the form of HMMs.

A tree structure was applied to speech recognition [7] by regarding all the component Gaussian PDFs of every state-GMM and every HMM. This large set of element PDFs were clustered into groups and each group was clustered again. For each cluster, and at each clustering iteration, the parameters of a single Gaussian cluster PDF was calculated to serve as an approximation to the element PDFs in the cluster. By repeating this procedure, a tree was constructed where each cluster-PDF is attached to a node of the tree.

When applying a test feature vector to the speech recognition system, all the mixture (component) PDFs in all the HMMs will be calculated to determine how well any of the HMMs describes the sequence of vectors that has been observed. The tree-structure makes this process more efficient by calculating only those element PDFs that are likely to have a large likelihood value for the particular feature vector. These PDFs are found (at the leaf-node level) by

performing a search through the tree: at each level, only the child nodes of the cluster PDFs with the highest likelihoods are selected for further evaluation. All other element PDFs are approximated by their respective cluster PDFs, which are calculated as part of the search procedure.

More recently, a tree structure was applied to speech recognition for improving model adaptation efficiency in speech recognition [8]. This technique was named structural maximum *a posteriori* (SMAP) adaptation. As described above, all the mixture PDFs are clustered into groups. For each group the parameters of a normalised representative (or summarised) Gaussian PDF are calculated. These representative PDFs are then grouped and summarised further until a full tree is constructed. During adaptation, the summarised PDFs are used as prior PDFs for the group of nodes that they represent. This method provides improved adaptation performance for situations where very little adaptation data are available.

Motivated by this tree-based approach, an hierarchical GMM (HGMM) was developed with which the SMAP technique was applied to speaker verification [9]. This HGMM takes a regular GMM and replaces each component Gaussian PDF with a local GMM. The HGMM therefore models the feature space with more precision than the original GMM. Each of the original mixture components serves as a root node for the local GMM it represents. In other words, the HGMM has two levels: the level for root nodes, and the level for leaf nodes making up the local GMMs.

The GMMs that are used for speaker recognition can be quite large. The evaluation of all the mixture components in a model can therefore be very time consuming. Many researchers apply a simple technique for improving model evaluation speed [6]. This technique reduces the required number of mixture component evaluations, based on two principles. Firstly, for large GMMs, it can be determined that only a handful of components contribute significantly to the model likelihood. Secondly, speaker models are typically adapted from a UBM and therefore retain some correspondence with the components of the UBM. When a feature vector is introduced to the verification system for evaluation, the UBM is evaluated first to find the C components with the highest contribution to the UBM likelihood. Only the corresponding components in the specific speaker models are then evaluated for the same feature vector. This technique will be referred to here as UBM-based selection.

The structural GMM (SGMM) was developed as a new approach to improve the speed of model evaluation in speaker verification [10]. The SGMM is defined with a more general tree structure than the HGMM, but thus far results have

only been published for SGMM systems using a tree-structure that is very similar to the HGMM. The SGMM does not incorporate SMAP adaptation, but it applies an artificial neural network (ANN) for calculating the final scores. The SGMM is initialised through hierarchical clustering where a group of mixture components are summarised by a single Gaussian PDF.

The SGMM-based system uses the same principles as the UBM-based selection by first finding the C components with the highest likelihood contributions from the structured background model (SBM). But, unlike the GMM, these components can be found much more quickly by searching through the tree. The search is performed by finding in each level the node with the highest score (likelihood contribution) and evaluating only the children of that node further. The nodes with the highest scores in each level of the SBM, including the C components found at the leaf-node level, are used for the background (impostor) scores by the ANN. For each target model only those nodes that correspond to the background score nodes in the SBM are evaluated and used for the target scores by the ANN. The ANN is trained to distinguish target speakers from the background by using the mentioned node scores. The SGMM system reduces the computational cost by a factor of 17 relative to the UBM-based selection for GMMs. It also shows little degradation in verification performance when not using the ANN.

2.5 Performance Differences

It is normally worthwhile to get a general idea of how different techniques perform relative to each other. This allows a researcher to choose an area of research where effort would be well-spent.

In general, it is quite difficult to compare different results that are reported by different researchers. This difficulty is caused by factors such as the kind of data employed, the different techniques used and non-standard performance measurements. A few performance results are quoted here for the modelling techniques discussed above. Only reports that provide comparative results are quoted.

2.5.1 GMM versus VQ

The introduction of the GMM to speaker recognition [2] was accompanied by comparative results for a number of modelling strategies. The two techniques of concern here are VQ and the GMM. The GMM system made use of 64 mixture components and the VQ system made use of 100 code-words. Both

systems were applied to the same speaker identification (SID) task under the same conditions. The KING corpus containing clean and telephone-quality conversational speech from 51 male speakers were used. Identification accuracies are shown in Table 2.1.

System	Accuracy
VQ-100	92.9%
GMM-64	94.5%

Table 2.1: Speaker Identification Accuracies for VQ and GMM

2.5.2 GMM versus HMM

An investigation into fusion techniques for speaker verification [37] also produced comparative performance results for GMM- and HMM-based systems. Three experiments were conducted on three different databases that were collected in-house by the researchers. The first database contains 10 enrolled male target speakers and 80 development speakers. Each target speaker was enrolled with a unique pass-phrase. The second database contains 56 enrolled target speakers and 47 development speakers. All target speakers enrolled with the same phrase (fixed-text). The third database contains 26 enrolled target speakers and 15 development speakers. All of these target speakers also enrolled with the same phrase, but using a cellular telephone. Thresholds were chosen so that systems operated with 0% false rejections. Only false acceptance rates were therefore reported, and for the HMM and GMM systems, these are shown in Table 2.2. The GMM and HMM systems were subjected to the same conditions.

System	Unique pass-phrase	Fixed-text	Fixed-text cellular
HMM	5.6%	2.8%	9.6%
GMM	5.0%	3.5%	9.6%

Table 2.2: False acceptance rates for HMM and GMM where the false rejection rate is 0%.

These results suggest that the HMM only performs well when all target speakers are enrolled with the same utterance. It confirms that the HMM is not particularly suitable for text-independent speaker recognition applications.

2.5.3 HGMM versus GMM

The experiment reported for the HGMM [9] made use of the data from the 1999 National Institute for Standards and Technology (NIST) speaker recognition evaluation (SRE). This corpus contains telephone-quality conversational speech from 230 male and 309 female speakers. The GMM system made use of 1024 mixture components. The HGMM system had 1024 leaf-nodes, but clustered into 4 groups, with each group represented by a root node. The GMM and HGMM systems were applied to the same task under the same conditions. The equal error rates (EER) for both systems are shown in Table 2.3.

System	EER
GMM-1024	14.67%
HGMM-4x256	12.00%

Table 2.3: Speaker verification EER for GMM and HGMM

These results indicate that SMAP definitely provides some advantage. Smaller versions of these two systems were also applied to the 2002 NIST Multi-modal Speaker Recognition Development corpus for both speaker verification and identification. The results are shown in Table 2.4.

System	EER	SID Accuracy
GMM-512	6.3%	95.2%
HGMM-4x128	4.3%	96.3%

Table 2.4: Speaker recognition results for GMM and HGMM using NIST 2002 Multi-modal data

2.5.4 SGMM/SBM versus GMM

The SGMM was introduced [10] with an experiment that made use of NIST SRE data from three different years. Four hours of speech from 240 male and 240 female speakers from the 1998 NIST SRE data (Switchboard II, phase 2) were used to train two gender-dependent SBMs. The 1997 NIST SRE data (Switchboard II, phase 1) were used as development test data to train the parameters used in making the verification decision. The systems were evaluated on the 1999 NIST SRE data (Switchboard-II, phase 3).

The GMM system is a special case of the SGMM system having only one tree level. All the systems have the same number of leaf-nodes (1024). Only a

few results are shown here. The SGMM was developed to improve computation speed and therefore a computational reduction factor F is included in the results shown in Table 2.5. These results were determined for systems not using the ANN for making decisions.

System Structure	EER	F
1x1024 (GMM)	12.9%	1
1x2x256	13.1%	2
1x4x128	13.3%	3
1x4x8x32	13.5%	17

Table 2.5: EER and computational reduction factor (F) for SGMM

The results show that the SGMM performs slightly worse than the GMM. However, a multi-level perceptron (MLP) ANN was applied to the systems to improve performance. The results for these modified systems are shown in Table 2.6.

System Structure	EER
1x1024 (GMM) + MLP	12.7%
1x4x8x32 SGMM + MLP	12.1%

Table 2.6: EER for GMM and SGMM augmented with MLP

2.6 Summary

This chapter discussed various topics available in literature relevant to speaker verification and speaker modelling. It was learned that the GMM provides the best verification performance in most circumstances. By arranging mixture components into a tree, either performance can be improved with SMAP, or speed can be improved by a fast search through the tree.

These tree-based approaches served as motivation for the work of this thesis. Although there is probably not much more that can be done for improvement, a few inadequacies exist in current methods:

1. All tree-based methods perform some kind of clustering and Gaussian distributions merging or approximation on the mixture components. This can add to the computational cost, because this clustering and approximating must be performed in addition to training the regular GMM.

However, the added cost is probably not very large compared to the possible speed gain.

2. Current tree-based methods use shallow trees with few levels and many children per node. This can be advantageous when many models are evaluated for the same feature vector and UBM-based selection is applied. But, UBM-based selection has the potential of degrading performance when nodes do not correspond well between the UBM and specific speaker models. When only a few models are evaluated per feature vector, a shallow tree does not improve speed as much as is potentially possible.
3. During evaluation of current tree-based models, only the nodes with the highest likelihood contributions are considered. Two nodes can potentially have very similar contributions, but only the largest one will be used and important information can be lost in this way. In one instance (for speech recognition) [7] the leaf-nodes are approximated by their ancestors when those ancestor-nodes have low contributions. But, in most cases the lower contributions are simply discarded. This behaviour is somewhat justified, because a shallow tree will cause much more nodes to be evaluated per speaker model if the lower contributions for levels closer to the root node are included.
4. It seems that the SGMM employs a MLP to compensate for the degraded performance due to the discarding of lower contributions that possibly convey important information. This does not feel like a very elegant approach to improve performance.

This thesis presents a new tree-based model for speaker verification that aims to address these inadequacies to some extent. It uses a binary tree structure, although in general it can have any shape. The binary tree is perfectly suited for applying the technique of binary-split model initialisation. The binary-split algorithm is one of many techniques that can be used to initialise and train a regular GMM. So, when it is applied to a tree-based model, this technique does not require an initial regular GMM to be trained first. The tree is constructed directly from the data.

Although a binary tree might require more levels to represent a given number of mixture components, only two nodes have to be evaluated per level during the search. This allows specific speaker models to be evaluated quickly, without needing to resort to UBM-based selection. When comparing two sibling nodes, the children of the node with the highest likelihood contribution are evaluated

further. But, unlike most other methods, the lower likelihood contribution of the other node is included in the final model likelihood. This lower contribution serves to model the underlying feature space with a single low-resolution PDF instead of multiple high-resolution PDFs.

The effect is that a final likelihood is produced for a virtual GMM that models feature space at varying resolutions, depending on the location of the test feature vector. This gives the model an adaptive characteristic and is therefore called a tree-based adaptive Gaussian mixture model (T-BAG mixture model, or T-BAGMM).

Including the lower contributions does not add to the computational cost, because those contributions are calculated as part of the tree search. But, inclusion of the lower contributions prevents the loss of potentially important information, such as when two sibling nodes have very similar contributions. It will be shown in Section 6.4 that a parameter can be included in the evaluation so that the children of the node with the lower contribution are also evaluated further when the lower contribution is very similar to that of the sibling node.

Using such a virtual GMM, which is still regarded as a valid PDF, is considered to be an elegant solution for retaining important information, or preventing performance degradation. When few models (eg. only one target and one impostor speaker model) need to be evaluated per feature vector, the T-BAGMM will also have a much lower computational cost than other methods that use fewer tree levels.



Chapter 3

Modelling Acoustic Properties of Speaker Identity

3.1 Introduction

The Gaussian mixture model (GMM) is a theoretical model that is used widely in both speech recognition and speaker recognition applications. The main attraction is its ability to model certain acoustic properties in speech. For speech recognition, it can model the sounds present in specific words or phonemes. For speaker recognition, it is used to model the sounds present in a person's voice.

This thesis covers techniques surrounding the GMM with regard to the application of speaker verification. In this chapter, Section 3.2 provides a preliminary introduction into the mechanics of the speaker verification task. Section 3.3 indicates how the GMM is integrated into the speaker verification system. Section 3.4 describes the details of the GMM, while Section 3.5 explains how the parameters of the model are determined.

3.2 Speaker Verification Overview

The speaker verification system aims to determine whether a person is whom he or she is claimed to be, based on the speech of that person. Because very little is known about the relationship between speech signals and the identity of the corresponding speakers, statistical methods provide a very appropriate and effective framework for this task. Given the speech segment or utterance \mathbf{X} , and a hypothesised speaker S (the target speaker), verification can accordingly

be formulated as a simple test between two hypotheses:

$$H_0 : \mathbf{X} \text{ was spoken by } S$$

$$H_1 : \mathbf{X} \text{ was not spoken by } S.$$

In this thesis, it is assumed that the speech segment contains speech from only one person. This is called single-speaker detection. Multi-speaker detection applies to speech segments containing speech from more than one person.

As will be described in Section 5.3.1, the optimum test to decide between H_0 and H_1 , is the likelihood ratio test

$$\frac{p(\mathbf{X}|H_0)}{p(\mathbf{X}|H_1)} \begin{cases} \geq \theta & \text{accept } H_0 \\ < \theta & \text{reject } H_0, \end{cases}$$

given that the likelihood functions $p(\mathbf{X}|H_0)$ and $p(\mathbf{X}|H_1)$ are known exactly. The likelihood ratio is compared to a threshold value θ to decide whether or not the null hypothesis H_0 is true.

In practical cases, the likelihood functions in the ratio are not known. A primary goal in implementing a speaker verification system, is to find estimates for these functions that make the decision as accurate as possible. Statistical pattern recognition approximates the actual likelihood functions with theoretical models that are mathematically described using probability density functions. The GMM is one such theoretical model. It is described in detail in the rest of this chapter. Other techniques such as discriminative modelling (eg. artificial neural networks) can also be applied to determine the values of the likelihood functions.

The parameters of theoretical models are estimated from training data. This training data, as well as the test data used in the likelihood ratio, are normally processed in the form of features. These features are chosen for their ability to help distinguish between speakers. It is required that features contain as much speaker-dependent information as possible, while discarding other kinds of redundant information. The process of extracting appropriate features from raw speech signals is covered in Section 5.2.

It is typical to work with the logarithms of the likelihood functions (also called log-likelihoods) for computational efficiency. This reduces the likelihood ratio to a subtraction operation

$$\Lambda = \log p(\mathbf{X}|H_0) - \log p(\mathbf{X}|H_1),$$

which is compared to an appropriate threshold. The result of the log-likelihood

ratio is called the score Λ . It indicates how well the test speech matches the model of the target speaker. Modern speaker verification systems employ the technique of score normalisation to allow a single threshold for all evaluations. In this thesis, Test normalisation (T-Norm) is used. T-Norm requires the evaluation of a set of impostor models, in addition to the target model, to obtain the final score. This is discussed with more detail in Section 5.3.2.

Figure 3.1 illustrates how the components in the verification process fit together.

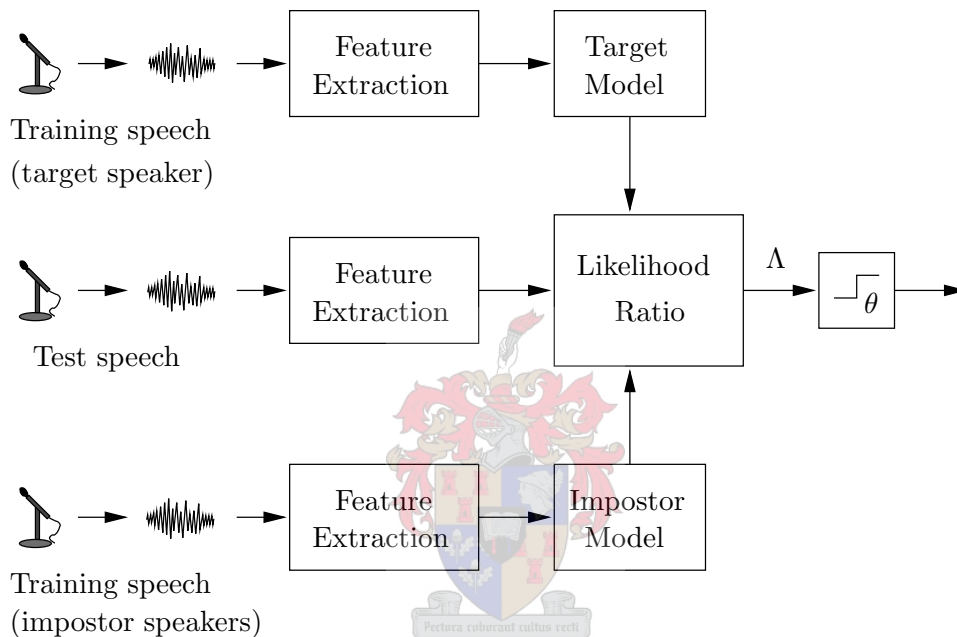


Figure 3.1: Simplified block diagram of the speaker verification system.

3.3 Speaker Modelling Overview

The previous section mentioned that speech processing is performed on features. Features are normally grouped into multi-dimensional vectors, called feature vectors. Feature vectors represent certain characteristics of the speech signal at certain time instances. Mathematically, they can be considered to be points in a multi-dimensional space, called feature space. A theoretical model such as the GMM tries to describe the stochastic (random) process responsible for generating observed points. In doing so, it tries to predict where future unseen points will be located. In probabilistic terms, the model can indicate the probability that a given point was generated by the modelled process. A model

giving results that match the truth is obviously desired. However, theoretical models can only provide approximations to reality.

The likelihood function $p(\mathbf{X}|H_j)$ that was introduced in the previous section indicates how likely it is that the hypothesis H_j is true, given that the data \mathbf{X} were observed. An alternative likelihood function $p(\mathbf{X}|\lambda_j)$ indicates how likely it is that λ_j are the true parameters for the model that was supposed to have generated the observed data \mathbf{X} . This likelihood can be used as a practical substitute for the theoretical likelihood functions in the likelihood ratio.

Choosing an appropriate model typically involves a trade-off between accuracy and the number of parameters. The Gaussian mixture model (GMM) is a PDF with potentially many parameters, although less than a histogram-based model for example. It has been shown by numerous examples [2, 6, 21] that the GMM can produce very high accuracies compared to other models in text-independent speaker recognition tasks. It can therefore be seen as a good trade-off between accuracy and the number of model parameters.

3.4 Theory of Gaussian Mixture Modelling

Mathematically, a GMM is a PDF that is defined as the sum of K weighted component Gaussian PDFs [2]. For the D -dimensional vector \mathbf{x} , the GMM representation of a speaker with model parameters $\lambda = \{w_k, \boldsymbol{\mu}_k, \Sigma_k : k = 1, 2, \dots, K\}$ is

$$p(\mathbf{x}|\lambda) = \sum_{k=1}^K w_k p(\mathbf{x}|\lambda_k) = \sum_{k=1}^K w_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k). \quad (3.1)$$

The weight w_k (or relative importance) of the k^{th} mixture component obeys the constraints $0 \leq w_k \leq 1$ and $\sum_{k=1}^K w_k = 1$. This ensures that the function will be a valid PDF. For each component, the mean and covariance parameters are collectively represented by $\lambda_k = \{\boldsymbol{\mu}_k, \Sigma_k\}$. The PDF of the k^{th} component is given by the multivariate (multi-dimensional) Gaussian density

$$\begin{aligned} p(\mathbf{x}|\lambda_k) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k) \\ &= \frac{1}{(2\pi)^{D/2} |\Sigma_k|^{1/2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right], \end{aligned} \quad (3.2)$$

where $(\mathbf{x} - \boldsymbol{\mu}_k)^T$ is the transpose of $(\mathbf{x} - \boldsymbol{\mu}_k)$.

The GMM can be visualised by the block diagram in Figure 3.2. A feature vector \mathbf{x} is evaluated on each of K Gaussian PDFs p_k , $k \in \{1, 2, \dots, K\}$ to produce component likelihoods that are weighed and summed to give the model likelihood $p(\mathbf{x}|\lambda)$.

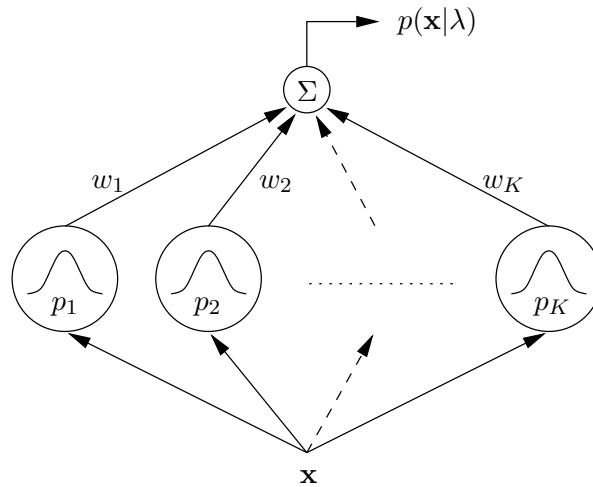


Figure 3.2: Block diagram representing a GMM. Each of the Gaussian PDF components p_k are evaluated for the test feature vector \mathbf{x} . Their resultant likelihoods are then weighed and summed to produce a model likelihood $p(\mathbf{x}|\lambda)$.

GMMs are well suited for applications where the data of a given model have many modes (or local maxima). Modes are areas where data points are much more densely packed than in the local surroundings. Each component of the GMM can model a local mode with a single Gaussian PDF.

GMM components can use three kinds of covariance matrices: full covariance, diagonal covariance and spherical covariance. Full covariance matrices come directly from theory. They consist of diagonal elements that indicate the variance of the data in each dimension of feature space, and non-diagonal elements that indicate correlation between the dimensions. For a diagonal covariance matrix only the elements on the diagonal of the full covariance matrix are kept. The non-diagonal elements are set to zero. Diagonal covariance can only be applied properly if there is no correlation between the different dimensions in feature space. Spherical covariance matrices are represented by a single variance value. This value is used for all the elements on the diagonal of the matrix and all non-diagonal elements are zero. It can be used when it is assumed that there is no correlation between the individual dimensions in feature space, and that the variance is the same for all dimensions. It can be concluded that the choice in the type of covariance depends on the accuracy and memory requirements as well as the application.

There are two main reasons why GMMs are used in speaker verification systems [2]. It is reasonable to assume that a person's voice can be characterised by a set of acoustic classes, such as the different phonetic classes. It

is suggested that the individual component densities in a multi-modal density, such as a GMM, can model these underlying acoustic classes. Furthermore, the acoustic classes indicate certain vocal tract configurations that may be speaker-dependent. From the use of cepstral feature vectors, it can be argued that the mean parameter μ_k represents the average spectral shape of the k^{th} acoustic class. Similarly, the covariance matrix Σ_k represents the possible variations from the average spectral shape.

The second reason for using GMMs, is that it can provide smooth approximations of densities that have arbitrary shapes. It is even possible to use diagonal covariance matrices when there are correlations between the dimensions in feature space. A linear combination of diagonal covariance Gaussian PDFs is able to model such correlations. The only difference is that more mixture components are required than when using full covariance matrices.

3.5 Model Training

Without an exact understanding of the actual data generating process, the only method of determining the model parameters, is by regarding observed data. Therefore any model is only as effective as the selection of observed data that are used to determine its parameters. Because the correct set of parameters are unlikely to be found from only a subset of all possible observations, it is said that the parameters are estimated from the data. The data used for parameter estimation are called training data.

The parameters of models used in speaker verification systems are usually determined by a supervised training process. In other words, the identities corresponding to the data are known at training time, and used accordingly to train models. When unsupervised training is used, the system must determine on its own how many identities there are and to which ones the data belong.

To train the parameters of a specific speaker model, as much speech data for that person must be gathered as possible. For modelling with GMMs, all the features that were extracted from the speaker's speech signals can be pooled into one large speaker-specific set. The GMM makes no provision for time-order (temporal) information.

3.5.1 Maximum Likelihood Estimation

Maximum Likelihood (ML) estimation is a very popular method for finding model parameters. It gives very good results when the training data sets of speakers are large.

Recall from Section 3.3 that the model likelihood $p(\mathbf{X}|\lambda)$ indicates how likely it is that λ are the true parameters for the model that was supposed to have generated the observed data \mathbf{X} . Note that λ is the independent variable and the data \mathbf{X} are known and fixed. Suppose that a model with a given set of parameters λ_j must be used to describe all possible data generated by a specific person. The correct set of parameters must be found so that the likelihood $p(\mathbf{X}|\lambda_j)$ of the model is higher than the likelihood of any other model, given speech from the corresponding person. Of course, the best that can be done is to find the set of parameters $\tilde{\lambda}$ that maximises the likelihood using the training data. For this reason, a large amount of training data is desired to cover as much of a person's acoustic characteristics as possible.

Formally, the ML-estimated parameters $\tilde{\lambda}$ are obtained for the sequence of observed feature vectors $\mathbf{X} = \{\mathbf{x}_t : t = 1, 2, \dots, T\}$ by maximising the model likelihood with respect to the parameters λ :

$$\tilde{\lambda} = \arg \max_{\lambda} p(\mathbf{X}|\lambda).$$

For simple models, this can normally be done by solving for λ in

$$\frac{d}{d\lambda} p(\mathbf{X}|\lambda) = 0.$$

Acoustic speaker modelling with GMMs assumes that the feature vectors \mathbf{X} of a certain speaker are statistically independent and identically distributed (IID). This means that the vectors were produced independently from one another by the same PDF. The joint likelihood of the individual feature vectors then becomes the product of their likelihood functions:

$$p(\mathbf{X}|\lambda) = \prod_{t=1}^T p(\mathbf{x}_t|\lambda). \quad (3.3)$$

For analytical simplicity, it is often useful to work with the log-likelihood¹

$$\log p(\mathbf{X}|\lambda) = \sum_{t=1}^T \log p(\mathbf{x}_t|\lambda). \quad (3.4)$$

This can still be maximised, because the logarithm is a monotonically increasing function.

¹As noted in Section 5.3.1, the natural logarithm (base- e) is normally applied.

For GMMs, $p(\mathbf{x}_t|\lambda)$ is given by eq. (3.1), which results in the log-likelihood

$$\log p(\mathbf{X}|\lambda) = \sum_{t=1}^T \log \left(\sum_{k=1}^K w_k p(\mathbf{x}_t|\lambda_k) \right).$$

Unfortunately, it is very difficult to maximise this function because it contains the logarithm of a sum. The next section explains how the ML estimates can be obtained through an iterative procedure by extending the observed feature vector data with estimates of so-called hidden component labels.

3.5.2 Expectation-Maximisation

When some data are missing, ML estimates of the model parameters λ can be obtained with the Expectation-Maximisation (EM) algorithm [38]. This method can be applied to the estimation of GMM parameters by assuming that the observed data can be extended with missing (hidden) component labels into what is called complete data [39]. The algorithm consists of iterating through two steps:

1. the Expectation step (E-step) determines the expectation of the complete-data likelihood with respect to the hidden data, and
2. the Maximisation step (M-step) maximises the expectation of the complete-data likelihood with respect to the unknown parameters λ .

Iteration is stopped when some convergence criterion is met.

To apply the EM algorithm to GMMs, consider the GMM from a data generating perspective. Each mixture component can be regarded as independently modelling a subpopulation ψ_k of the data, described by the PDF $p(\mathbf{x}|\lambda_k)$. Any subpopulation can generate an observation vector \mathbf{x} with a probability that is determined by the product of the corresponding PDF and a prior probability $P(\psi_k) = w_k$ (from the Bayes formula). Consequently, the GMM determines the probability that the vector was generated by any one of its subpopulations.

Let $\mathcal{Y} = \{Y_t : t = 1, 2, \dots, T\}$ be the set of unknown discrete random variables indicating the labels of the components that were responsible for generating the observed feature vectors $\mathbf{X} = \{\mathbf{x}_t : t = 1, 2, \dots, T\}$. Assume the existence of a complete set of data $\mathcal{Z} = \{\mathbf{X}, \mathcal{Y}\}$. Define $\lambda^{[i]}$ as the estimates for the model parameters at iteration i . The EM algorithm starts by selecting an appropriate initial set of model parameters $\lambda^{[i=0]}$.

The E-step finds the expected value of the complete-data log-likelihood with respect to the unknown data \mathcal{Y} , given the known data \mathbf{X} and the current parameter estimates:

$$Q\left(\lambda|\lambda^{[i]}\right) = E\left[\log p(\mathcal{Z}|\lambda) | \mathbf{X}, \lambda^{[i]}\right] = E\left[\log p(\mathbf{X}, \mathcal{Y}|\lambda) | \mathbf{X}, \lambda^{[i]}\right]. \quad (3.5)$$

$Q\left(\lambda|\lambda^{[i]}\right)$ is called the auxiliary function. It must be understood here that \mathbf{X} and $\lambda^{[i]}$ are known constants and λ is a regular variable that must be optimised. From the definition of expected values [13, 39], eq. (3.5) can be written as ²

$$\begin{aligned} Q\left(\lambda|\lambda^{[i]}\right) &= E\left[\log p(\mathbf{X}, \mathcal{Y}|\lambda) | \mathbf{X}, \lambda^{[i]}\right] \\ &= \int_{\mathcal{L}} (\log p(\mathbf{X}, \mathbf{y}|\lambda)) p(\mathbf{y}|\mathbf{X}, \lambda^{[i]}) d\mathbf{y}. \end{aligned} \quad (3.6)$$

The vector variable $\mathbf{y} = [y_1 \ y_2 \ \dots \ y_T]^T$ represents realisations of the random variables in \mathcal{Y} that are governed by the joint PDF $p(\mathbf{y})$. Integration is performed over \mathcal{L} , which is the space of all possible values for \mathbf{y} (i.e. all combinations of labels).

The function $\log p(\mathbf{X}, \mathbf{y}|\lambda)$ in eq. (3.6) is the complete-data log-likelihood when the labels are known. This can be written as

$$\begin{aligned} \log p(\mathbf{X}, \mathbf{y}|\lambda) &= \log \prod_{t=1}^T p(\mathbf{x}_t, y_t|\lambda) \\ &= \sum_{t=1}^T \log p(\mathbf{x}_t, y_t|\lambda). \end{aligned} \quad (3.7)$$

The rules of conditional probability can be used to show that

$$\begin{aligned} p(\mathbf{x}_t, y_t|\lambda) &= \frac{p(\mathbf{x}_t, y_t, \lambda)}{P(\lambda)} \\ &= \frac{p(\mathbf{x}_t|y_t, \lambda) P(y_t, \lambda)}{P(\lambda)} \\ &= \frac{p(\mathbf{x}_t|y_t, \lambda) P(y_t|\lambda) P(\lambda)}{P(\lambda)} \\ &= p(\mathbf{x}_t|y_t, \lambda) P(y_t|\lambda). \end{aligned}$$

For GMMs, it is known that $p(\mathbf{x}_t|y_t, \lambda) = p(\mathbf{x}_t|\lambda_{y_t})$, because if both the labels and the parameters are known, then the parameters of the corresponding mixture components are known and can be used as substitutes in the expres-

²Recall that $E[g(Y)|A] = \int_{-\infty}^{\infty} g(y) p(y|A) dy$.

sion. Similarly, it is also known that $P(y_t|\lambda) = P(Y_t = y_t|\lambda) = w_{y_t}$, being the weight (or prior probability) of the corresponding mixture component. Therefore, eq. (3.7) results in

$$\log p(\mathbf{X}, \mathbf{y}|\lambda) = \sum_{t=1}^T \log(w_{y_t} p(\mathbf{x}_t|\lambda_{y_t})).$$

To evaluate eq. (3.6), it is also necessary to find an expression for the PDF $p(\mathbf{y}|\mathbf{X}, \lambda^{[i]})$. Using the data \mathbf{X} and the current parameters $\lambda^{[i]}$, it is easy to calculate the mixture component likelihoods $p(\mathbf{x}_t|\lambda_k)$ for every $t \in \{1, 2, \dots, T\}$ and $k \in \{1, 2, \dots, K\}$. Because the weight of the k^{th} component is the prior probability $w_k = P(Y_t = k|\lambda)$, the Bayes rule can be used to obtain the PDF for the unobserved labels

$$p(y_t|\mathbf{x}_t, \lambda^{[i]}) = \frac{p(\mathbf{x}_t|y_t, \lambda^{[i]}) P(y_t|\lambda^{[i]})}{p(\mathbf{x}_t|\lambda^{[i]})} = \frac{w_{y_t}^{[i]} p(\mathbf{x}_t|\lambda_{y_t}^{[i]})}{\sum_{k=1}^K w_k^{[i]} p(\mathbf{x}_t|\lambda_k^{[i]})}. \quad (3.8)$$

This gives the result

$$p(\mathbf{y}|\mathbf{X}, \lambda^{[i]}) = \prod_{t=1}^T p(y_t|\mathbf{x}_t, \lambda^{[i]}) = \prod_{t=1}^T \frac{w_{y_t}^{[i]} p(\mathbf{x}_t|\lambda_{y_t}^{[i]})}{\sum_{k=1}^K w_k^{[i]} p(\mathbf{x}_t|\lambda_k^{[i]})}$$

that can be substituted into eq. (3.6).

The M-step finds $\lambda^{[i+1]}$, the value of λ that maximises $Q(\lambda|\lambda^{[i]})$:

$$\lambda^{[i+1]} = \arg \max_{\lambda} Q(\lambda|\lambda^{[i]}).$$

The complete derivation [39] is mathematically quite demanding, but results in the following re-estimation formulas [2, 39]:

Weights:

$$w_k^{[i+1]} = \frac{1}{T} \sum_{t=1}^T c_{kt} \quad (3.9)$$

Means:

$$\boldsymbol{\mu}_k^{[i+1]} = \frac{\sum_{t=1}^T c_{kt} \mathbf{x}_t}{\sum_{t=1}^T c_{kt}} \quad (3.10)$$

Covariance matrices:

$$\Sigma_k^{[i+1]} = \frac{\sum_{t=1}^T c_{kt} (\mathbf{x}_t - \boldsymbol{\mu}_k^{[i+1]}) (\mathbf{x}_t - \boldsymbol{\mu}_k^{[i+1]})^T}{\sum_{t=1}^T c_{kt}}. \quad (3.11)$$

When using diagonal covariance matrices, as discussed in Section 3.4, eq. (3.11) reduces to:

$$\sigma_k^{2[i+1]} = \frac{\sum_{t=1}^T c_{kt} \mathbf{x}_t^2}{\sum_{t=1}^T c_{kt}} - \left(\boldsymbol{\mu}_k^{[i]} \right)^2. \quad (3.12)$$

The factor $c_{kt} = P(k|\mathbf{x}_t, \lambda^{[i]})$ is called the component responsibility. It is the probability that k is the label for \mathbf{x}_t , given the current parameters. In other words, it is the value obtained when setting $y_t = k$ in eq. (3.8). Note that all of the above re-estimation formulas perform both the E-step and the M-step simultaneously.

3.5.3 Model Initialisation

The EM-algorithm requires an initial set of model parameters. The choice of initial parameters is quite crucial because the EM-algorithm can converge to any local optimum, although convergence to the global optimum is always preferred. Unfortunately, there are no theoretical rules by which the best set of initial parameters can be determined.

The binary-split initialisation technique was used in this research to produce the desired number of GMM components. This technique does not strictly require any prior knowledge about the data distribution. It is an iterative procedure that works well with the EM-algorithm. The details [40] are given by Algorithm 1.

Step 3 of the algorithm requires some explanation. The PDF to select should be one with a large weight (i.e. it models a large subset of the data) and a large variance. The variance on the principal axis is typically used for this. However, performing eigen analysis on all the mixture components can be computationally expensive. The implementation used in this research calculates a score value equal to the mixture weight divided by the smallest diagonal element in the inverse covariance matrix. The component with the smallest score value is then selected for splitting.

3.5.4 Maximum a Posteriori (MAP) Adaptation

Normally, very little data are available for training the parameters of speaker models. This has led researchers to adopt the technique of speaker adaptation [6]. It is related to the technique that was originally used in automatic speech recognition (ASR) [5], where a model of general speech is adapted to the speech of a specific person. This is done in order to enhance recognition of words spoken by that person.

-
1. Choose any mean μ and covariance matrix Σ for a single Gaussian PDF to model all the data
 2. Perform one or more iterations of the EM-algorithm to obtain a better fit
 3. From all the PDFs produced after step 1, select the one with the "widest" variance
 4. Determine the principal axis (eigen vector) of the PDF selected in step 3 using Eigen analysis
 5. Calculate two new means located along the principal axis, at $+0.7979$ and -0.7979 standard deviations away from the original mean
 6. Reduce the variance along the principal axis to a fraction 0.363 of the original variance; make a copy of it
 7. Now there are two new means and two new covariance matrices corresponding to two new Gaussian PDFs; Repeat from step 3 until the number of Gaussian PDFs have doubled
 8. Repeat from step 2

Iteration can be stopped once the desired number of mixture components are obtained.

Algorithm 1: Binary-split model initialisation

For speaker recognition, however, speaker adaptation provides a way of extending the amount of available training data. A general speaker model, called a Universal Background Model (UBM), is trained with the speech of many different speakers using ML estimation. This model can be trained very well, because a large set of data is available. The UBM represents characteristics common to all speakers. This provides possible prior information of what a speaker model should look like in general. It is assumed that the UBM parameters need only to be adjusted in a minor way to produce a good model of any specific speaker.

Maximum *a Posteriori* (MAP) adaptation is a form of Bayesian adaptation. Where ML estimation maximises the likelihood, MAP estimation maximises the posterior probability in the Bayes formula, eq. (5.3). In doing so, prior knowledge is incorporated into the estimation procedure.

If the parameters λ are taken to be random, MAP estimation is formally stated as maximising the posterior PDF $p(\lambda|\mathbf{X})$ with respect to the parameters λ :

$$\begin{aligned}\hat{\lambda} &= \arg \max_{\lambda} p(\lambda|\mathbf{X}) \\ &= \arg \max_{\lambda} p(\mathbf{X}|\lambda) p(\lambda),\end{aligned}\quad (3.13)$$

where $p(\lambda)$ is the prior PDF of the parameters. The evidence factor $p(\mathbf{X})$ is not included because it is not a function of λ and therefore has no influence on maximising the posterior PDF. If it is assumed that there are no prior information about λ , then eq. (3.13) becomes the familiar ML estimation.

Others have determined empirically that the best results for speaker recognition are obtained by MAP adaptation of only the mean vectors in the GMM [6]. The same strategy is used in this thesis, although an experiment is also conducted in Chapter 7 that confirms the reports of others. The formal derivation of the MAP EM re-estimation formulas is quite involved [5], but the results (excluding weights adaptation) are:

Means:

$$\hat{\boldsymbol{\mu}}_k^{[i+1]} = \frac{\tau_k \boldsymbol{\mu}_{kp} + \sum_{t=1}^T c_{kt} \mathbf{x}_t}{\tau_k + \sum_{t=1}^T c_{kt}} \quad (3.14)$$

Covariance matrices:

$$\hat{\Sigma}_k^{[i+1]} = \frac{\tau_k \Sigma_{kp} \left(\sum_{t=1}^T c_{kt} \right) \Sigma_k^{[i+1]} + \tau_k \left(\boldsymbol{\mu}_{kp} - \hat{\boldsymbol{\mu}}_k^{[i+1]} \right) \left(\boldsymbol{\mu}_{kp} - \hat{\boldsymbol{\mu}}_k^{[i+1]} \right)^T}{\tau_k + \sum_{t=1}^T c_{kt}} \quad (3.15)$$

$\Sigma_k^{[i+1]}$ is the current ML estimate of the covariance matrix from the adaptation (training) data. $\boldsymbol{\mu}_{kp}$ is the prior mean and Σ_{kp} is the prior covariance matrix for the the k^{th} mixture component. These prior parameters are simply taken from the corresponding component in the UBM. τ_k symbolises a prior observation-count associated with the prior mean and is called the relevance factor. The factor $c_{kt} = P(k|\mathbf{x}_t, \lambda^{[i]})$ is defined at the end of Section 3.5.2 as the probability that k is the label for \mathbf{x}_t , given the current parameters. It is computed with eq. (3.8) by setting $y_t = k$.

MAP estimation supplements the scarce data of a specific speaker with the abundance of data used for training the UBM. The UBM parameters are weighted with a relevance factor. When more speaker data are available, the formulas tend to ignore the UBM contribution. When very little data are available, the model parameters are almost completely defined by the UBM.

An intuitive derivation [40] for the covariance matrix MAP re-estimation formula is given in Section A.2 of Appendix A. An implementation based on this derivation was produced and used during the execution of this research.

3.6 Summary

In this chapter, the likelihood ratio is briefly introduced as a solution to the speaker verification task. It was shown how the respective probabilities in the likelihood ratio can be described by theoretical speaker models. The theoretical aspects of the GMM as one such speaker model was discussed and some detail. Estimation of the GMM parameters from training data was also discussed. It was shown that specific speaker models can be adapted from a generalised speaker model or UBM to produce well-trained models even when training data are scarce.



Chapter 4

Tree-based Adaptive Gaussian Mixture Models

4.1 Introduction

4.1.1 The Speed Problem

Many mixture components are normally required when Gaussian mixture models (GMMs) are used for speaker verification. There can be as much as 2048 components per model when diagonal covariance matrices are used. Typical performance evaluations require thousands of trials. Each trial can require the evaluation of 100 or even more GMMs per test feature vector. This roughly equals the evaluation of up-to 200 000 Gaussian probability density functions (PDFs) per test feature vector. It is clear that a full performance evaluation requires an enormous computational capacity.

Section 5.4 will discuss how more than one processor can be employed to reduce the running time for a full performance evaluation. However, even when up-to 10 processors are used in this way, the execution of such an evaluation can take many days to complete. A number of approaches exist for reducing execution time by reducing the required number of Gaussian PDF evaluations. These techniques compute only the mixture components that contribute the most to the model likelihood. This contribution is defined as the component likelihood multiplied by its mixture weight. The only components that will have significantly large contributions, are those that model data in the vicinity of the test vector. The contributions of these selected components are then usually summed to obtain an approximation to the final likelihood of the model.

4.1.2 Popular Approach to Speed Improvement

To improve speed, many researchers use a technique that selects the C components with the highest expected likelihood contribution [6, 28]. This technique requires that the speaker models are adapted from a universal background model (UBM). For each test feature vector, all the components in the UBM are evaluated. The C components that contribute the most to the UBM likelihood are selected. It is then assumed that the corresponding components in the speaker-specific model will also have the highest contribution to the likelihood of that model. Let K be the number of mixture components per model and N be the number of models to evaluate (eg. for T-Norm) per test feature vector. This method requires the evaluation of $(K + N \times C)$ rather than $(N \times K)$ Gaussian PDFs per vector. Speed improvement is increased as N is increased.

Although it has been shown that this method is quite effective, it has a few obvious problems. In general, the resulting likelihood is only a rough approximation of the actual likelihood. Also, it might happen that the components of a speaker-specific model were altered considerably during adaptation. The components with the highest contribution to the likelihood of the speaker-specific model will therefore not necessarily correspond well to those of the UBM. Furthermore, the evaluation of all the UBM components still requires much computation time. It is also necessary to keep a copy of the UBM for use at each verification trial. Lastly, because the UBM is a required part of this technique, it can not be applied to some other tasks (eg. phoneme or word recognition) where it is not possible or practical to adapt models from a UBM.

4.1.3 Tree-based Approach to Speed Improvement

Another approach to reduce the number of Gaussian PDF evaluations consists of organising the mixture components into a tree structure [7, 8, 9, 10]. Each layer of the tree represents a more approximate or lower-resolution GMM for the data than the layers further away from the root node. As mentioned above, only the components that contribute the most to the model likelihood are evaluated. In this case, however, those components are found by traversing the tree from the root node. At each layer, the contributions (likelihood times weight) of sibling nodes are compared to each other. Only the subtree rooted at the node with the largest contribution is traversed further. These comparisons are repeated until the components of the regular GMM are reached in the leaf nodes. If the tree is constructed and trained properly, these leaf nodes will have the highest contributions to the model likelihood.

This type of search can be performed directly on each speaker-specific model. For models with large component counts, it requires much less computation time than the UBM-based selection approach. The actual time required depends on the specific structure of the tree, as shown in Table 4.1. Another important influence on the actual speed, is whether or not the tree is well balanced [41]. For a perfectly balanced binary tree, the number of Gaussian PDF evaluations are $(2 \log_2 K)$. The coefficient of 2 indicates that two evaluations must be performed for the comparison at each layer. There is no comparison made at the leaf node layer, but the selected leaf node PDFs must also be evaluated. Finding their contributions is, after all, the main reason for searching through the tree.

Children per node	No. of levels	Leaf node count	Number of Gaussian PDF evaluations
2	11	$2^{11} = 2048$	$2 \times 11 = 22$
3	7	$3^7 = 2187$	$3 \times 7 = 21$
4	6	$4^6 = 4096$	$4 \times 6 = 24$
5	5	$5^5 = 3125$	$5 \times 5 = 25$

Table 4.1: Theoretical trade-offs for different balanced tree structures. These figures were chosen so that there would be enough leaf nodes to accommodate 2048 mixture components.

The speed improvement can also be effective in speech recognition applications. It allows the use of more mixture components (and therefore higher accuracy) when operating at the same speed.

The main disadvantage of tree-based methods is the increased memory requirement. In modern times, however, this is not much of a problem, because computer memory is relatively inexpensive. Regardless of this, the speed improvement outweighs the cost of memory for speaker verification research. This is true because many trials must be performed for different system parameters. With regular GMMs this might take weeks or months, where tree-based GMMs need only hours or days. Another concern is that, in the unlikely case that the trees of many models are very badly balanced, the resulting speed improvement (if any) might not be worth the cost in effort and memory requirements.

4.1.4 The Tree-Based Adaptive GMM

The current tree-based techniques discard the contributions of nodes that were not followed during traversal of the tree. This thesis presents an alternative tree-based GMM that includes the lower contributions of approximate PDFs that were already evaluated. It was felt that more could be gained by making use of those otherwise discarded results. As a result, a virtual GMM is constructed of which the components model the underlying data at different resolutions, depending on their proximity to the test feature vector. This gives the model an adaptive property. Consequently, the new model is called a tree-based adaptive Gaussian mixture model (T-BAG¹ mixture model or T-BAGMM). The T-BAGMM produces a smooth approximation to the actual model likelihood.

The next section will discuss the theory of T-BAGMM construction and evaluation. Section 4.3 provides the details of how training is performed for T-BAGMMs.

4.2 Theory

In this thesis, the T-BAGMM is a binary tree structure (two children per node) as shown in Figure 4.1. Although Table 4.1 suggests that 3 children per node would provide the best improvement in speed, the binary structure was chosen for the sake of algorithmic simplicity. The binary tree structure is also perfect for the application of the binary-split model initialisation procedure (discussed in Section 4.3).

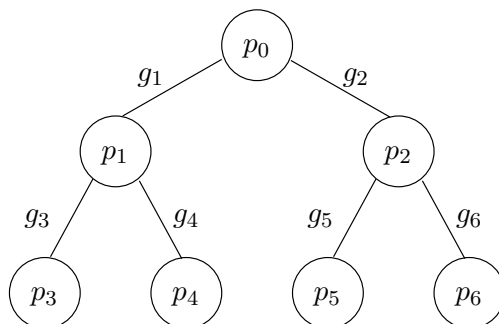


Figure 4.1: Tree-based GMM representation. Each node has a node weight g_i and a Gaussian PDF $p_i = p(\mathbf{x}|\lambda_i)$.

¹This acronym was selected from a few nominations, because it presents a humorous play on the fact that the DSP laboratory at the University of Stellenbosch is famously associated with high rates of coffee consumption.

Each node in the tree has a Gaussian PDF that models the collective data of the node's descendants with an approximate (or lower-resolution) Gaussian PDF. The leaf nodes correspond exactly to the components of the equivalent regular GMM.

Each node has a node weight g_i as well as a (virtual) mixture weight w_i . Two sibling nodes can be seen as the mixture components of a two-component sub-GMM that models a subset of the data. The *node weights* of the two sibling nodes are the mixture weights of this sub-GMM. The *node weights* of two sibling nodes therefore sum to unity. The *mixture weight* of a node is used when the node is selected to be a mixture component in a virtual GMM. It is calculated by multiplying all the *node weights* encountered when traversing back up the tree toward the root node. For example, the *mixture weight* of node 5 in Figure 4.1 is $w_5 = g_2 \times g_5$.

To make this calculation easier for algorithms, node weights can also be indexed according to the tree level. In this case, the level-based node weight g'_i is the node weight at the i^{th} tree level. This only makes sense when traversing back up the tree towards the root node. To present the same example as previously, the *mixture weight* of node 5 is now calculated as $w_5 = g'_1 \times g'_2$, where $g'_1 = g_2$ and $g'_2 = g_5$. To complement this tree level indexing, a function $L_T(k)$ can be defined to return the tree level of node k . For example, $L_T(5) = 2$ and $L_T(2) = 1$. These definitions will be used throughout the rest of the chapter.

The *mixture weights* always sum to unity for all possible virtual GMMs. It is perfectly possible to only use *mixture weights* in the tree structure. However, doing so will make weight estimation much more difficult. To estimate *mixture weights*, all the nodes in the same tree level must be accessed at the same time. It is clear that this would require too much overhead and more calculations than necessary.

The primary aim of the T-BAGMM is to obtain high evaluation speed by being able to select the mixture components with the highest contributions to the model likelihood in a very short time. The secondary aim of the T-BAGMM is to produce the score of a (virtual) GMM that models data with high detail in the vicinity of the test feature vector (in feature space) and lower detail in other regions. Here, *detail* refers to the number of mixture components that are used to model a given hyper-volume in feature space. The secondary aim is accomplished as a by-product of fulfilling the primary aim.

To find the nodes with the highest contributions to the model likelihood, a search must be performed on the tree nodes. The tree is traversed from the root node to the leaf nodes and each tree level is visited only once. It is assumed

that a higher contribution (likelihood of the node PDF multiplied by the node mixture weight) to the model likelihood will be provided by a node that models data in closer proximity to the test feature vector than other nodes do. This also applies to the approximate Gaussian PDFs described above.

Based on this assumption, two sibling nodes are compared at each level during the search. The children of the node with the highest contribution are compared to each other at the next level. The node with the lower contribution is selected to become a mixture component in the virtual GMM. Both siblings on the leaf-node level are selected to become mixture components in the virtual GMM. Selecting a node merely refers to the act of including its contribution in the summation of eq. (3.1). For obtaining valid search results, it is essential that the children of an approximate node collectively model the same data as their parent. An example of a virtual GMM is shown in Figure 4.2. The basic search procedure is described in more detail by Algorithm 2 and a simple visual example is given by Figure 4.3.

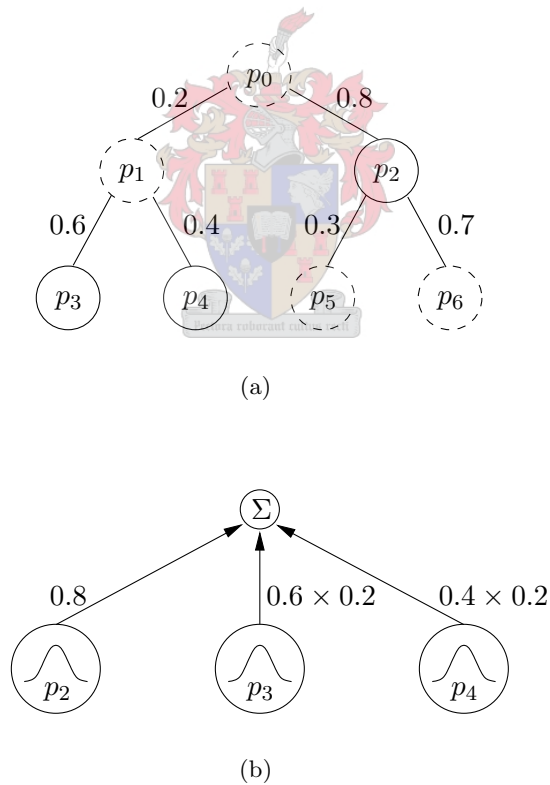


Figure 4.2: Example of a virtual GMM. (a) shows the original tree. Nodes with solid outlines were selected for the virtual GMM. (b) shows the virtual GMM that was constructed from the tree in (a). Note that the resulting mixture weights do add up to one.

-
1. Obtain test feature vector \mathbf{x}_t
 2. Start at root node
 3. If node is a leaf node, then return the likelihood of the node $p(\mathbf{x}_t|\lambda_k)$ using the node PDF; otherwise continue with step 4
 4. Calculate the **approximate** likelihood of the left subtree $p(\mathbf{x}_t|\lambda_{k,left})$ from the approximate PDF in the left child node; multiply mixture weight of left subtree $w_{k,left} = g_{k,left} \prod_{i=1}^{L_T(k)} g'_i$ with this likelihood to obtain left contribution
 5. Calculate the **approximate** likelihood of the right subtree $p(\mathbf{x}_t|\lambda_{k,right})$ from the approximate PDF in the right child node; multiply mixture weight of right subtree $w_{k,right} = g_{k,right} \prod_{i=1}^{L_T(k)} g'_i$ with this likelihood to obtain right contribution
 6. If left contribution \geq right contribution, then obtain **detailed** likelihood of left subtree $p(\mathbf{x}_t|\lambda_{k,left})$ by repeating from step 2 for left child node
 7. If right contribution \geq left contribution, then obtain **detailed** likelihood of right subtree $p(\mathbf{x}_t|\lambda_{k,right})$ by repeating from step 2 for right child node
 8. return $(g_{k,left}p(\mathbf{x}_t|\lambda_{k,left}) + g_{k,right}p(\mathbf{x}_t|\lambda_{k,right}))$

Notes on the algorithm:

- The returned value can be produced from any combination of approximate and detailed contributions. Hence, lower-resolution nodes are also included in the final model likelihood.
- The indices $k, left$ and $k, right$ refer to the left and right children of node k .
- In the unlikely situation that the contributions of both sibling nodes are equal, it is unknown which subtree should be evaluated in more detail. If an arbitrary selection is made to evaluate only one subtree, that selection might not be the correct one. The result is that more nodes will be evaluated than usual.

Algorithm 2: Recursive evaluation of a T-BAGMM

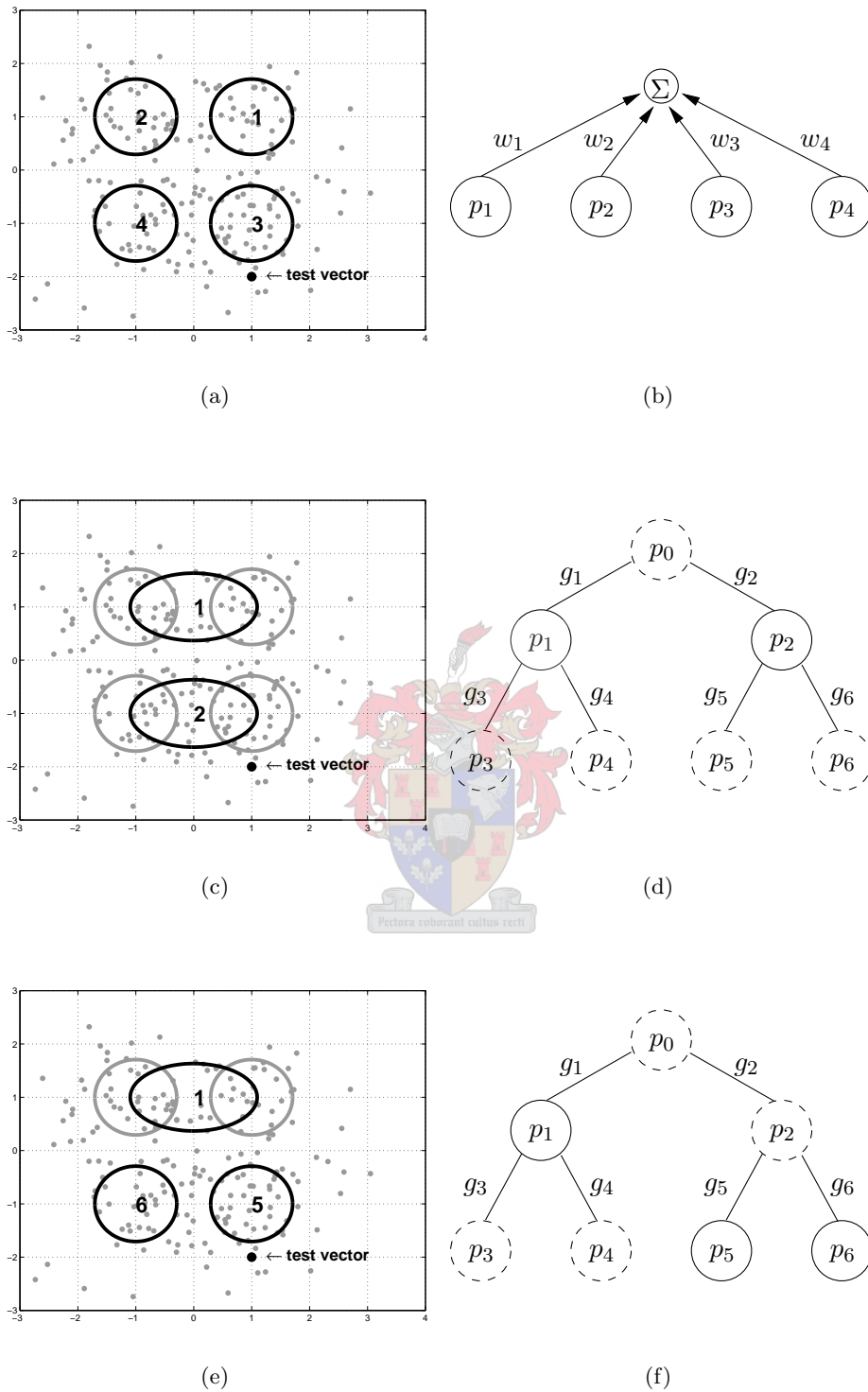


Figure 4.3: Very simple example of T-BAGMM evaluation. (a) and (b) show the regular GMM that will be approximated by a T-BAGMM. The grey dots in the scatter plot (a) are two-dimensional feature points. The circles in (a) indicate the standard deviation curves of each component PDF. (c) and (d) represent the comparison between the two nodes in the first tree level. Here node 2 has a higher likelihood contribution. (e) and (f) show the final selection of nodes for the resulting 3-component virtual GMM. The grey curves in (c) and (e) are shown for reference.

It is possible to define a threshold value against which node contributions can be compared. Instead of comparing two nodes against each other, they can be compared to this threshold. If a node's contribution is above the threshold, then the children of that node will be subjected to comparison at the next level. It is therefore possible to extract a more detailed virtual GMM from the tree. However, it is not very straightforward to define such a threshold. This is covered in Section 6.4.1.

The improvement in speed of the T-BAGMM over the regular GMM can be approximated by the ratio of Gaussian PDF evaluations:

$$\text{speed improvement factor} = \frac{\text{Gaussian PDF evaluations for GMM}}{\text{Gaussian PDF evaluations for T-BAGMM}}.$$

This ratio is dependant only on the number of mixture components. For 2048 components, the speed improvement is a factor of

$$\frac{2048}{2 \log_2 2048} = \frac{2048}{22} = 93.$$

Chapter 7 shows that such a remarkable increase in the practical speed measurement is indeed obtained, while suffering only a slight loss in accuracy.

4.3 Training

Maximum likelihood (ML) estimation and the expectation maximisation (EM) algorithm can also be applied to the training of a T-BAGMM. This is true for maximum *a posteriori* (MAP) adaptation as well. In order for model evaluation to be successful, every node in the tree must be re-estimated during training. Unfortunately, this eliminates any advantage in speed improvement obtained at evaluation time, because too much time must be spent on training.

Fortunately, the EM algorithm allows the use of the T-BAGMM evaluation algorithm to increase the speed of training as well. For each training feature vector, as with regular GMM training, all nodes (components) must be evaluated to calculate the responsibilities $c_{kt} = P(k|\mathbf{x}_t, \lambda^{[i]})$ that are used in the re-estimation formulas eq. (3.9), eq. (3.10) and eq. (3.11). Note that a node's responsibility is merely a normalised form of the node's likelihood contribution. With a T-BAGMM, only the nodes with the highest likelihood contributions (and hence, responsibilities) are taken into account, because very small responsibilities have barely any influence on the final model parameters. For a single training feature vector, a quick T-BAGMM search can identify those nodes to which the training vector will contribute significantly. During re-estimation,

only the vectors with significant contribution to a given node are included in the calculation of that node's parameters.

In order to perform this training method efficiently, the re-estimation formulas must be converted to be suitable for on-line estimation. This allows a vector to be incorporated into the calculation as soon as it is processed, instead of having to remember which vectors should be included for which nodes. The derivation of the on-line re-estimation formulas is covered in Section A.3. The basic procedure for training a T-BAGMM is shown in Algorithm 3.

-
-
1. Obtain training feature vector \mathbf{x}_t
 2. Perform T-BAGMM evaluation using \mathbf{x}_t to obtain likelihoods for the model and for the individual nodes
 3. For each node that was evaluated in step 2,
 - (a) calculate node responsibility from the likelihood obtained in step 2; where applicable, the approximate likelihood of a node is replaced by the higher-resolution likelihood obtained from its children
 - (b) apply \mathbf{x}_t and its responsibility to on-line re-estimation formulas for the specific node

Algorithm 3: Training of a T-BAGMM

As mentioned above, model parameter estimation is similar to that of regular GMMs. The exact same re-estimation formulas are used. Only the calculation of node responsibilities differ slightly from the calculation of mixture component responsibilities. The responsibility of a regular mixture component comes from eq. (3.8) and is written as

$$p(k|\mathbf{x}_t, \lambda) = \frac{w_k p(\mathbf{x}_t|\lambda_k)}{\sum_{i=1}^K w_i p(\mathbf{x}_t|\lambda_i)}. \quad (4.1)$$

On the other hand, the responsibility of a T-BAGMM node is written as

$$p(k|\mathbf{x}_t, \lambda) = \frac{\prod_{i=1}^{L_T(k)} g'_i p(\mathbf{x}_t|\lambda_k)}{g_1 p(\mathbf{x}_t|\lambda_1) + g_2 p(\mathbf{x}_t|\lambda_2)}. \quad (4.2)$$

The numerators of both equations are identical, except for the fact that the mixture weight of the T-BAGMM node is written as the product of level-based

node weights, or

$$w_k = \prod_{i=1}^{L_T(k)} g'_i.$$

This is explained in the previous section. The denominator of eq. (4.2) is in practical terms equivalent to that of eq. (4.1). For eq. (4.2), it is only written as the sum of the likelihood contributions for the two main subtrees. But, although it represents the model likelihood in both instances, the T-BAGMM version is an approximation, due to the evaluation algorithm.

Model initialisation is done with the binary-split algorithm, similar to what is described in Section 3.5.3. Part of the reason for choosing a binary tree structure for the T-BAGMM, is because it fits so well into the binary-split algorithm. When applied to the tree structure, each split is responsible for creating two new child nodes. The parent of these new nodes is, of course, the node that is being split. Each new child node is assigned an initial node weight $g_{k,left} = g_{k,right} = 0.5$ before ML estimation is performed. This satisfies the requirement that the node weights of two sibling nodes must sum to unity. It also does not make any assumptions about the actual distribution of weights. Some further issues regarding the implementation of the algorithm for the T-BAGMM structure is covered in Section 6.4.2. By using the binary-split algorithm for initialisation of both regular GMMs and T-BAGMMs, models can be trained that correspond very well between the two structures. This ensures that as little difference as possible exists for the purpose of comparing performance later.

To train models using MAP adaptation, a T-BAGMM UBM must be trained first. As for regular GMMs, this is done with the binary-split initialisation followed by ML estimation using the EM algorithm. MAP adaptation can then be applied to each node of the UBM individually.

4.4 Summary

In this chapter, the new T-BAGMM was introduced. This new model can replace the regular GMM in order to reduce execution time. The T-BAGMM can have any tree structure, but the development of a binary tree version was discussed. The evaluation algorithm was provided with regard to this binary tree structure. The model can be used to improve the speed of model training as well. A simple comparative summary of the main differences between the different GMM approaches is shown in Table 4.2.

	GMM	UBM-based component selection	T-BAGMM
component / node count	K eg. 2048	K eg. 2048	$2K - 1$ eg. 4095
evaluations per feature vector for one target and one impostor model	$2K$ eg. 4096	$K + C$ eg. 2048 + 5 = 2053	$2(2 \log_2 K)$ eg. $2 \times 22 = 44$
evaluations per vector for one target and 100 T-Norm impostors	$101K$ eg. 206 848	$K + 101C$ eg. 2048 + 505 = 2553	$101(2 \log_2 K)$ eg. $101 \times 22 = 2222$
kind of approximation	none	rough	smooth
valid PDF?	yes	no	yes
model structure	sum of weighted PDFs	sum of weighted PDFs	binary tree of PDFs with layers of resolution

Notes on the table entries:

- For the UBM-based selection examples, C is the number of selected components and was chosen to be 5. For the T-BAGMM selection examples, 2 components were chosen, but 10 extra lower resolution nodes are included in the virtual GMM.
- The examples show that for 2048 mixture components, the speed can be increased roughly by a factor of 93 with the T-BAGMM. The improvement by UBM-based selection is dependant on the number of T-Norm impostor models. It requires 119 impostors in order to reach the same improvement factor.

Table 4.2: Comparative summary between different model evaluation techniques

Chapter 5

The Speaker Verification System

5.1 Introduction

A general overview of modern speaker verification system construction was already given in Section 3.2. This chapter fills in some of the detail involved in performing speaker verification. Section 5.2 will describe the particulars of transforming raw speech signals to a form suitable for processing by a speaker verification system. Section 5.3 will then discuss how decisions are made. Lastly, Section 5.4 will cover the process distribution technique that was used for this research to reduce the execution time of full system performance evaluations.

5.2 Feature Extraction

Features are numerical measurements used in computations that try to discriminate between classes. Finding features for discriminating between speakers in the acoustic domain requires in-depth knowledge of speech signals. Currently, most researchers regard Mel-frequency cepstral coefficients (MFCCs) as the features that perform the best for acoustic speaker recognition. MFCCs, along with other signal-enhancing preprocessing techniques were used in this research. The descriptions that follow are summarised in Figure 5.1.

5.2.1 Front-end Signal Processing

In order to extract useful features, the signal must be transformed into a form suitable for processing. Signals in DSP are sequences of values, measured at discrete times. For the continuous-time speech signal $s(t)$, the discrete-time

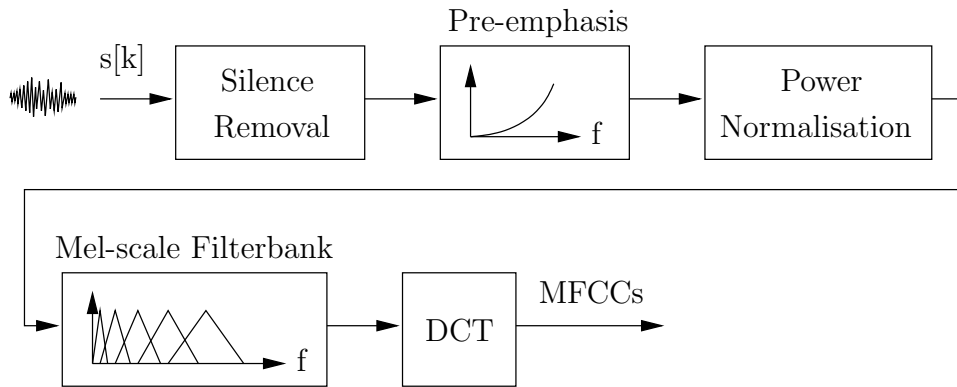


Figure 5.1: Block diagram of feature extraction.

signal is represented by $s[k] = s(kT_0)$, $k = 0, 1, 2, \dots$ where T_0 is the sampling period (the time interval between successive measurements). The discrete time k can be negative, but it is preferred to work with the starting-time at zero.

In a speech signal, the various types of information that are present can change rapidly through time. For this reason, the signal is divided into frames

$$f[n] = \{s[nW + k] : k = 0, \dots, W - 1\},$$

each consisting of W samples. By choosing W appropriately, it is then assumed that the signal is stationary (does not change) inside a frame. This is called the quasi-stationarity assumption. W must be large enough to include sufficient information, but it must also be small enough to ensure that the assumption of stationarity is reasonable. Frames normally overlap with their starting points following each other by L samples ($L < W$), because the signal does in fact change during the length of one frame. For speech signals, the frame width W is typically in the effective range of 15-30 ms and the following distance L is usually set to the equivalent of 10 ms [42].

Only the processing techniques that were used in this research are described here.

Silence Removal

Silences in speech convey no identity information in the acoustic domain. It may even have a detrimental effect on the results of some algorithms. A simple technique to eliminate silences from the signal was employed. The parts of the signal having very little power can be considered to be silences. These parts are

identified and removed as follows: The power of each frame is calculated by

$$F[n] = \frac{1}{W} \sum_{k=0}^{W-1} |s[nW + k]|^2.$$

The second percentile is chosen as the power floor ϕ_0 (the minimum power value for acceptable speech). This means that 2% of the frame-power values for the entire signal are below this floor value. This technique is robust against extreme power values (such as those nearing infinity) because they do not have as great an influence as they would when amplitude-based percentages were used. A small offset γ is added to the power floor and all the frames having less power than the value $(\phi_0 + \gamma)$ are temporarily removed from the signal. The second percentile of the remaining frame-power values is then chosen as a new power floor ϕ_1 . All frames in the original signal having less power than ϕ_1 are discarded.

Pre-emphasis

High-frequency components of the speech spectrum usually carry less energy than low-frequency components. This does not make the information carried in the high frequencies less important. However, speech modelling techniques tend to model the lower-powered high-frequency components badly. For this reason, a pre-emphasis filter

$$H(z) = 1 - 0.98z^{-1}$$

is applied to the signal in order to amplify the high-frequency components.

Power Normalisation

The signal power (loudness) may vary among different signals. It is preferable to normalise the power so that all speech segments can use the same scale and therefore be modelled on equal terms. A power normalisation technique is used that is robust against very large power values. The power is calculated for each frame (as shown above). The 75th percentile is then selected to become unity power. In other words, each sample in the signal is divided by the square root of the 75th percentile of the frame-power values.

5.2.2 Mel-Frequency Cepstral Coefficients

After processing the signal with the techniques described above, features can be extracted to characterise the speech. Cepstral coefficients based on the Mel frequency scale were used in this research.

Each feature vector is extracted from a frame. The frame is passed through a Hamming filter [16] and converted to the frequency domain using the discrete Fourier transform (DFT). Mel-scale frequency is related to linear frequency by the formula

$$\text{Mel}(f) = 1127 \ln \left(1 + \frac{f}{700} \right).$$

The frequency range in Mel-scale is divided into a number of equal-sized bands. In linear frequency, triangle filters are positioned so that the width of each filter is equal to two bands in the Mel scale. Two successive filters also overlap each other by one of these Mel-scale bands. The value for energy in each band after filtering is called a Mel filter bank coefficient.

Cepstral analysis involves working with the spectrum of the spectrum, hence the term cepstrum (the letters of the first syllable are in reversed order). More specifically, the inverse Fourier transform is applied to the log-spectrum of the signal. Mel filter bank coefficients m_j come directly from the signal spectrum. They can be transformed into Mel-frequency cepstral coefficients (MFCCs) c_i by using the discrete cosine transform (DCT), a simplified version of the DFT:

$$c_i = \sum_{j=1}^B m_j \cos \left(\frac{\pi i}{B} (j - 0.5) \right).$$

B is the number of Mel filter bank coefficients. The resulting MFCCs for each frame are grouped into a D -dimensional feature vector \mathbf{x} .

5.2.3 Channel Compensation

Various acoustic environments, microphones and communication channels can participate in the speaker verification process. Verification might for example be required by an automated telephone response system. In that case either land-line telephones or cellular telephones (indoor or outdoor) could be used. In this thesis the combination of these factors will simply be called the channel. Each channel has its own characteristics and distorts the speech signal accordingly. This presents a problem, because modelling techniques are often sensitive to such distortions.

To compensate for linear channel distortions, the popular technique of cepstral mean subtraction (CMS) is employed. It has to be assumed that the channel acts as a linear filter $h(t)$. Let $y(t)$ then be the corrupted signal, which is the result of convolution between the clean speech signal $s(t)$ and the filter:

$$y(t) = s(t) * h(t).$$

In the frequency domain, this translates to multiplication:

$$|Y(f)| = |S(f)| |H(f)|.$$

Applying the logarithm changes this to addition:

$$\log |Y(f)| = \log |S(f)| + \log |H(f)|.$$

Using the inverse Fourier transform, the cepstral coefficient vector corresponding to the n^{th} frame is extracted for the corrupted speech, the clean speech and the filter:

$$\mathbf{y}_n = \mathbf{s}_n + \mathbf{h}.$$

The maximum-likelihood mean estimate of the distorted signal cepstral coefficient vectors is

$$\boldsymbol{\mu}_y = \frac{1}{N} \sum_{n=1}^N \mathbf{y}_n = \frac{1}{N} \sum_{n=1}^N (\mathbf{s}_n + \mathbf{h}),$$

where there are N vectors. It is assumed that the filter is time invariant (does not change with time), resulting in

$$\boldsymbol{\mu}_y = \mathbf{h} + \frac{1}{N} \sum_{n=1}^N \mathbf{s}_n = \mathbf{h} + \boldsymbol{\mu}_s. \quad (5.1)$$

It is typical to assume that the speech signal is balanced with respect to the voiced, unvoiced and plosive sounds [23]. This would mean that the signal mean tends toward zero ($\boldsymbol{\mu}_s \rightarrow 0$) so that eq. (5.1) becomes $\boldsymbol{\mu}_y \approx \mathbf{h}$. The cepstral coefficient vector for the clean speech signal can therefore be obtained approximately by subtracting the cepstral mean

$$\mathbf{s}_n = \mathbf{y}_n - \boldsymbol{\mu}_y.$$

It must be noted that although CMS is quite effective, it provides only an approximation to the clean speech. It has also been shown that the assumed

balance in the clean speech signal varies between languages [24]. CMS also does not only remove channel effects, but anything that remains constant during the length of the signal is removed as well. This might even include information about the speaker.

5.2.4 Dynamic Features

Two different speakers can possibly produce the same sounds. But, depending on differences in social background and other similar factors, there might be differences in the transitions between sounds. This information can be incorporated as dynamic features to further aid in speaker discrimination. Two kinds of dynamic features can be employed: first derivatives (Δ) and second derivatives ($\Delta\Delta$).

The Δ -features are obtained by simply calculating the difference between two successive feature vectors. The resultant vector is appended to the second feature vector, making the procedure causal (i.e. only history is taken into account). The new feature vector then has double the dimension of the original. An alternative method uses five successive feature vectors. In this case, the Δ -features can be obtained by applying the following discrete differential filter:

$$\Delta\mathbf{x}_k = 0.125\mathbf{x}_{k+2} + 0.25\mathbf{x}_{k+1} - 0.25\mathbf{x}_{k-1} - 0.125\mathbf{x}_{k-2}. \quad (5.2)$$

It will be shown in Chapter 7 that eq. (5.2) gives better results than the two-vector difference.

The $\Delta\Delta$ -features are obtained by the exact same procedure, except that the difference between the Δ -features of successive vectors are calculated instead.

5.2.5 Feature Normalisation

Features often occupy a wide range of values. This can present a problem with regard to numerical precision in computers. To prevent such problems, feature values should be normalised. One simple method for normalising features, is to divide the value of each feature by some scaling factor:

$$\text{normalised feature value} = \frac{\text{feature value}}{\text{scaling factor}}.$$

For this thesis, a single scaling vector is calculated for a set of training feature vectors that were set aside for this purpose. All feature vectors are then scaled by this same vector before being processed further. The scaling factor on each dimension is set to be the standard deviation as calculated for

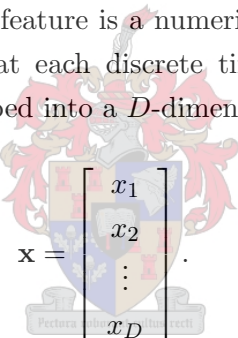
the corresponding dimension over the training vectors. This results in unity variance across each dimension for all normalised feature vectors in the training set. The training set should therefore be as representative as possible, so that the result will generalise to other feature vectors.

5.3 Evaluation

5.3.1 Bayesian Decision Theory

The problem of deciding which class is responsible for generating a test speech sample (utterance) can be approached with the statistical method of Bayesian decision theory [3]. This method assumes that the problem is formulated in probabilistic terms and that the relevant probabilities are known. In practical applications these probabilities are not known, but can be estimated as discussed in Chapter 3. This section will discuss the verification decision by first regarding the more general classification decision.

As stated in Section 5.2, a feature is a numerical measurement. Normally, a few features are measured at each discrete time instant. These features $\{x_d : d = 1, 2, \dots, D\}$ are grouped into a D -dimensional feature vector



$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}.$$

Let $\Omega \in \{\omega_j : j = 1, 2, \dots, C\}$ be the discrete random variable representing the class to which the feature vector \mathbf{x} belongs when there are C classes. Also, for the classification process, there are C hypotheses $\{H_j : j = 1, 2, \dots, C\}$. H_j makes the hypothesis that class ω_j is the true value of Ω (i.e. $\Omega = \omega_j$) for the vector \mathbf{x} . The measurement x_d is considered to be represented by a continuous random variable X_d whose distribution depends on the class ω_j . This can be expressed by the class-conditional probability density function (PDF)¹ $p(x_d|\Omega = \omega_j) = p(x_d|H_j)$. When determining the value of this PDF for all the features in the vector \mathbf{x} , a more compact notation is used to represent the joint probability density:

$$p(x_1, x_2, \dots, x_D|H_j) = p(\mathbf{x}|H_j).$$

¹The strict notation for the PDF is $p_{X_d}(x_d|\Omega = \omega_j)$, but the shorter version will be used when no ambiguity exists.

It is almost always possible to determine the *a priori* (prior) probability $P(H_j)$ of encountering each class. With this information, the joint probability density of finding both the class ω_j and the measurements \mathbf{x} can be written in two ways:

$$\begin{aligned} p(\mathbf{x}, H_j) &= P(H_j|\mathbf{x}) p(\mathbf{x}) \\ &= p(\mathbf{x}|H_j) P(H_j). \end{aligned}$$

This can be rearranged to give the Bayes formula:

$$P(H_j|\mathbf{x}) = \frac{p(\mathbf{x}|H_j) P(H_j)}{p(\mathbf{x})}, \quad (5.3)$$

where $p(\mathbf{x})$ is the marginal PDF:

$$p(\mathbf{x}) = \sum_{j=1}^C p(\mathbf{x}, H_j) = \sum_{j=1}^C p(\mathbf{x}|H_j) P(H_j).$$

Informally, eq. (5.3) has the form

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}.$$

From this it is possible to determine the *a posteriori* (posterior) probability that $\Omega = \omega_j$ given the feature measurements \mathbf{x} . In other words, the prior probability is combined with measured information to calculate the probability that the class ω_j is encountered. The likelihood factor is used as an indication that ω_j is more likely to be the true value of Ω if $p(\mathbf{x}|H_j)$ is large. The evidence factor $p(\mathbf{x})$ can be seen as a scaling factor to ensure that the posterior probabilities across all the classes sum to one.

For the verification task, it must be determined whether the claimed identity is present in a test utterance. Let there be two actions α_0 and α_1 , corresponding to accepting and rejecting the claim respectively. Also, let there be loss factors β_{ij} that represent the penalty for taking action α_i when the true identity is ω_j . When taking action α_i for an observed feature vector \mathbf{x} , the expected loss (or risk) is

$$R(\alpha_i|\mathbf{x}) = \sum_{j=1}^C \beta_{ij} P(H_j|\mathbf{x}), \quad (5.4)$$

because $P(H_j|\mathbf{x})$ is the probability that the true identity is ω_j .

Only two identities are used in speaker verification, namely the target ω_0 and all possible impostors ω_1 . Therefore, eq. (5.4) can be written more specifically:

$$\begin{aligned} R(\alpha_0|\mathbf{x}) &= \beta_{00}P(H_0|\mathbf{x}) + \beta_{01}P(H_1|\mathbf{x}) \\ R(\alpha_1|\mathbf{x}) &= \beta_{10}P(H_0|\mathbf{x}) + \beta_{11}P(H_1|\mathbf{x}). \end{aligned}$$

The best action to choose is the one with the minimum risk. Thus, the claim is accepted (action α_0) if

$$\begin{aligned} R(\alpha_1|\mathbf{x}) &> R(\alpha_0|\mathbf{x}) \\ \beta_{10}P(H_0|\mathbf{x}) + \beta_{11}P(H_1|\mathbf{x}) &> \beta_{00}P(H_0|\mathbf{x}) + \beta_{01}P(H_1|\mathbf{x}) \\ (\beta_{10} - \beta_{00})P(H_0|\mathbf{x}) &> (\beta_{01} - \beta_{11})P(H_1|\mathbf{x}) \end{aligned}$$

and otherwise rejected. Using the Bayes formula, an equivalent rule decides to accept the claim if

$$(\beta_{10} - \beta_{00})p(\mathbf{x}|H_0)P(H_0) > (\beta_{01} - \beta_{11})p(\mathbf{x}|H_1)P(H_1).$$

The $p(\mathbf{x})$ factor cancels out and it is seen that the decision can be made with only the likelihoods and prior probabilities.

Normally, the cost of making an error is larger than the cost of being correct, causing the factors $(\beta_{10} - \beta_{00})$ and $(\beta_{01} - \beta_{11})$ to both be positive. The decision to accept the claim can then be made if

$$\frac{p(\mathbf{x}|H_0)}{p(\mathbf{x}|H_1)} > \frac{(\beta_{01} - \beta_{11})P(H_1)}{(\beta_{10} - \beta_{00})P(H_0)}. \quad (5.5)$$

The ratio on the left-hand side is called the likelihood ratio. The term on the right-hand side is interpreted as being a threshold value θ that can be determined independently of the measurements. Other researchers seem to also allow the acceptance of the claim when the risks of both actions are equal. This is also done in the rest of this thesis.

With single-speaker detection, it is assumed that an entire utterance contains the speech of only one person. The decision can therefore be based on the utterance $\mathbf{X}_T = \{\mathbf{x}_t : t = 1, 2, \dots, T\}$ instead of only a single feature vector. By using the joint probability densities of the individual feature vectors, the likelihood ratio test becomes

$$\frac{p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T|H_0)}{p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T|H_1)} = \frac{p(\mathbf{X}_T|H_0)}{p(\mathbf{X}_T|H_1)} \begin{cases} \geq \theta & \text{accept} \\ < \theta & \text{reject} \end{cases} \quad (5.6)$$

For computational purposes, it is often preferred to work with the log-likelihood ratio²

$$\Lambda(\mathbf{X}_T) = \log p(\mathbf{X}_T|H_0) - \log p(\mathbf{X}_T|H_1).$$

This log-likelihood ratio of the test utterance is called the score of that utterance. The score is compared to an appropriate threshold to make the decision. As mentioned in Section 3.2, the PDFs in the likelihood ratio are approximated by theoretical models, such as the GMM.

5.3.2 Score Normalisation (T-Norm)

For any given target speaker, a number of target and impostor trials can be performed. A target trial requires the verification system to make a decision for a test utterance that was spoken by the target speaker (the claimed identity). An impostor trial requires a decision for a test utterance that was not spoken by the target speaker. The trials performed for any given target speaker can be characterised by a target score distribution and an impostor score distribution. The target score distribution describes the possible scores that can be produced by target trials according to some PDF. The impostor score distribution does the same for impostor trials. It is usually assumed that these distributions are Gaussian.

Except for the identity of the speaker, each test utterance can differ from other test utterances in many respects. Differences include the various microphones and communication channels used for recording. This can cause differences in the score distributions for the different target speakers. These variations in the score distributions prevent the verifier from effectively comparing the output scores to a single threshold for all target speakers. If it is undesirable to determine a threshold for each target speaker independently, then every score must be normalised to a global scale.

Test normalisation (T-Norm) [28, 29] is a distribution scaling technique. It normalises output scores so that the impostor score distribution will have a zero mean and unit variance for any given test utterance. Because all impostor trial scores are not available when an output score must be normalised, a subset of impostor model scores is used to estimate the scaling parameters. More specifically, the test utterance of a particular trial is used to calculate scores (log-likelihood ratios) for N non-target speakers. The mean μ_I and standard deviation σ_I for these N scores are then used to transform the score s from the

²The natural logarithm (base- e) is typically applied when working with probability densities of the exponential family.

target speaker into a normalised score

$$s' = \frac{s - \mu_I}{\sigma_I}. \quad (5.7)$$

Section A.1 in Appendix A shows that the log-likelihoods can be used instead of the log-likelihood ratios to obtain the same normalised score. It is therefore unnecessary to compute any likelihood ratios while calculating the parameters used in T-Norm.

T-Norm is able to calculate the scaling parameters for each test utterance. This means that differences in the acoustic environment between trials will not play such a large role in degrading performance as is the case with some other score normalisation methods.

5.3.3 Threshold Selection

Practical verification systems require a threshold θ to make decisions with the likelihood ratio test. A single fixed threshold must be chosen before deployment, or it must be adaptable according to some rule while the system is in operation. Only single fixed thresholds are considered in this thesis.

During the operation of a verification system, two errors can occur. A false rejection (FR) or miss is the error made when the system fails to detect the claimed identity, even though the speech of that person is present in the test utterance. A false acceptance or false alarm (FA) is the error made if the system accepts the claim when the test utterance was spoken by an impostor.

The choice of an appropriate threshold for making decisions involves a trade-off between the two types of errors. In Section 5.3.1, the threshold was defined in terms of loss factors β_{ij} . By assigning values to these loss factors the tradeoff between the two error types can be adjusted. A more direct approach is used in the National Institute for Standards and Technology Speaker Recognition Evaluation (NIST SRE) [43]. By minimising the detection cost function (DCF)

$$C_{det}(\theta) = C_{FR} \times FRR(\theta) \times P(target) + C_{FA} \times FAR(\theta) \times (1 - P(target)) \quad (5.8)$$

the desired threshold can be found. C_{FR} and C_{FA} are the costs associated with encountering FRs and FAs respectively. The $FRR(\theta)$ and $FAR(\theta)$ are the rates or probabilities of FRs and FAs respectively for a given threshold. $P(target)$ is the prior probability of encountering a target trial (one where the target speaker is present in the test utterance). Because a test utterance can only come from a target speaker or an impostor speaker, the prior probability

assigned to encountering an impostor is $(1 - P(target))$.

The $FRR(\theta)$ and $FAR(\theta)$ have to be measured by running many trials using a development data set. It is possible that the testing conditions of the development data set are different from the conditions while the system is operational. This would mean that the DCF and therefore the optimal threshold might change. Some researchers [21] take account of an asymmetry around the minimum DCF point to employ a heuristic threshold selection procedure. A threshold with a slightly higher DCF value than the minimum is chosen in the area to the side of the minimum where the DCF slope is smaller. This prevents the cost from increasing rapidly when the DCF changes so that the selected threshold moves to the side with the steeper slope.

5.4 Distributed Processing for the Verification Task

The technique of distributed processing is used widely throughout all kinds of research fields where large numbers of identical computations are required. Distribution is a very appropriate proposition for applications using GMMs with large numbers of mixture components, as is typically the case for speaker verification. Each mixture component could potentially be evaluated on a separate processor and the results combined to form the final score. This idea can also be extended to the task of model training. This kind of distribution was not readily available for the software package that was used for the research of this thesis.

When working with a large population of models and many test trials, another form of distribution can be applied. A small set of processors can be used optimally by training a subset of models on each, or by running a subset of trials on each. This technique was implemented for the research of this thesis using Linux shell scripts. It was applied to the 2004 NIST Speaker Recognition Evaluation (SRE) [43], which is a good example of where this kind of distribution makes sense. Specific details of how this distribution method was implemented is provided in Section 6.6.

5.5 Summary

This chapter, in combination with Chapter 3, described in relative detail the overall construction and operation of a speaker verification system. MFCC features are extracted from the raw speech signal and processed for optimum aid in discrimination. Using statistical models that describe how features are generated by speakers, the likelihood ratio test, or a normalised version thereof, is can make the decision of whether or not the target speaker is responsible for creating the test utterance.



Chapter 6

Implementation Issues

6.1 Introduction

The previous chapters provided the theory necessary for understanding how to implement a speaker verification system. However, theory does not provide all the details needed for a working implementation. This chapter aims to discuss the issues surrounding the implementation of a practical, working speaker verification system that makes use of the Tree-based adaptive Gaussian mixture models (T-BAGMMs) for speaker models.

Firstly, Section 6.2 gives a discussion of how the research was approached. It outlines the steps that were taken for learning how to implement a fully working speaker verification system.

Practical speaker verification systems must process real-world data. Section 6.3 describes the kinds of data that are required as well as how the data are applied to the system. It gives an overview of the speech databases that were used for this thesis and also indicates how data were selected for training purposes.

The implementation for this thesis was made using the PatrecII software library and executables from the University of Stellenbosch DSP group. In the fulfilment of the research for this thesis, the T-BAGMM component had to be added to the library. The implementation of the relevant algorithms and related issues are discussed in Section 6.4.

For performance evaluation, a baseline system is needed against which alternative system configurations can be compared. Section 6.5 provides a description of the specific baseline configuration used in Chapter 7. It also gives parameter values and their justifications.

Finally, the specific implementation of process distribution that was used for this thesis is discussed in Section 6.6.

6.2 Methodology

Any research requires at least a general guideline to follow in order to produce useful results. This thesis required the mastering of different levels of skill and knowledge. The original goal was to construct a GMM-based speaker verification system with as high an accuracy as possible. This system would be used in conjunction with a non-acoustic speaker verification system. Firstly, a course on pattern recognition was followed in order to learn the underlying concepts and techniques that are used in speaker verification systems. A course on speech processing was also followed to gain insight and background knowledge about how speech is processed in digital form. An extensive study on available literature was also done in order to become familiar with the construction of current speaker verification systems. Special attention was given to the speaker modelling component.

The patrecII software library, which is being developed by the DSP group at the University of Stellenbosch, was chosen for the construction of the system. It was readily available and included all the important parts that are necessary to build a complete GMM-based speaker verification system. It was necessary to learn how to use the library and its accompanying tools in order to construct a fully working system. A first system was built using regular GMMs, where each speaker model was trained directly from training data. In other words, the speaker models were not adapted from a universal background model (UBM). It was therefore necessary to keep the number of mixture components below 64. This system used the NTIMIT¹ speech database for both training and testing. This is a fairly small database and not particularly suited for real-world text-independent applications. It contains telephone-quality speech of predefined as well as random speech, although all utterances are recited from predefined text.

During the execution of this research, the DSP group of the University of Stellenbosch participated in the 2004 National Institute for Standards and Technology (NIST) speaker recognition evaluation (SRE). This was done in collaboration with Spescom DataVoice (Pty) Ltd. As part of this research, a process distribution system was implemented and used in the NIST SRE. The same distribution system would then also be used for the research work of this thesis. Additionally, the NIST SRE provided an ideal platform for building a speaker verification system that could be tested on a large set of real-world natural speech data in international competitive conditions. After becoming familiar with the NIST SRE rules and data, a UBM/GMM-based system was

¹Web page (2005):

<http://www ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S2>

constructed. The speaker models with up-to 2048 diagonal covariance components were trained by mean-only MAP adaptation from the UBM. It was soon discovered that many days were required to complete the NIST SRE trials, even when using up-to 10 processors. This was not acceptable if many experimental investigations were to be performed. This led the research to the consideration of implementing a tree-based GMM for the obvious speed gains that it would provide. At this point, the T-BAGMM idea was born [40].

The algorithms for the T-BAGMM component had to be implemented and integrated into the existing patrecII framework. This integration presented quite a steep learning curve and therefore required much time. Initially, a software component for MAP adaptation of both the mean and covariance matrix was created as part of the learning process, but also for possible use during experimentation. Other parts of the patrecII system also had to be improved during the implementation process.

As mentioned, an implementation of process distribution, based on Linux shell scripting, was implemented as part of this thesis, and used successfully in the 2004 NIST SRE. After observing a few problems with the initial implementation of the scripts, alterations were made before applying it to the research of this thesis.

Finally, with a fully working system, the T-BAGMM was applied to the 2004 NIST SRE conditions. A considerable improvement in speed was observed, and it matched closely to the rough theoretical predictions of Section 4.2. It was decided to perform comparative experiments to characterise the speed gains with respect to the number of mixture components as well as a parameter called the node score beam width. This parameter can be used to adjust a speed-accuracy trade-off in the T-BAGMM (see Section 6.4.1). Afterwards, a number of other experiments were performed to show how the system performance is influenced by various techniques.

6.3 Working with data

6.3.1 Applying data to the system

There are two modes in which a speaker verification system can be operated. These modes are essentially the same, except for the manner in which data are applied. The first mode (practical mode), is the normal operation where the system is used by some real client. The speech of the client is presented to the system for either training or evaluation. Training speech is used to train (estimate the parameters of) a model for the client. For evaluation, the

trained model is used to determine whether the evaluation speech belongs to the corresponding client. The system must record the evaluation speech, perform the verification process and provide a decision in as short a time as possible. Note that for text-independent speaker verification, the system has no prior knowledge about the content of the evaluation speech.

The second mode, is the simulated operation (development mode) where the performance of the system must be determined. This method of operation is used by the research presented in this thesis. Here the normal operating conditions are simulated by applying a large set of pre-recorded data in the same manner as for normal operation.

Two data sets are used during the development of a practical system: training data and evaluation (testing) data. The same speakers are present in the speech of both sets. For strict text-independent speaker verification the speech content of the two sets must be unrelated (i.e. the evaluation speech must not be present in the data used to train a model).

During evaluation, the speech of some person is presented to the system. A claim is made to the identity of the person who produced this speech. The claimed identity is called the target. The system must then test whether the evaluation speech truly belongs to the target speaker. If this is in fact true, the test is called a target trial. If, on the other hand, the evaluation speech was from someone other than the target, it is called an impostor trial.

For each target, the verification system requires a model that was trained from corresponding speech in the training set. The system must be able to discriminate between speech from the target and any possible impostor. This requires one or more models that represent impostors for every target. Other target models can be used as impostors for any given target, but there can also be speech in the training set for many speakers other than the targets. This can be used to train impostor models that are independent of all target models and is, of course, much easier to manage. Speaker models can be adapted from a general speaker model (or UBM). The UBM is trained from the speech of many different speakers and therefore has an abundance of training data.

A large number of trials are performed with the speech from evaluation set in order to determine how accurately the system performs. It is normally possible to get measurements at different operating points (for different decision thresholds). This allows the developer to select the threshold that would provide the best error trade-off. Both the training and evaluation data are typically contained in a single database, called the development database. Once the system is completed and the decision threshold chosen, new target models can

be trained for clients and the system can be activated for use.

6.3.2 Speech databases

Various commercial speech databases (or corpora) have been compiled and are for sale to researchers and developers. The research of this thesis made use of two databases that were compiled by the Linguistic Data Consortium (LDC), hosted at the University of Pennsylvania². Both databases were obtained by the University of Stellenbosch during participation in the NIST SREs of 2000 and 2004. The Switchboard-2 database was used in the 2000 NIST SRE. The data used for the 2004 NIST SRE were collected for the LDC Mixer project. The latter has not been released to the public yet, but will be called the Mixer database in this thesis.

Switchboard-2 Phase III

The Switchboard-2 corpus consists of a few separate sub-corpora: Phase I, Phase II, Phase III. Only Phase III was used for the research of this thesis. This database contains speech from 292 male and 348 female (640 total) different people. The speech is recorded telephone conversations. Each conversation has a duration of 5 minutes and there are 2657 conversations. Both sides of each conversation are stored in a single 2-channel format file. All silence intervals have already been removed. The technical specifications of the Switchboard-2 Phase III³ corpus are shown in Table 6.1

LDC Catalog No.	LDC2002S06
Authors	David Graff, David Miller, Kevin Walker
ISBN	1-58563-222-8
Data Type	speech
Sample Rate	8000 Hz
Sampling Format	2 channel μ -law
Data Source	telephone
Application	speaker identification
Language	English
Distribution	20 compact disks (CDs)
Non-member Price	US\$4000

Table 6.1: Technical specifications of the Switchboard-2 Phase III speech corpus

²Their web site: <http://www ldc.upenn.edu/>

³Obtained from web page (2005):
<http://www ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002S06>

Mixer

The Mixer database is not an officially released database. The data were collected during the Mixer project by making use of LDCs "Fishboard" platform for automated call initiation⁴. The data consist of mostly English conversational telephone speech, but includes speech from other languages as well. Each side of a conversation is stored in a separate single-channel file. Each such conversation side contains the last five minutes of a six-minute conversation. There are also files that contain only excerpts of about 10 seconds of speech, as well as some containing excerpts of about 30 seconds of speech. In addition, there are files where the two sides of a conversation, minus the first minute, were summed together. No silence removal have been performed. The data includes a limited number of conversations where speech was recorded using non-telephone channels and a variety of microphone types. For the conversations that were recorded over a telephone channel, the type of transmission may be regular (land-line), cordless or cellular and the instrument used may be regular (hand-held), ear-bud, head-mounted or speaker-phone.

There are data for 3355 speakers in the database and each person participated in various conversations. The data are accompanied by errorful word transcriptions that were produced by an automatic speech recognition (ASR) system. Some technical specifications of the Mixer corpus are shown in Table 6.2

Data Type	speech
Sample Rate	8000 Hz
Sampling Format	1 channel μ -law
Data Source	telephone and non-telephone
Application	speaker identification
Language	English, Arabic, Mandarin, Russian, Spanish
Distribution	5 DVDs

Table 6.2: Technical specifications of the Mixer speech corpus

6.3.3 Selection of training and evaluation data

The Switchboard-2 Phase III corpus was used as the development database, but only for training the UBM and impostor models. Although this research is concerned with performance evaluation according to the 2004 NIST SRE, it is not necessary for the system to operate in the application mode where decisions must be made. It was therefore not necessary to train development target

⁴More information is available at the web page (2005):
<http://www ldc.upenn.edu/Projects/EARS/Fisher/Mixer/>

models and choose a decision threshold. The Mixer database is the evaluation database that is used to evaluate the system performance. It contains training and testing data according to the 2004 SRE conditions.

UBM training data

The UBM must model general speaker information. Ideally, it should include all the different sounds that any person can make. In other words, the UBM models a speaker-independent distribution of features [6]. The data used to train the UBM must therefore come from a wide variety of speakers and include a wide variety of words. Conversational speech seems to be well suited for this purpose.

This thesis uses a simple approach for training the UBM. All the UBM training data are pooled together. The pooled data are used to find maximum likelihood estimates for the UBM parameters with the EM algorithm. On the other hand, if it is known that models of female target speakers will only be compared to female test speech, then a UBM can alternatively be trained with speech data from only female voices. All female target and impostor speakers can then be adapted from this gender-dependant UBM. The same applies to different languages, dialects and types of communication channels.

However, in the case where there is no prior knowledge about the test speech samples, the UBM must be trained with a balance between different possible subpopulations. For example, there should be an equal amount of data for both male and female speech and likewise for different channels. If this balance is not achieved, then the UBM and adapted models might be biased and so reduce accuracies in certain situations. When male models are adapted from a predominantly female-based UBM, the system might be inclined to reject the corresponding male's test speech, even though it matches with the model.

There is no theoretical principle according to which the correct amount of speech data or the correct number of speakers for the UBM training data can be determined. The main concern is to have as much variety in speakers as possible, but the data of each speaker should cover most of the speaker's voice characteristics. This thesis uses 60 seconds of speech per person with a minimum of 200 speakers in the UBM training data. From studying literature [6], this seemed to be sufficient, although thorough tests could not be performed to verify it.

An equal number of male and female speakers were selected for the UBM training data. It was impossible to regard any other possible subpopulations, because no training data were available for that purpose. The UBM training data were taken from the Switchboard-2 Phase III corpus. All of the speech in this corpus was recorded over a regular (land-line) telephone channel. This would most likely cause a bias towards this kind of channel, but unfortunately no training data were available for other channels. The Switchboard-2 Phase III corpus is otherwise a good choice, because it consists of conversational speech, which should contain enough variety in speech sounds.

T-Norm impostor data

T-Norm impostor models are used to normalise the target score. These models serve as examples of any speech other than that of the target and must be largely independent of the target models. A set of 100 speakers from the Switchboard-2 Phase III corpus were selected as impostors. To prevent a bias in the system, as with the UBM, an equal number of male and female speakers were selected for impostors. These speakers are a subset of the speakers that were chosen to train the UBM. Of course, this means that the target models do share some information with the impostor models. But, the amount of speech per person that was used for the UBM was little in comparison to what was used to adapt the speaker models. The fact that the UBM represents the general speaker also helps to eliminate much of the remaining dependence, because many other speakers are also present in the UBM. However, it is unknown whether total independence between target and impostor models would provide better verification performance. Such an experimental investigation was planned for this research, but time constraints did not allow it.

For each of the impostor speakers, a GMM or T-BAGMM model was trained using 3.6 minutes of speech data. The amount of training data per impostor model was chosen according to the following criteria:

- enough training data are needed to cover most of the acoustic properties for the person's voice,
- it should be easy to implement using the existing software.

The second criterion was the most defining, because the software only allowed training data set sizes that were multiples of 3.6 minutes. It was decided that 3 minutes of speech should be enough to cover a very substantial range of the speaker's acoustic properties, but also that 7.2 minutes might be unnecessarily

long. This thesis does not cover the evaluation component of the speaker verification system in enough detail to perform experiments that would determine the optimum amount of training data for impostor models. For the same reason, in addition to time limitations, it also could not be expected to produce an implementation for the software that can provide a finer granularity for the training data set sizes.

Evaluation data

Because this research was mainly performed according to the 2004 NIST SRE conditions, the evaluation data were selected from the Mixer database. For each test condition, NIST provided indices to define the training speech for target models and the test speech for trials. More information will be given in Section 7.4.

6.4 T-BAGMM algorithms

6.4.1 Evaluation

Implementing the basic algorithm for evaluating a T-BAGMM (see Section 4.2) is a relatively simple matter. However, for this thesis, it was decided to include a parameter with which the speed of evaluation can be adjusted. This parameter determines the number of nodes that are evaluated based on how much the nodes contribute to the model likelihood. The parameter is called the node score beam width δ . For each pair of sibling nodes, this beam width is used to calculate a node score threshold ϕ . By defining some maximum node score s_{max} , the threshold is calculated as

$$\phi = s_{max} - \delta.$$

The node score s_n (likelihood contribution) of each of the two sibling nodes is compared to this threshold to determine whether the subtree of the node will be evaluated in more detail. In other words, a node's subtree is evaluated if

$$s_n > \phi.$$

A few implementation iterations were made to determine an appropriate definition for s_{max} :

1. For the first attempt, the highest score of two sibling nodes was chosen, giving

$$s_{max} = \max(s_{n,left}, s_{n,right}).$$

But, this definition has a fundamental problem. It considers only two nodes. As the traversal of the tree progresses further away from the root node, some nodes have extremely low scores. Although these nodes should not be included in the evaluation, this definition for s_{max} does not provide for such exclusion when the node score beam width is large. It is desirable to compare the node scores with those in the higher-scoring subtree(s). But, this is not possible with this definition. This problem was observed when nodes with extremely low responsibilities were included for re-estimation during training.

2. In order to prevent the evaluation of nodes that have very low scores, it was decided to use a global s_{max} for the entire tree. As the tree is traversed, the value of s_{max} is updated when a higher node score is encountered. The problem with this method, is that comparisons between nodes of different tree levels are not appropriate (fair). It is incorrect to compare node scores of a higher resolution with a threshold calculated for nodes of lower resolution, and *vice versa*. In practice, this was very clear during training. At the time, the second tree traversal of the tree (that is responsible for re-estimation) repeated comparisons with the updated node score threshold. The result was that none of the more approximate nodes had a high enough score to enable further traversal. This prevented the UBM from being trained properly.
3. Finally, a slightly modified version of the first method was used. Again, an s_{max} was chosen for each pair of sibling nodes. However, the choice was the highest of either one of the sibling node scores or the s_{max} that was chosen for their parent. This ensures that the evaluation of nodes in the lower-scoring subtree stops when the node scores become too small. It also prevents the comparison of nodes with very different resolutions. A resolution difference of one tree level seemed close enough to be allowed. For training, it was also decided to "remember" the threshold that was used in the first tree traversal (that is responsible for determining node responsibilities). The same threshold values can then be used in the second tree traversal. But, it was discovered that evaluation speed

was far higher than predicted by approximate theoretical calculations. It was found that the tree was not always traversed up-to the leaf nodes. As should have been expected, this happened because node scores of different tree levels were compared to each other. In some cases, both sibling node scores could be lower than that of their parent, even if they are in the highest-scoring path. This is a problem, because the purpose of using a tree, is to find the highest-scoring leaf nodes. The problem was alleviated by selecting s_{max} to be the highest of the two sibling node scores when they both were lower than the s_{max} of their parent, but only if they were in the highest-scoring search path. By forcing traversal up-to the leaf nodes in this manner, more realistic speed measurements were obtained as well as better system performance.

The final algorithm for evaluating a T-BAGMM is shown in Algorithm 4.

6.4.2 Training

It was already seen in Section 4.3 how training speed can also be improved by incorporating the T-BAGMM evaluation algorithm into the estimation process. Similarly, the speed of training can also be adjusted with the same node score threshold that was introduced in the previous section. This will allow more nodes to be re-estimated and should therefore improve the quality of the resulting models. The training algorithm that was given in Section 4.3 remains the same, because the evaluation algorithm selects the nodes that will be re-estimated.

The binary-split model initialisation algorithm had to be adapted for application to the T-BAGMM. It was decided to keep a vector table with references to all of the current leaf nodes for any given iteration. This vector table initially contains only a reference to the root node. When the root node is split, two child nodes are created and linked to the root node. The root node's reference is removed from the vector table and replaced by references to each of the two new leaf nodes. In this way, the vector table is used to locate the leaf nodes that should be split. The rest of the algorithm remains the same and this process continues until the desired number of leaf nodes have been produced.

Finally, provision was made so that the models could have the potential to grow as large as the data would allow. As the number of leaf nodes increase through the binary-split algorithm, some new nodes may end up having very small responsibilities for all feature vectors in the training set. This in turn would cause the parameters of those nodes to be badly estimated. In such cases, both that node and its sibling node are removed from the tree and the

algorithm is prevented from splitting the parent again in the future. If the target number of leaf nodes is chosen too large for a particular data set, then this mechanism would provide a resulting model with the maximum complexity allowed by the data.

-
-
1. Obtain test feature vector \mathbf{x}_t
 2. If node is a leaf node, then return the likelihood of the node $p(\mathbf{x}_t|\lambda_l)$ using the node PDF; otherwise continue with step 3
 3. Calculate the approximate likelihood of the left subtree $p(\mathbf{x}_t|\lambda_{k,left})$ from the approximate PDF in the left child node; multiply mixture weight of left subtree $w_{k,left} = g_{k,left} \prod_{i=1}^{L_T(k)} g'_i$ with this likelihood to obtain left node score $s_{n,left}$
 4. Calculate the approximate likelihood of the right subtree $p(\mathbf{x}_t|\lambda_{k,right})$ from the approximate PDF in the right child node; multiply mixture weight of right subtree $w_{k,right} = g_{k,right} \prod_{i=1}^{L_T(k)} g'_i$ with this likelihood to obtain right node score $s_{n,right}$
 5. Choose $s_{max} = \max(s_{n,left}, s_{n,right}, s_{max,parent})$
 6. If $(s_{max} = s_{max,parent})$ and this node is in the path where all nodes have maximum scores, then choose $s_{max} = \max(s_{n,left}, s_{n,right})$
 7. Calculate $\phi = s_{max} - \delta$
 8. If left score $\geq \phi$ threshold and left child is not a leaf node, then obtain detailed likelihood of left subtree $p(\mathbf{x}_t|\lambda_{k,left})$ by repeating from step 2 for left child node
 9. If right score $\geq \phi$ and right child is not a leaf node, then obtain detailed likelihood of right subtree $p(\mathbf{x}_t|\lambda_{k,right})$ by repeating from step 2 for right child node
 10. return $(g_{k,left}p(\mathbf{x}_t|\lambda_{k,left}) + g_{k,right}p(\mathbf{x}_t|\lambda_{k,right}))$ as the detailed likelihood

Notes on the algorithm:

- The function $L_T(k)$ indicates the tree level in which node k is located.

Algorithm 4: Recursive Evaluation of a T-BAGMM, making use of a node score threshold

6.5 The baseline system configuration

6.5.1 Introduction

For research purposes, it is important to have a baseline system configuration against which alternative system configurations can be compared. It is then possible to see whether alterations improve or degrade the system performance.

Two baseline systems are used for this thesis. Both are identical, except that one uses GMM-based speaker models and the other uses T-BAGMM-based speaker models. The GMM-based configuration is used merely for performance comparison between the GMM and the new T-BAGMM. All of the other investigations compare performance against the T-BAGMM baseline system. Because the GMM-based system requires a great amount of time for a full system evaluation, it was decided to keep the baseline system complexity at a minimum. This section gives a detailed description of the baseline configuration used for this thesis.

6.5.2 Front-end processing

The different channels (or conversation sides) in the Switchboard-2 conversations had to be separated before actual processing could be performed. In the Mixer database, the conversation sides are already separated. Although silences are already removed from the speech in the Switchboard-2 database, the same front-end processing was applied to both the Switchboard-2 data and the Mixer data. If the silence removal component had any effect on the Switchboard-2 data, it would be minimal.

Frames with a length of 20 ms was extracted at a rate of 10 ms per frame. These values were chosen because nearly all other researchers use the same (or nearly same) values. For every frame, silence removal, power normalisation and feature extraction was performed. Following in the choice of other researchers, 12 MFCCs were extracted per frame using 22 triangular filters in the frequency range of 300 Hz to 3300 Hz (the telephone band). These MFCCs were collected into 12-dimensional feature vectors. Normally, 13 coefficients are calculated and the zeroth coefficient (energy) discarded, but here the zeroth coefficient is included in the set of 12. This was done because the actual implementation of the feature extraction software was not understood properly at the time when the baseline system was configured. The effect of this is investigated in Chapter 7, though.

No cepstral mean subtraction (CMS) or dynamic feature calculation was performed for the baseline system. A scaling feature normaliser was trained using speech of the 200 speakers in the UBM data set. This was used to normalise all feature vectors.

6.5.3 UBM

The UBM is a regular GMM with 2048 components using diagonal covariance matrices. This is considered as the best choice by other researchers [2, 6]. Not only is diagonal covariance GMMs computationally (and storage-wise) more efficient than full covariance GMMs, but it can model data equally as well if more mixture components are used. Furthermore, the calculation of MFCCs tend to decorrelate the data, which means that diagonal covariance can be used without much degradation of performance.

The UBM is trained with 60 s of speech from each of 200 speakers in the Switchboard-2 Phase III database. This set of speakers consists of 100 male and 100 female speakers so that the UBM will be gender-balanced (refer to Section 6.3.3). No training data were available to ensure balance for other subpopulations. The selection of UBM speakers is essentially random. The list of speakers was scanned and the first 100 of each gender, of those whom had enough speech data, were chosen. The list of speakers is in no apparent order.

The UBM is initialised with the binary-split algorithm followed by 9 iterations of maximum likelihood estimation with the expectation-maximisation (EM) algorithm. Only 5 EM iterations should be necessary for sufficient convergence [6], but another 4 was added for toleration.

6.5.4 Speaker models

Each speaker model was created with mean-only maximum *a posteriori* (MAP) adaptation from the UBM. A relevance factor of 16 was used, because other researchers have determined that this value is a good choice [6]. A maximum of 9 EM iterations were performed. Training could stop earlier when the improvement in the log-likelihood value for the entire training data set of the corresponding person fell below 5×10^{-4} . Such a low improvement value indicates convergence of the EM algorithm. The models of the baseline T-BAGMM system were trained with a training node score beam width of $\delta_{train} = 3$ (logarithmic scale). This value was empirically determined as a good trade-off between speed and model quality.

Target models were trained according to the conditions of the 2004 NIST SRE. These conditions are presented later in Section 7.4. T-Norm impostor models were trained from the same speaker data as the UBM, but with 3.6 minutes of speech per speaker.

6.5.5 Evaluation

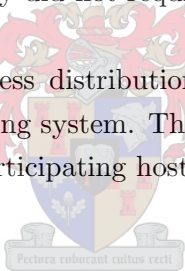
Test normalisation (T-Norm) was performed on the test utterance log-likelihood scores using 20 impostor models. These impostor models were trained as discussed in the previous section.

6.6 Process distribution

Process distribution was needed to complete the 2004 NIST SRE before the deadline. Many fully functional software systems already exist to perform this function. However, at the time, it was decided that it would take too long to learn the setup and usage procedures of available software. Also, the kind of distribution that was necessary did not require such a complicated software implementation.

Consequently, a simple process distribution was implemented with Bash shell scripts on the Linux operating system. The Secure shell (SSH) software is used for communication with participating hosts.

6.6.1 First generation



Two main scripts are responsible for handling the distribution: a manager and a performer. The manager script implements a simple loop to send process jobs to each participating host. The manager merely executes the performer script on the remote machine by using the SSH software. The performer script must then use the information that was passed to it, to locate and process the correct set of data. This requires that all the data must be available on each participating host.

The first generation of scripts used a rudimentary file-based lock mechanism to prevent more than one job to occupy a single processor simultaneously. Such a locking mechanism was required so that the manager could not send more than one job to any given processor.

6.6.2 Second generation

After the completion of the 2004 NIST SRE, the distribution scripts were altered to improve its performance. The new scripts were used for the research

of this thesis. The most important change in the second generation of the distribution scripts, was the replacement of the file-based lock mechanism. As trials were executed, it was observed that the file locks did not function properly. It sometimes happened that a processor would receive a second job before the lock file for the first job was created. The problem arose mainly because accommodation was made for machines with multiple processors.

To circumvent the problem, a network pipe was used. Each remote machine runs a simple server-like script that listens for network connections (by means of the netcat utility) only when a processor has completed its last assigned job. The intention was to allow direct connections from the host running the distribution loop (manager), without the use of SSH. But experimentation indicated that this might not be possible. Instead, the performer tries to connect to the server script. A successful connection can only be made if the server is listening and that can only happen when a processor is available. All functionality is now transferred from the performer script to the server script.

Along with the second generation of distribution scripts, the shell file system (SHFS) was applied so that each participating host could access the data from a single machine. The shell file system obtains access to remote files through the SSH software. No significant performance loss was experienced with this method of file access.

6.7 Summary

This chapter discussed the practical issues regarding the implementation of a fully working speaker verification system that use either GMMs or T-BAGMMs for speaker models. It was seen how the research was approached by first gaining familiarity with the subject matter and relevant software. The Switchboard-2 Phase III and Mixer databases were introduced and it was shown how data were selected for application to the research.

The discussion surrounding the implementation of the T-BAGMM algorithms introduced a parameter, called the node score beam width δ , with which the speed-accuracy trade-off can be adjusted. This parameter requires a maximum node score s_{max} for operation and the search for a proper definition of the latter was described.

A description of the baseline system configuration was given with regards to all the participating components. Specifics were given about the choice of parameter values for front-end processing. It was discussed why diagonal covariance matrices are best suited for GMMs in speaker verification.

Finally, it was described how process distribution was implemented using two Linux shell scripts: a manager and a performer. The manager script is responsible for executing the performer script on each participating host and the performer must locate the data and execute the correct process.

The next chapter will show the results of all the experiments that were performed for this theses, based on the 2004 NIST SRE conditions.



Chapter 7

Experimental Investigation

7.1 Introduction

It is important to determine whether theoretical expectations are matched in practice. This chapter covers the experimental investigation of a few of the important parts in the speaker verification system. It especially aims to show how the new T-BAGMM performs against the GMM with regard to speed and accuracy. Investigations are also made to determine how the other parts influence the system performance. Section 7.2 explains the technique for visually interpreting the results obtained from verification trials in a sensible way. Section 7.3 describes the formal method for determining whether seemingly improved results have any true significance. Section 7.4 gives an overview of how evaluation was performed for the rest of the chapter.

7.2 The Detection Error Trade-off (DET) Curve

Two errors can occur during verification. A false rejection (FR) or miss is the error made when the system fails to detect the claimed identity, even though the speech of that person is present in the test utterance. A false acceptance or false alarm (FA) is the error made if the system accepts the claim when the test utterance was spoken by an impostor.

The choice of an appropriate threshold for making decisions involves a trade-off between the two types of errors. For this reason, the performance of a system can not be represented by a single performance number. A more complete characterisation is given by some performance curve such as the detection error trade-off (DET) curve [30].

The DET curve indicates the false alarm rate or probability (FAR) on the horizontal axis and the false rejection rate (FRR) on the vertical axis. A normal

deviate scale is used on both axes to better distinguish between different well-performing systems. This scale tends to produce near-linear curves.

To generate values for the DET curve, a large number of trials must be executed to produce output scores. These scores are then compared to various thresholds and the FAR and FRR are calculated for each threshold. These rates (or probabilities) are usually calculated and shown as percentages. Each of these percentages is viewed as a percentage of area beneath the normal PDF (i.e. zero mean and unit variance), with the area filling up from $-\infty$. The standard units (multiple of the standard deviation σ) corresponding to these areas are then plotted on the DET curve. If, for example, $FAR = 84\%$ and $FRR = 16\%$, then the standard unit corresponding to FAR is $+1$ and the standard unit for FRR is -1 . This would give the point $(x, y) = (+1, -1)$ on the DET curve.

The DET curve was used to evaluate the system performance for all the experimental investigations that are presented in Sections 7.5 through 7.11. It must be noted that the DET curve is not restricted to the application of speaker verification systems, but can be used for any type of detection system.

7.3 Significance Testing (McNemar's Test)

When comparing the performance measurements between two system configurations, it might seem as though one system performs better than the other. In order to determine whether the difference is not simply due to chance effects, it is necessary to subject the results to a statistical significance test [12]. One such test, called McNemar's test [31], is appropriate for verification tasks. It is important to note that a significance test can not determine the cause of a difference, but only whether the difference is real.

Let there be a comparison between two systems A_1 and A_2 . These systems are subjected to the same N trials. The performance of the two systems can be represented by the following random variables:

N_{00} = Number of trials that A_1 verifies correctly and A_2 verifies correctly

N_{01} = Number of trials that A_1 verifies correctly and A_2 verifies incorrectly

N_{10} = Number of trials that A_1 verifies incorrectly and A_2 verifies correctly

N_{11} = Number of trials that A_1 verifies incorrectly and A_2 verifies incorrectly.

Although these variables are random, they satisfy the constraint $N = N_{00} + N_{01} + N_{10} + N_{11}$. The following probabilities are defined:

- q_{00} =probability that A_1 verifies correctly and A_2 verifies correctly
- q_{01} =probability that A_1 verifies correctly and A_2 verifies incorrectly
- q_{10} =probability that A_1 verifies incorrectly and A_2 verifies correctly
- q_{11} =probability that A_1 verifies incorrectly and A_2 verifies incorrectly.

The probability that system A_1 makes an error is $p_1 = q_{10} + q_{11}$ and the probability that system A_2 makes an error is $p_2 = q_{01} + q_{11}$.

The null hypothesis H_0 states that there is no difference between the two systems. In other words, they have the same chances for making an error:

$$\begin{aligned} p_1 &= p_2 \\ q_{10} + q_{11} &= q_{01} + q_{11} \\ q_{10} &= q_{01}. \end{aligned}$$

It is the aim of the significance test to determine whether the null hypothesis is true. Define $q = q_{10}/(q_{10} + q_{01})$ to be the probability that A_1 makes an error when only one of the systems makes an error. The null hypothesis then states that $q = \frac{1}{2}$, or that it is equally likely for either system to make an error when only one system makes an error.

Let $K = N_{01} + N_{10}$ be the random variable representing the number of trials where only one system makes an error. When N trials are executed, a possible observation can be made that $K = k$ and also that $N_{10} = N_{10}$. These trials can be seen as being Bernoulli trials. The probability that A_1 makes an error while A_2 does not, for N_{10} times out of k trials, is

$$P(N_{10} = N_{10}) = \binom{k}{N_{10}} q^{N_{10}} (1 - q)^{k - N_{10}}.$$

The random variable N_{10} therefore has the Binomial distribution

$$p(n_{10}|k, q) = \sum_{m=0}^k \binom{k}{m} q^m (1 - q)^{k-m} \delta(n_{10} - m).$$

Here, n_{10} is a regular variable representing the possible realisations of the random variable N_{10} . Under the null hypothesis H_0 , this PDF becomes

$$\begin{aligned} p\left(n_{10}|k, \frac{1}{2}\right) &= \sum_{m=0}^k \binom{k}{m} \left(\frac{1}{2}\right)^m \left(1 - \frac{1}{2}\right)^{k-m} \delta(n_{10} - m) \\ &= \sum_{m=0}^k \binom{k}{m} \left(\frac{1}{2}\right)^k \delta(n_{10} - m), \end{aligned}$$

and the expectation holds that $E[N_{10}] = \frac{k}{2}$ (i.e. it is expected that $N_{10} = N_{01}$, because $k = N_{10} + N_{01}$). If the actual observation N_{10} is very close to $\frac{k}{2}$ (i.e. $|N_{10} - \frac{k}{2}| \rightarrow 0$), then H_0 is more likely to be true. Similarly, if $|N_{10} - \frac{k}{2}| \gg 0$, then H_0 is less likely to be true. A two-tailed test is applied to H_0 because it does not matter to which side of $\frac{k}{2}$ the observation N_{10} lies. It is only important to consider how far away the observation is. The probability of H_0 being true can therefore be computed as

$$\begin{aligned} P(H_0) &= 2 \sum_{m=0}^{N_{10}} \binom{k}{m} \left(\frac{1}{2}\right)^k && \text{when } N_{10} < \frac{k}{2} \\ &= 2 \sum_{m=N_{10}}^k \binom{k}{m} \left(\frac{1}{2}\right)^k && \text{when } N_{10} > \frac{k}{2} \\ &= 1.0 && \text{when } N_{10} = \frac{k}{2}. \end{aligned}$$

When $P(H_0) < \alpha$ for some significance level α , the null hypothesis (that there is no difference between the systems) is rejected. Typical significance levels are 0.001, 0.01, 0.05 or 0.1.

For speaker verification systems, significance tests must be performed with regard to the system performance. The system performance is characterised by the FRR and FAR, which are calculated for many different operating points (choices of thresholds). It is therefore sensible (or fair) to compare two systems at operating points where the ratio of their FRRs to FARs are equal [44].

The FRR and FAR values are calculated for different thresholds. Each system uses its own set of thresholds, and the number of thresholds may differ among the various systems. Suppose that n is the index of a given threshold value in the set of thresholds for system A_1 . Similarly, m is the index for a given threshold value in the set of thresholds for system A_2 . Now, let $r_1(n) = FRR_1(n)/FAR_1(n)$ be the error rate ratio for system A_1 using the n^{th} threshold. Also, let $r_2(m) = FRR_2(m)/FAR_2(m)$ be the error rate ratio for system A_2 using the m^{th} threshold. The two systems should be compared

with McNemar's significance test where $r_1(n) = r_2(m)$ for all appropriate m and n . It is most likely that there are no thresholds for which such matching error rate ratios can be found in both systems.

To alleviate the problem, the closest matches are compared. The procedure used to find these closest matches, iterates through the set of thresholds for one of the systems. For the sake of explanation, the thresholds of system A_1 are chosen. For the n^{th} threshold $\theta_1(n)$, the error rate ratio $r_1(n)$ is determined for system A_1 . The closest matching $r_2(m) \approx r_1(n)$ must then be found. Lastly, the error counts N_{10} and N_{01} used in the significance test is computed for A_1 at threshold $\theta_1(n)$ and for A_2 at threshold $\theta_2(m)$. From these counts, $P(H_0)$ can be calculated to determine the level at which differences in the systems are significant. This procedure is performed for every threshold $\theta_1(n)$ (i.e. for all appropriate n). With this method, the results of the significance tests can be combined with the DET curve for visual presentation.

7.4 General System Evaluation: 2004 NIST SRE

7.4.1 Overview

All of the important investigations for this research were performed according to conditions that were defined for the 2004 National Institute of Standards and Technology (NIST) speaker recognition evaluation (SRE) [43]. The data used in the 2004 SRE were collected for the Mixer project by the Linguistic Data Consortium (LDC) and is covered by Section 6.3.

Many test conditions were defined for the SRE, but only one particular test condition was chosen for this research. It involved training target models with 8 (eight) conversation sides and performing a trial with 1 (one) full conversation side as test speech. Each conversation side contains the speech of one side of a 6 (six) minute conversation from which the first minute was removed. In other words, target models were trained with 40 minutes of conversational data from (ideally) one person.

The signal includes both speech and silences, and sometimes background noise as well. The test utterance consists of 5 minutes of speech with the same characteristics as the training speech. Because silences often make a substantial part of the signal, the duration of actual speech can be approximated to be about two thirds of the quoted times.

Some conversation sides are dominated by silences and some contain no speech at all. Some even have nothing but telephone control tones. These signals were included in the database, because no (or very little) auditing was

done on the data prior to its release to the sites that participated in the SRE. These signals obviously cause degraded results, but fortunately they are rare and probably do not have a significant influence.

Because the important investigations of this research were performed only after the 2004 NIST SRE had been completed, there was no need to select a decision threshold. The SRE was simply used as a platform for performing the investigations.

Apart from the DET curve, two more specific performance measurements were chosen. Firstly, the popular equal error rate (EER) shows how a system performs when it is equally likely to make either false alarms (FAs) or false rejections (FRs). Many researchers quote improvements in this figure in order to convey benefits of new techniques. Secondly, the FR rate (FRR) where the FA rate (FAR) equals 1% is an indication of system performance in the region where verification systems operate for applications such as access control. This is not a standard performance figure, but it was used for this thesis to investigate effects that were produced by the T-BAGMM in that region.

The University of Stellenbosch participated in the 2004 NIST SRE in collaboration with Spescom DataVoice (Pty) Ltd. The DataVoice system was rated among the better performing systems in that particular SRE. This system is used by this thesis as a reference system. Although this reference system was configured and trained much better than can be achieved by this research, its performance is used as a target goal for improvements to the baseline system.

7.4.2 Brief Description of the Reference System

The reference system was developed with data from the 1998 to 2002 NIST SREs, as well as some telephone conversations recorded locally by DataVoice.

Front-end processing

Speech signals were analysed with 32 ms frames, their starting points following 10 ms after each other. For each frame, 20 triangular band-pass filters were used in the 300 Hz to 3200 Hz range to extract 12 Mel-frequency cepstral coefficients (MFCCs) after discarding the zeroth coefficient. Each speech segment was subjected to short-time Gaussianisation [25] using a 3-second sliding window. Δ -features were then calculated with a 5-frame window to provide 24-dimensional feature vectors.

Silences were removed from the Mixer data by examining frame energies. Energy peaks and valleys were used to segment the speech into sections. The low energy frames of each section were discarded. Within each section, these

low energy frames acts as separators for what is called syllable nuclei. The mean energy of each syllable nucleus is examined further and those nuclei with very low energy are discarded.

A 5-component GMM is trained with the expectation maximisation (EM) algorithm using the log energies of the remaining nuclei for the entire speech segment. If a mixture component is found to have a small mean value (low energy) and a small variance, but also a large mixture weight, then that component is assumed to represent the background noise. A threshold is then set to 3 standard deviations above the mean of that component. All frames below this threshold are discarded. This step is not used for very short speech segments. The entire silence removal process retains an average of 25% of the frames.

Universal Background Model (UBM)

A single 512-component GMM-based UBM was trained using the pooled data from the 1999 to 2002 NIST SREs. Only the test segments were used. The speech segments were shortened by using only every 20th frame. The model was trained with LBG¹ k-means initialisation, followed by 5 iterations of the EM algorithm.

Speaker Models

Speaker models were trained with 5 iterations of mean-only MAP adaptation, using a relevance factor of 16. Training was accelerated by a variation of the UBM-based selection procedure. For each feature vector, the 5 components that contribute the most to the UBM likelihoods were selected. Only these components were re-estimated for the corresponding feature vectors.

Evaluation

Models were evaluated using the UBM-based selection procedure to compute only 5 mixture components for every test feature vector. Test normalisation (T-Norm) was applied with impostors consisting of 50 male and 50 female, randomly selected speakers from the 2002 NIST SRE data.

¹The Linde, Buzo, and Gray vector quantisation method

7.4.3 Procedure for Full Performance Evaluation

To execute a full performance evaluation, the following steps must be performed:

1. Create a data set for all of the speech data and perform front-end processing and feature extraction on the data
2. Create a UBM training data set and train the UBM
3. Create a training data set for each impostor model and train impostor models
4. Create a training data set for each target model and train target models
5. Create a data set for the test speech and verify the test speech against the appropriate target models

Fortunately, it is not necessary to repeat all of these steps for every new experimental investigation.

7.5 The Baseline System Performance

7.5.1 Motivation

The first experimental evaluation that was based on the NIST SRE conditions, was executed to determine the performance of the baseline system. The result can be compared to that of the reference system to get an indication of how much room there is for improvement. The result can also be used in further experimental work to determine how much certain techniques improve or degrade system performance. Because this was the first evaluation, it also served as a platform for gaining familiarity with working under the NIST SRE conditions and system evaluation in general.

A second evaluation was also performed where the GMMs in the baseline system were replaced with T-BAGMMs. The results were compared to the GMM-based baseline system in order to determine whether the tree-based approximations showed any evidence of degraded performance.

7.5.2 Setup and Execution

The details of configuring the baseline system were already discussed in Section 6.5. Each of the tasks outlined by Section 7.4.3, except for the UBM training, was distributed among 10 to 12 computers having different speed and

memory capabilities. The least capable machine had a 700 MHz processor and 256 Mb primary memory.

A full system evaluation was performed for the baseline GMM system as well as the baseline T-BAGMM system. A testing node score beam width $\delta_{test} = 0$ (refer to Section 6.4.1) was used for T-BAGMM trials. This means that for each pair of sibling nodes, only the subtree of the node with the highest likelihood contribution will be traversed further. The results are shown in the next section and discussed in Section 7.5.4.

7.5.3 Results

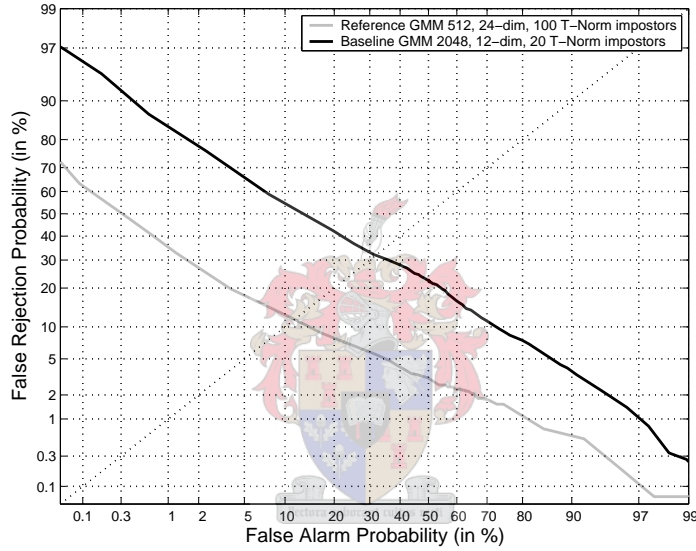


Figure 7.1: DET curve showing the difference between the performance of the baseline and the reference systems.

System	FRR at 1% FAR (in %)	EER (in %)
Baseline GMM	83.68	32.00
Baseline T-BAGMM	82.22	33.55
Reference GMM	35.81	11.60

Table 7.1: Comparison of error rates between the baseline GMM, the baseline T-BAGMM and the reference system

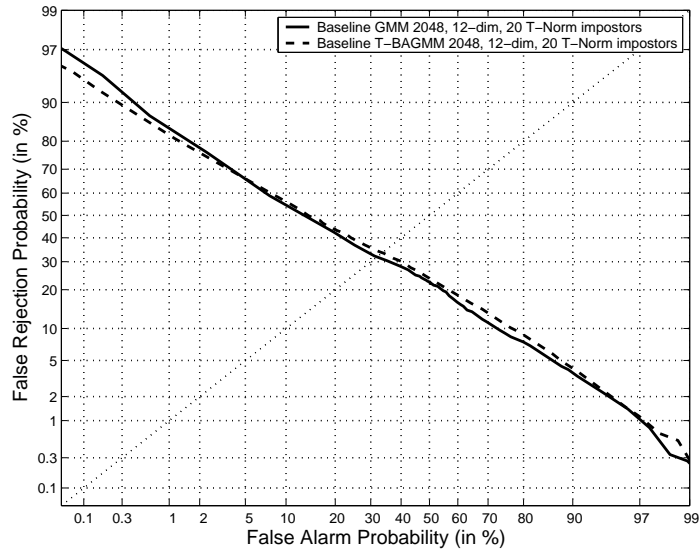


Figure 7.2: DET curve showing the difference between the performance of the baseline GMM and the baseline T-BAGMM system.

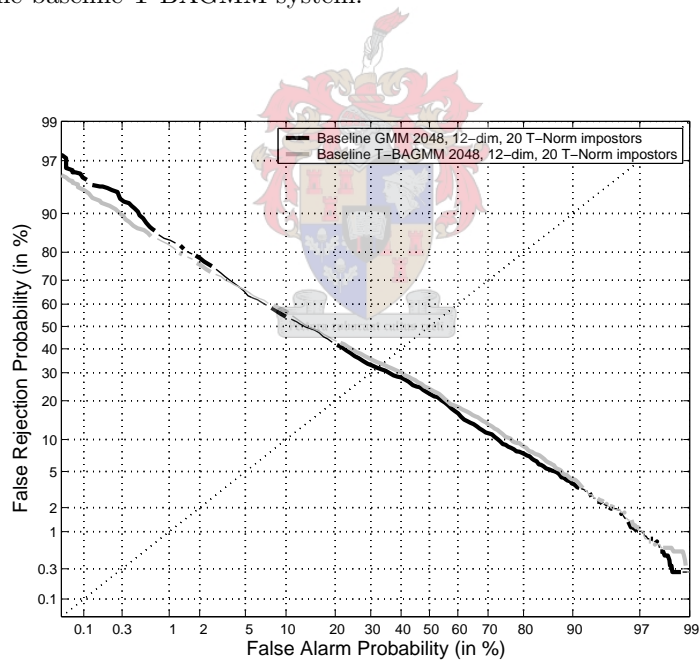


Figure 7.3: DET curve showing the statistical significance of differences between the performance of the baseline GMM and the baseline T-BAGMM system. A significance level of 5 % ($\alpha = 0.05$) was used for this figure. Areas with thick lines show significant difference and areas with thin lines indicate where differences are doubtful.

7.5.4 Interpretation

A clear performance gap between the baseline GMM and the reference system can be seen. This was expected, because the baseline system has such a low complexity compared to the reference system. The rest of this chapter will show how much of this performance gap can be closed by various improvements.

At a first glance, it seems as though the performance of the T-BAGMM system is virtually identical to that of the GMM system. However, closer inspection reveals that there is a slight degradation in the EER, but also a slight improvement in the low FAR regions. The results of the McNemar statistical significance test shown in Figure 7.3 indicate that these differences are not merely due to chance effects, but are in fact significant. The numerical values for these differences, as shown in Table 7.1, turns out to be relatively small, though. The difference in the EER is less than 2 %.

The fact that there is an improved performance for low FAR regions is unexpected, but intriguing. Practical speaker verification systems typically operate at low FARs when used for applications such as access control. It is possible that the approximations that are introduced by the T-BAGMM may prevent some kind of bias caused by over-training, but this is only speculation. Although it seems to be worthwhile to investigate, it was not yet possible to find the true cause of this phenomenon within the extent of this research. However, it is interesting to note the resemblance with the effect that Test normalisation (T-Norm) has in rotating the DET curve counter-clockwise [29].

7.6 Computational Characteristics of the T-BAGMM

7.6.1 Motivation

The new T-BAGMM was designed to increase computation speed. In the previous section, initial performance differences were seen between the GMM and this new T-BAGMM. This section will take investigations one step further to study the speed differences in detail. Performance differences will also be shown in relation to the speed differences.

The speed increase is dependent on the number of mixture components (leaf nodes), as well as the node score beam width δ . This section investigates these dependencies in detail and presents a characterisation of speed improvement with regard to each. These characterisations should be useful to other researchers when a choice for trade-off between accuracy and speed must be

made.

7.6.2 Setup and Execution

During this investigation, matching GMM and T-BAGMM systems were compared to each other with regard to execution time. Two speed comparisons were made: one regarding the UBM training, and the other regarding the completion of all the trials for the test condition. Differences in accuracy were also noted for these speed comparisons.

The systems that were subjected to these comparisons had the same basic configuration as that of the baseline system described in Section 6.5. The baseline configuration (having such low complexity) was chosen so that evaluation of the GMM-based systems could be performed in the shortest possible time.²

Training speed improvement was characterised by using the UBM training time. The UBM training time was measured as the total number of elapsed CPU clock cycles for 9 iterations of the EM algorithm. These were the 9 iterations directly after the model was completely initialised via the binary-split procedure. Measurements were made for component counts of $K \in \{2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$. For each of these component counts, the training node score beam width was set to each of $\delta_{train} \in \{0, 1, 3, 7, 10\}$ for the T-BAGMM UBM. All of these training runs could be performed because the UBM training time was reasonably short for each of them.

Evaluation speed improvement was characterised by means of the trial execution time. The trial execution time was measured as the number of CPU cycles elapsed while calculating the score of the given test utterance. Obtaining the score of an entire test utterance consists of evaluating the target model and the 20 impostor models for each feature vector in the test utterance. This excludes the loading of data and models into the computer's memory.

For the characterisation of evaluation speed improvement, T-BAGMM models were trained with a training node score beam width of $\delta_{train} = 3$. For the T-BAGMM systems, measurements were made for each of the testing node score beam widths $\delta_{test} \in \{0, 1, 3, 7, 10\}$. This required so many full system evaluations that it was decided to obtain measurements only for component counts of $K \in \{2, 8, 32, 128, 512, 2048\}$.

From the full system evaluations that had to be performed for characterising evaluation speed improvement, information about accuracy could also be obtained. This information, together with the speed characterisations, can be used

²Full execution of training and testing with a 2048-component GMM system takes about two weeks when distributing the work among 10 processors.

to find suitable trade-offs for different situations. Unfortunately, these measurements for accuracy could only be gathered for T-BAGMM models trained with one specific δ_{train} . It would be good to have accuracy measurements for different δ_{train} in order to find suitable trade-offs for model training. But, that would require too much time to complete, and so it was decided to leave the exercise for future investigations.

However, a compromise can be made by regarding the model log-likelihood for the training data. It is thought reasonable to suppose that if models of similar structure fit the same training data similarly (have similar likelihoods), then they should produce similar system performance. Therefore, if the training data log-likelihood for T-BAGMMs are nearly equal to that of the GMMs, then they should exhibit nearly equal system performance. Of course, it is expected that their performance will differ, but this technique can say nothing about the manner in which they differ. For different δ_{train} , the UBM training data log-likelihood was available after training of both the GMM and the T-BAGMM UBMs. The difference in these likelihoods was used as an indication of performance differences.

The software used for this research included a component score beam width for GMM training, $\delta_{train,GMM}$. This is similar to the training node score beam width of the T-BAGMM, δ_{train} . It is used to ignore component contributions during training when their responsibilities are too low. To compare the T-BAGMMs to well-trained GMMs, a value of $\delta_{train,GMM} = 10$ (logarithmic scale) was chosen for training the GMMs.

As mentioned above, it was decided to measure evaluation times with T-BAGMMs that were trained with a $\delta_{train} = 3$. This applied to both UBM training and specific speaker model training. This value was chosen (after considering the results obtained from UBM training measurements), because it seemed to be a good trade-off between accuracy and speed of training. It was essential to train good models as quickly as possible, because so many full system evaluations were required.

7.6.3 Results

Training Results

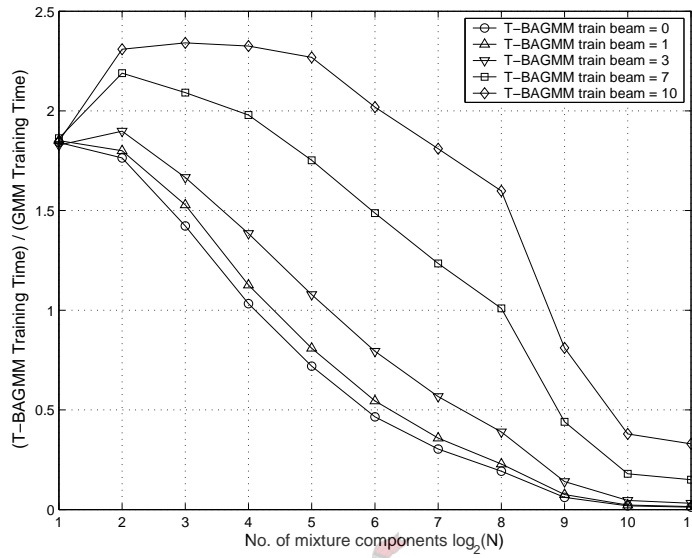


Figure 7.4: Training times for the T-BAGMM UBM relative to that of GMM UBM. Lower values are better. Where the ratio is unity, the training times for both models are the same.

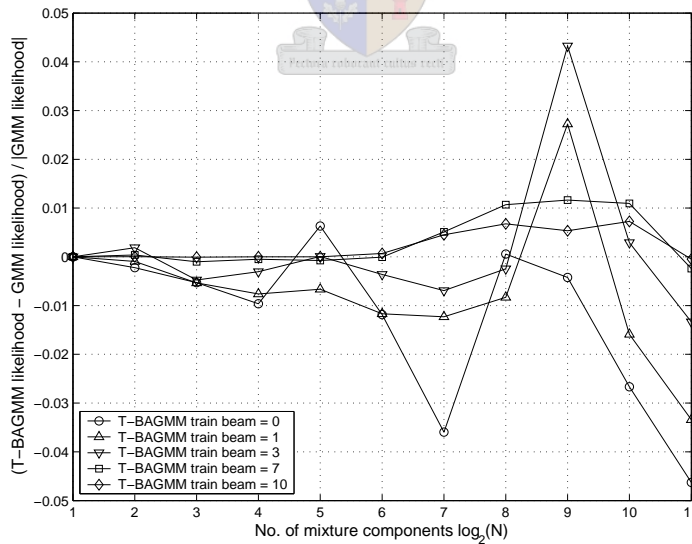


Figure 7.5: Relative difference between training data log-likelihoods of T-BAGMM UBM and that of GMM UBM. Zero values indicate identical performance. Positive values represent a better fit of the training data by the T-BAGMM.

Evaluation Results

Figure 7.6 and Figure 7.7 show the relative difference in evaluation time between the T-BAGMM and the GMM. Also shown, is a curve indicating the theoretical lower boundary for the reduction in evaluation time. In Figure 7.7, it is a upper boundary on the speed improvement. This boundary only takes into account the number of components or nodes that are evaluated:

$$\frac{\text{number of nodes evaluated for } T - \text{BAGMM}}{\text{number of components evaluated for GMM}}$$

The theoretical lower boundary is calculated for a test node score beam width of $\delta_{test} = 0$, because it is impossible to theoretically determine the number of evaluated nodes for any other beam widths. This means that only the subtree of the node with the highest score will be evaluated further when two sibling nodes are compared to the node score threshold.

It is interesting to note in Figure 7.7 that the measured speed increase for 2048 mixture components can be seen to exceed the theoretical upper boundary. This is discussed further in Section 7.6.4.

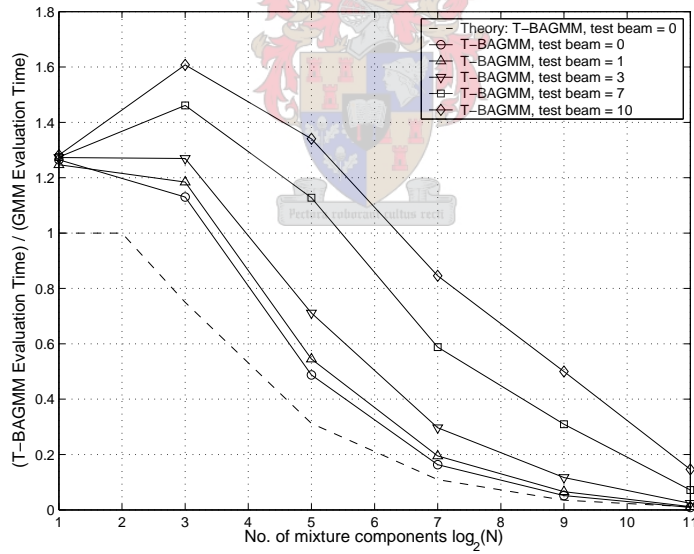


Figure 7.6: Evaluation time for the T-BAGMM relative to that of the corresponding GMM, shown with respect to the mixture component count. Curves for different test beam widths are shown. The theoretical lower boundary is also shown.

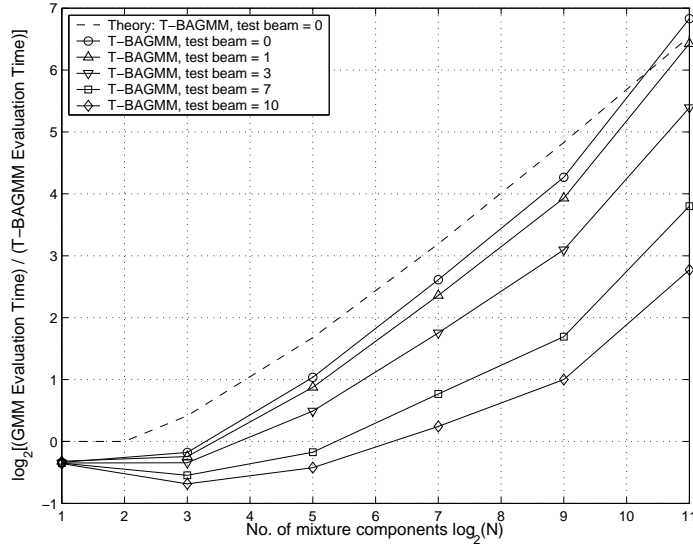


Figure 7.7: Evaluation time for the GMM relative to that of the corresponding T-BAGMM (base-2 logarithmic scale), shown with respect to the mixture component count. Curves for different test beam widths are shown. This figure shows more clearly how the measured speed increase exceeds the theoretical upper boundary.

Component count	2	8	32	128	512	2048
GMM	40.01	35.88	34.07	33.01	32.36	32.00
T-BAGMM, $\delta_{test} = 0$	40.01	35.78	34.22	33.71	33.79	33.55
T-BAGMM, $\delta_{test} = 1$	40.01	35.79	34.13	33.58	33.53	33.49
T-BAGMM, $\delta_{test} = 3$	40.01	35.78	34.15	33.53	32.86	33.40
T-BAGMM, $\delta_{test} = 7$	40.01	35.62	34.06	33.16	32.82	32.12
T-BAGMM, $\delta_{test} = 10$	40.01	35.58	33.98	33.08	33.08	32.19

Table 7.2: EER measurements (values in %)

Component count	2	8	32	128	512	2048
GMM	94.56	90.06	87.27	85.44	83.71	83.68
T-BAGMM, $\delta_{test} = 0$	94.56	88.60	86.22	84.61	83.21	82.22
T-BAGMM, $\delta_{test} = 1$	94.56	88.26	86.69	84.36	83.18	81.46
T-BAGMM, $\delta_{test} = 3$	94.56	87.92	86.66	84.39	82.24	81.63
T-BAGMM, $\delta_{test} = 7$	94.56	88.03	87.26	85.33	83.92	83.25
T-BAGMM, $\delta_{test} = 10$	94.56	87.88	87.28	85.69	83.82	84.38

Table 7.3: FRR measurements where FAR=1% (values in %)

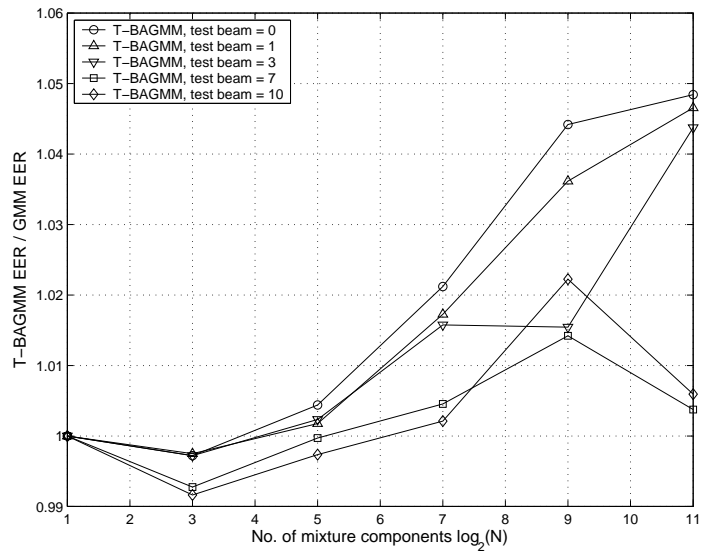


Figure 7.8: EER for the T-BAGMM relative to that of the corresponding GMM, shown with respect to the mixture component count. Lower values are better. This figure shows that the accuracy of a T-BAGMM system differs very little from that of a regular GMM system.

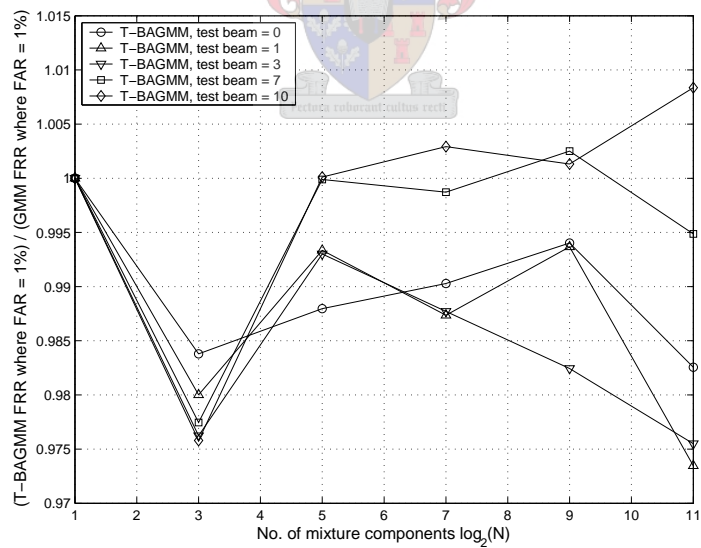


Figure 7.9: FRR at 1% FAR for the T-BAGMM relative to that of the corresponding GMM, shown with respect to the mixture component count. Lower values are better. This figure shows that the accuracy of a T-BAGMM system using a low beam width is consistently better in this region than that of a regular GMM system.

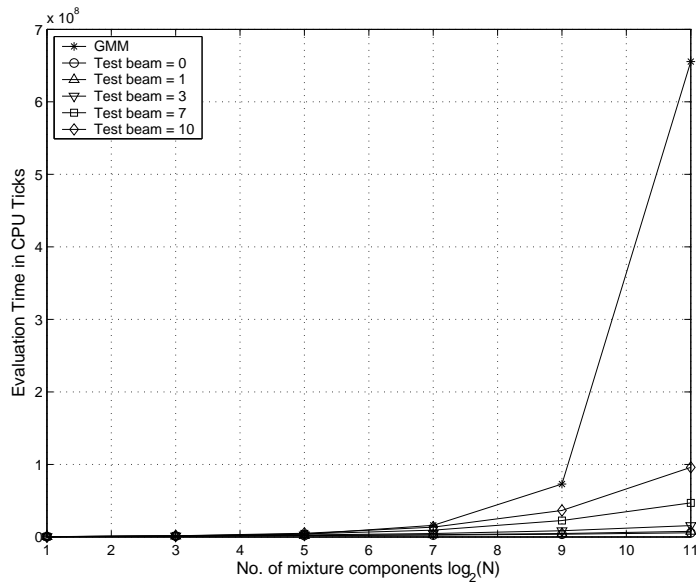


Figure 7.10: Absolute evaluation time given as the number of processor clock cycles (CPU ticks). This figure shows how the evaluation times differ in reality between systems using different component counts. These differences are not visible on the figures that show relative times.

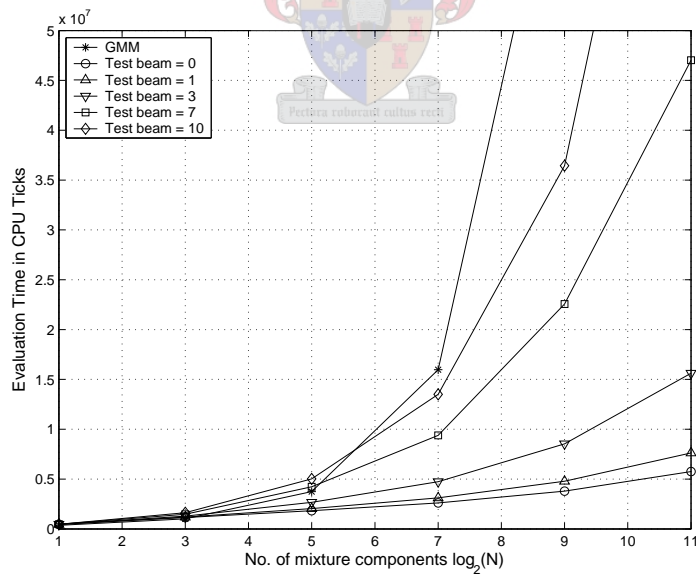


Figure 7.11: Absolute evaluation time given as the number of processor clock cycles (CPU ticks). This is a close-up of the previous figure. It can be consulted to avoid a speed penalty when more mixture components are needed (i.e. higher accuracy) for a particular application.

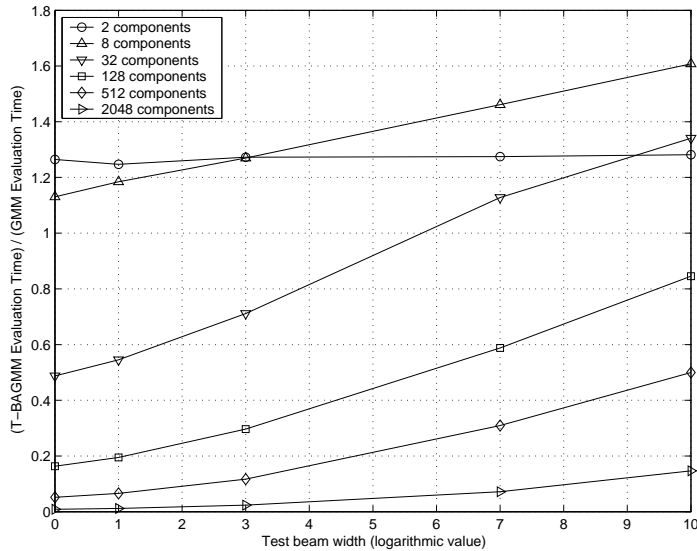


Figure 7.12: Evaluation time of the T-BAGMM relative to that of the corresponding GMM, shown with respect to the testing node score beam width. Curves for different mixture component counts are shown. This figure gives an alternative perspective to the speed-accuracy trade-off.

7.6.4 Interpretation

Before considering the results, three points must be noted. Firstly, the speed improvement curves do not present very precise measurements. Measuring with regard to CPU clock cycles was considered to be the most accurate. But, different processors with different architectures were used co-operatively. A difference in architecture means that one processor might execute more instructions per clock cycle than another one. Fortunately, many jobs were distributed among the processors. This would hopefully produce an average time measurement and ensure that the mentioned differences do not affect the results significantly. Yet, the results that were obtained can at least be used as an indication of possible trade-offs.

Secondly, the program code for the T-BAGMM was not yet optimised by the time these investigations were made. Therefore, the speed improvements shown here can be regarded as a worst-case instance. Furthermore, the T-BAGMM structure prevents the application of certain optimisation techniques that were used for the GMM program component. In turn, this will probably prevent the speed improvement curve from matching the theoretical predictions exactly.

Thirdly, the accuracy measurements should not be regarded as being generally applicable. It is most likely dependent on the specific kind of data that

were used. However, the accuracy measurements should correspond well with that of data having similar quality (i.e. conversational telephone speech).

From Figure 7.4 and Figure 7.6, it can be seen that large gains in speed are observed, especially when using a large number of mixture components. For larger node score beam widths and low component counts, the speed for both training and testing is decreased considerably. This was expected, because many more nodes must be evaluated. Also, in the case where 2 mixture components are used per model, the difference in overhead is clearly seen. For this case, the structures of the GMM and T-BAGMM are conceptually exactly the same, but more overhead is required for managing the T-BAGMM structure. The fact that the T-BAGMM program code was not optimised also plays a role here.

Figure 7.6 shows that the practical result converges with the theoretical prediction. However, the alternative perspective of Figure 7.7 indicates that the practical measurements for 2048 components are even better than predicted. One possible explanation for this regards the number of tree levels that are traversed per test feature vector. The program code allowed the binary-split algorithm to stop the splitting of nodes for which too little training data were available (i.e. when the combined responsibilities of the training vectors were not large enough). It seems that many such nodes may have been located in the 9th or 10th tree level. Furthermore, many test feature vectors may have caused the model evaluation to traverse up-to those nodes. This would then result in a time measurement for the T-BAGMMs corresponding more to the situation where the trees have 9 or 10 levels rather than the expected 11 levels.

Tree levels	Speed improvement factor
11 (measurement)	113.81
9	$\frac{2048}{9 \times 2} = 113.78$
10	$\frac{2048}{10 \times 2} = 102.40$
11	$\frac{2048}{11 \times 2} = 93.09$

Table 7.4: Speed improvement factors for the case where a GMM has 2048 mixture components and the corresponding T-BAGMM has a depth of either 9, 10 or 11 levels.

Unfortunately, this could not be verified properly in the time given, but an approximate verification can be made. From Table 7.4, it can be seen that the measured speed improvement does correspond somewhat (taking note of measurement inaccuracies) to the prediction when making use of the above explanation. This does not mean that the given explanation is true, and it also

does not explain why models with 1024 mixture components do not exhibit the same behaviour. Further investigation is required.

When looking at Figure 7.5, it is seen that the likelihoods of the T-BAGMM UBMs have no more than a 5% relative difference from the corresponding GMM likelihoods. This indicates that the T-BAGMM should provide nearly the same accuracy as the GMM. In the previous section, this fact has already been verified to a certain extent for a specific case. Figure 7.8 shows that the maximum relative difference in the EER is less than 5%. For that case, this difference translates to 5% of 32%, which is effectively a 1.6% difference in the absolute EER values. For higher testing node score beam widths, the EER for the T-BAGMM approaches that of the GMM to a certain extent.

It is interesting to note that the measured EER values for T-BAGMMs with 8 mixture components are better than that of the corresponding GMMs. Figure 7.9 also shows better performance in the low FAR region for this case. Yet, it is observed that the execution speed is lower than that of the 8-component GMMs. These differences in accuracy are very small and probably nothing more than interesting. However, for models with more than 8 components, the larger node score beam widths definitely produce accuracies that are closer to that of the regular GMM. It seems that performance is consistently better in the low FAR region when using a small testing node score beam width. Overall, there is very little difference in accuracy between the GMM and T-BAGMM.

The capability of the T-BAGMM can be summarised by the following example: For the regular GMM, a reduction in the number of mixture components from 2048 to 32 will provide a speed increase factor of 64. This will increase the EER from 32% by 2% to 34%. However, by using a T-BAGMM with a testing node score beam width of $\delta_{test} = 1$, the speed can be increased by a factor of 86. The EER will increase from 32% by 1.5% to 33.5%, but the FRR reading where FAR=1% will decrease from 83.7% by 2.2% to 81.5%. Although more memory will be required, this example indicates that the T-BAGMM would be much more preferred for increasing speed if accuracy is important for a given application. Unfortunately, a similar comparison with other speed improvement techniques could not be made. The result of this example might also differ when other improvements to the baseline verification system are considered.

Overall, these time characteristics can be a valuable tool. For any given application, it is possible to determine whether any speed gains will be seen when the T-BAGMM is used. By looking at the different curves, it is also possible to find out how much speed improvement should be observed. In conjunction with the accuracy measurements, a suitable trade-off can be chosen.

7.7 Influence of Mixture Component Count

7.7.1 Motivation

The structure of the GMM is defined by the number of mixture components. It is already known that the number of mixture components have some kind of influence on the performance of a speaker verification system. This section continues the investigation of Section 7.6 by providing some more detailed results that show this influence on the baseline system performance. It is, of course, possible that the exact influence might be different for more complex systems, but the general behaviour can be observed in this case.

7.7.2 Setup and Execution

The results of this section were obtained in Section 7.6. But, for this section, only the GMM systems and the T-BAGMM systems with $\delta_{test} = 0$ are of interest.

7.7.3 Results

The Gaussian Mixture Models

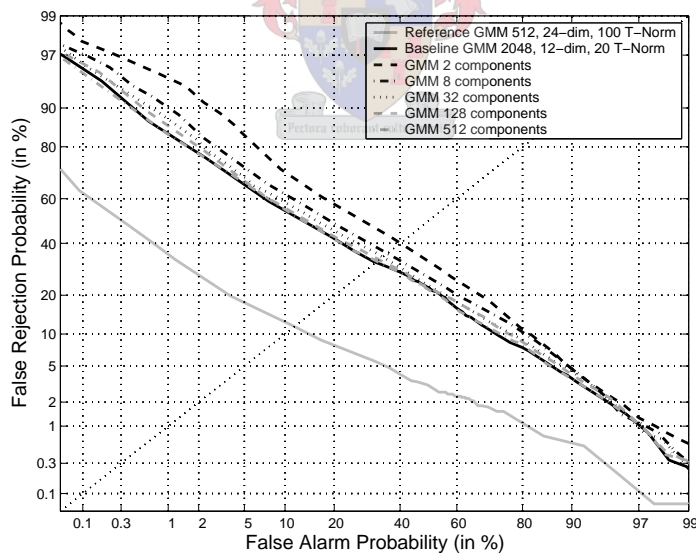


Figure 7.13: DET curve showing the influence of the number of GMM components.

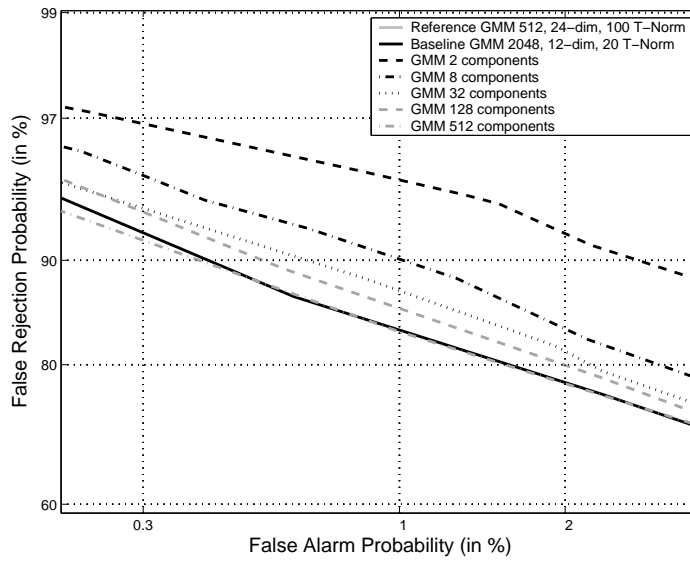


Figure 7.14: DET curve close-up showing the influence of the number of GMM components in the low FAR region.

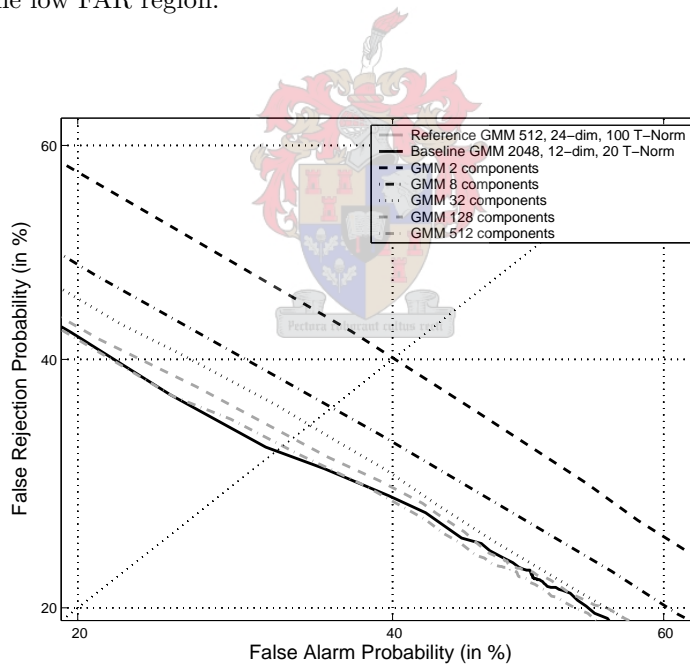


Figure 7.15: DET curve close-up showing the influence of the number of GMM mixture components at the EER.

The Tree-based Adaptive Gaussian Mixture Models

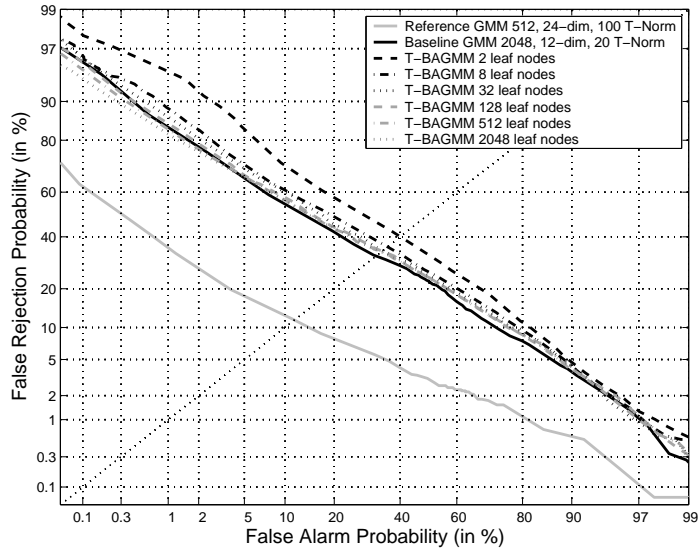


Figure 7.16: DET curve showing the influence of the number of T-BAGMM leaf nodes.

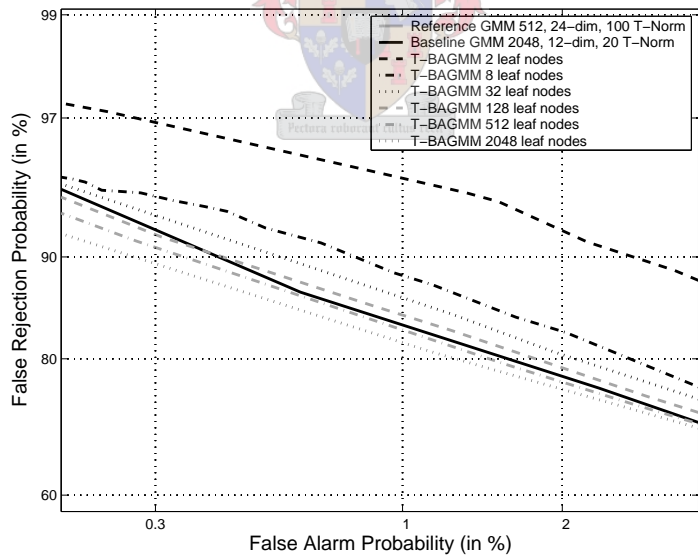


Figure 7.17: DET curve close-up showing the influence of the number of T-BAGMM leaf nodes in the low FAR region.

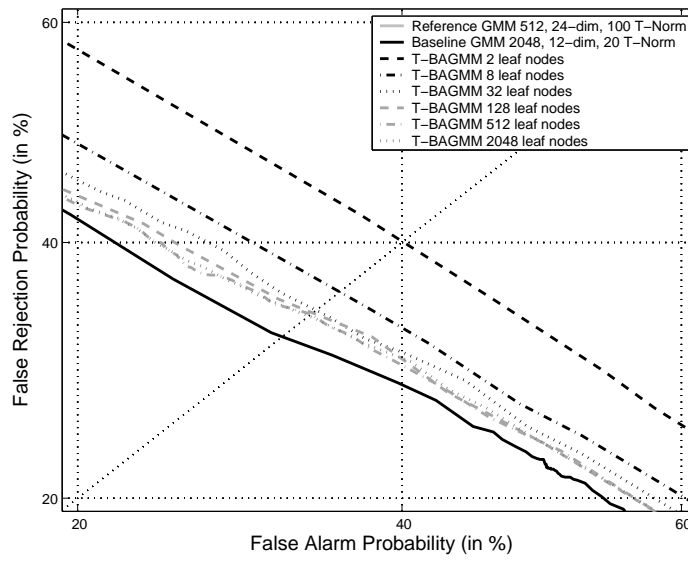


Figure 7.18: DET curve close-up showing the influence of the number of T-BAGMM leaf nodes at the EER.

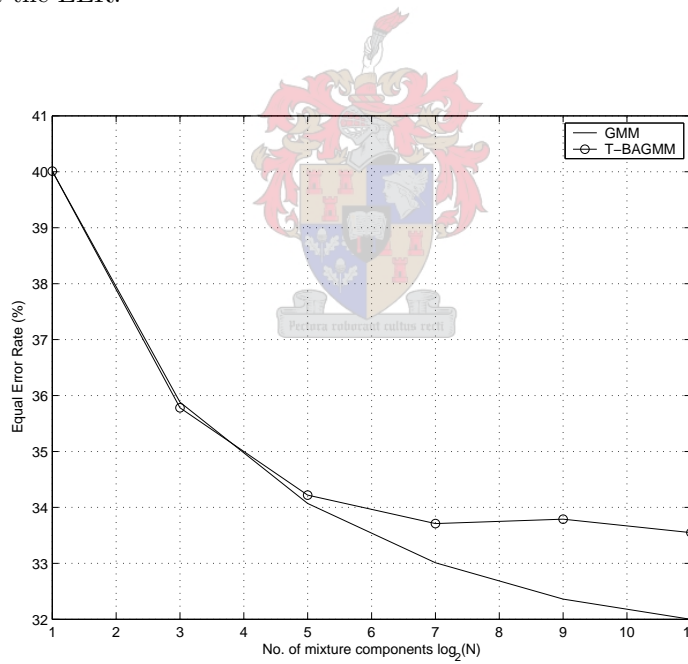


Figure 7.19: Comparison of the EER between the GMM and T-BAGMM, shown with respect to the number of mixture components (leaf nodes).

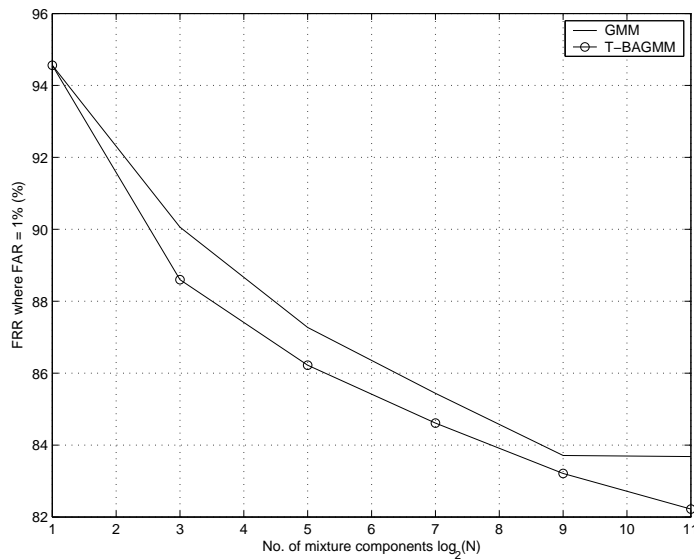


Figure 7.20: Comparison of the FRR between the GMM and T-BAGMM where the FAR=1%, shown with respect to the number of mixture components (leaf nodes).

7.7.4 Observations

As expected, the results show a decrease in system performance for both GMMs and T-BAGMMs when the number of mixture components are reduced. It can also be observed that 512 and 2048 mixture components result in virtually identical performance for the GMM case. This is also true for the T-BAGMM at the EER, but not entirely so in the low FAR region. It can again be seen that the T-BAGMM performs consistently better in the low FAR region than the GMM, but also consistently worse at the EER (except for 8-component models, of course). The results of this section presents nothing new, but it serves as a good reference for making decisions about system configuration.

7.8 Influence of Front-end Processing

7.8.1 Motivation

One objective of this thesis involves the construction of a fully working speaker verification system. It is therefore logical to investigate the influence of the various system improvements so that informed decisions can be made about the system's construction. This section investigates the influence of a few front-end processing techniques. These include cepstral mean subtraction (CMS), dynamic features and alternative cepstral feature selection. CMS is a chan-

nel compensation technique aimed at removing differences in speech that were caused by various channel conditions. Dynamic features incorporate the dynamics of the speech signal into features that can contribute to better recognition. Alternative cepstral feature selection refers to the popular technique of discarding the zeroth cepstral feature coefficient, which represents the energy of the frame.

7.8.2 Setup and Execution

For this investigation, different combinations of the mentioned techniques were applied to the baseline T-BAGMM system. CMS is simply performed by calculating the average (mean) feature vector for each conversation side and subtracting it from each feature vector of the corresponding conversation side. The average feature vector is calculated by adding the values for any given dimension across all the feature vectors and dividing the sum by the number of vectors. By doing this for every dimension of the feature vectors, a new vector can be constructed with the same dimension. This calculation is the maximum likelihood (ML) estimation of the mean vector for the data set.

Dynamic features were calculated as described in Section 5.2.4 after applying CMS to the MFCC features. The calculation was attempted for both a 2-point window (difference between two vectors) and a 5-point window (discrete differential filter). In the results that are shown here, the indicator "24-dim" refers to a feature vector of 12 MFCC features that is supplemented with 12 Δ -features. Similarly, the indicator "36-dim" refers to a "24-dim" feature vector that is supplemented with 12 $\Delta\Delta$ -features.

For alternative cepstral feature selection, 13 MFCCs were calculated instead of the usual 12. However, the zeroth coefficient is discarded to produce a 12-dimensional feature vector. This will be indicated with the label "13-1".

A new scaling feature normaliser was trained with the speech from the 200 UBM speakers after any different combination of CMS and dynamic features was applied to the data. These were used to normalise all features in the different experiments presented here.

7.8.3 Results

Cepstral Mean Subtraction

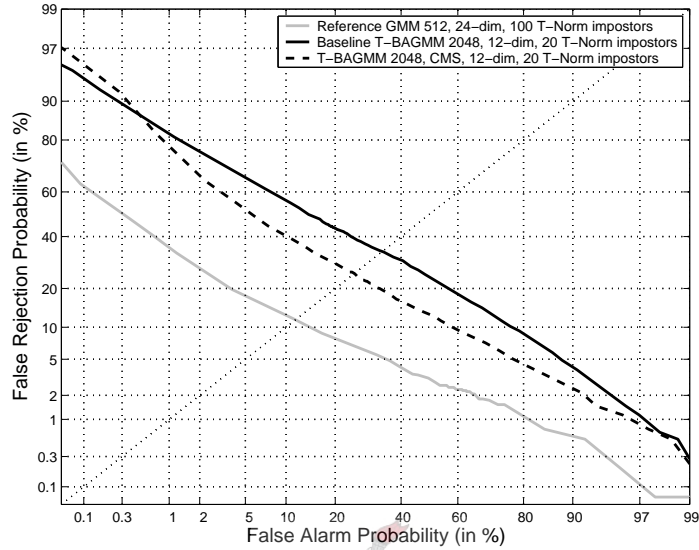


Figure 7.21: DET curve showing the influence of CMS.

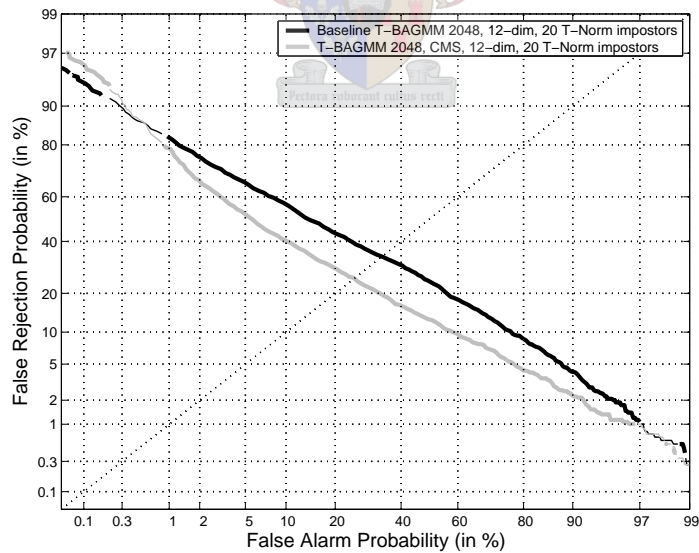


Figure 7.22: DET curve showing the statistical significance of differences between the performance of the baseline T-BAGMM system with and without CMS applied. A significance level of 5 % ($\alpha = 0.05$) was used for this figure. Areas with thick lines show significant difference and areas with thin lines indicate where differences are doubtful.

Dynamic Features

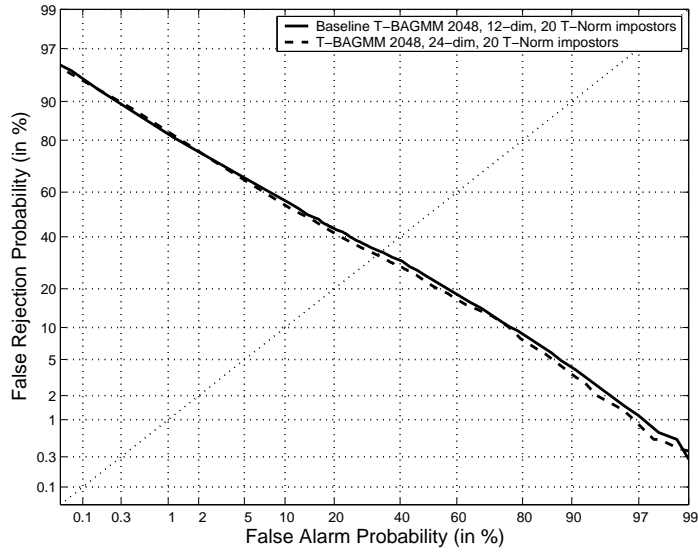


Figure 7.23: DET curve showing the influence of Δ -features without CMS applied. Curves are shown for calculation of Δ -features with a 2-point window.

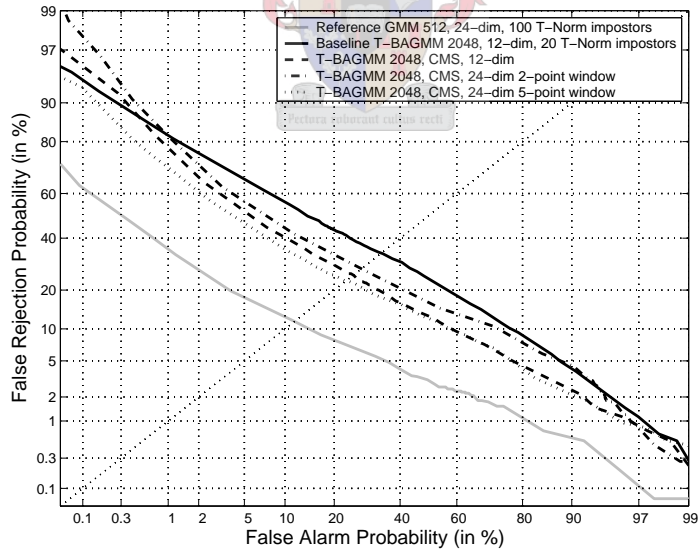


Figure 7.24: DET curve showing the influence of Δ -features, with CMS applied. Curves are shown for calculation of Δ -features with a 2-point window as well as with a 5-point window.

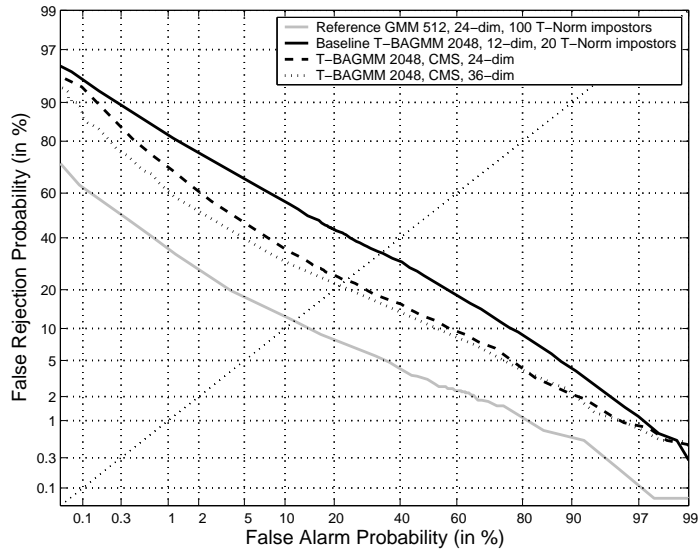


Figure 7.25: DET curve showing the influence of $\Delta\Delta$ -features. Only 5-point windows were used here.

Alternative Cepstral Feature Selection

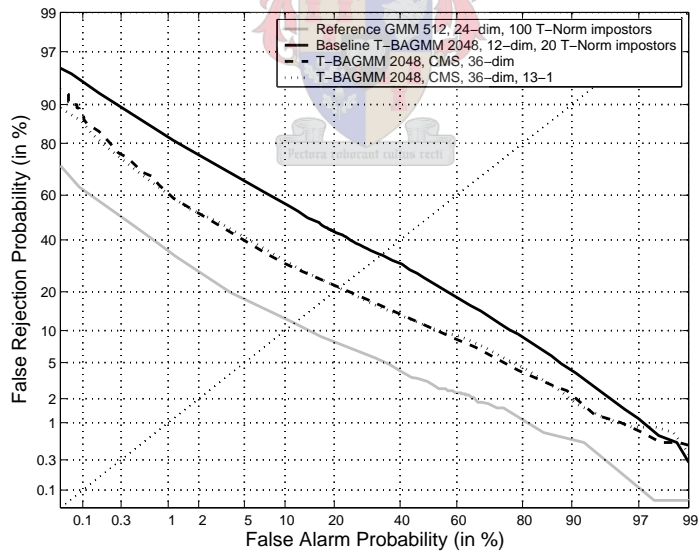


Figure 7.26: DET curve showing the influence of alternative cepstral feature selection (13 MFCCs minus the zeroth coefficient).

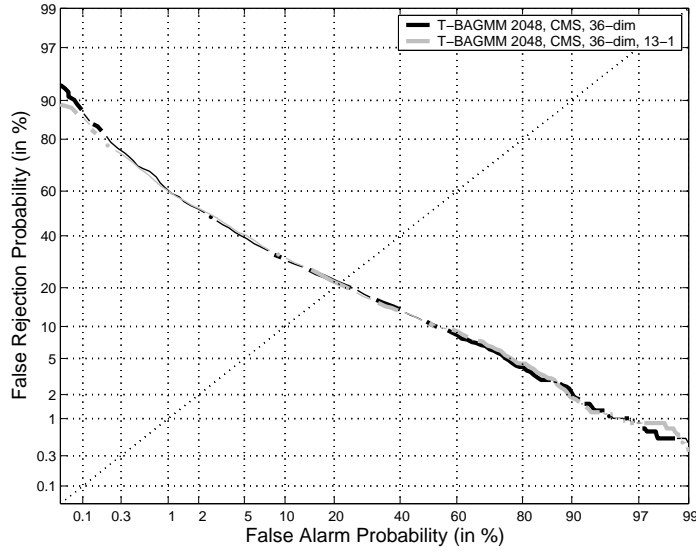


Figure 7.27: DET curve showing the statistical significance of differences between the performance of the T-BAGMM system using basic or alternative cepstral feature selection. A significance level of 5 % ($\alpha = 0.05$) was used for this figure. Areas with thick lines show significant difference and areas with thin lines indicate were differences are doubtful.

7.8.4 Interpretation

It is clear that CMS provides an enormous improvement in the general system performance. In this specific case, an 8.49% improvement in the EER is seen, decreasing from 33.55% to 25.06%. It is unfortunate that performance is worsened in the very low FAR regions. The result of the McNemar statistical significance test speaks for itself.

If Δ -features are calculated with a 2-point window and used without CMS, then a slight improvement in the middle and lower FRR regions is observed. This is confirmed by the McNemar test. However, when CMS is applied, the calculation of Δ -features with a 2-point window degrades performance overall. But, when using a 5-point window with CMS, the performance is improved, especially in the lower FAR region. This seems to act as compensation for the degrading effect that CMS has in that area. Using Δ -features in this way provides a further 2.01% improvement in the EER, from 25.06% to 23.05%.

When adding $\Delta\Delta$ -features that were also calculated with a 5-point window, the performance is improved still, and again more so in the lower FAR region. An additional improvement of 1.69% is observed for the EER, from 23.05% to 21.36%.

The overall improvement in the EER from the baseline T-BAGMM, is 12.19%, decreasing from 33.55% to 21.36%. The overall improvement in the FRR (where FAR=1%) from the baseline T-BAGMM, is 21.9%, decreasing from 82.22% to 60.32%.

It seems as though the alternative cepstral feature selection does not improve the system performance very significantly, or even at all. There is a slight improvement of 0.16% in the EER, from 21.36% to 21.20%. The FRR where FAR=1% does show a very small improvement of 0.24% from 60.32% to 60.08%, but the McNemar test shows doubt about its significance.

All of these results seem to indicate that front-end processing, and especially channel compensation, plays a very important role in obtaining good system performance. CMS, or any similar or better channel compensation, seems to be a requirement rather than an option. Also, the discrete differential filter using the 5-point window seems to be the preferred method for calculating dynamic features.

7.9 Influence of the T-Norm Impostor Set Size

7.9.1 Motivation

When using test normalisation (T-Norm), it is important to know how many impostor models should be used. Each impostor model is a complete speaker model. The required memory space increases with the number of impostors, but the accuracy might not follow the same trend. It is therefore preferred to find a suitable trade-off between memory requirements and accuracy. This section presents a simple investigation into the effect of the impostor set size on accuracy.

It seems reasonable to assume that the amount and quality of the impostor training data also has an influence on accuracy. But, this could not be investigated in the extent of this thesis.

7.9.2 Setup and Execution

For this investigation, the baseline T-BAGMM system including various combinations of CMS, Δ - and $\Delta\Delta$ -features were used. The large memory requirements of the models necessitated a reduction of the model size to 512 mixture components (or leaf nodes) when using 36-dimensional feature vectors and more than 50 impostor models. According to the results of Section 7.7, this model size seems to be sufficiently accurate to provide sensible results for this investigation.

The original memory management strategy had to be altered to make memory available so that more impostor models could be accommodated. Originally, all impostor models and all target models for all the trials in a given distributed job were loaded into main memory. Many of the trials shared the same target models. The ratio of trials to unique target models is anything between 20:1 and 70:1 (possibly even higher). But, an instance for each target model was loaded into main memory for each trial in which it was required. This is obviously not a very clever strategy, and it soon became apparent that too much memory is required for this scheme. It was therefore decided to load only unique instances of target models into main memory.

It was necessary to keep all the impostor models in main memory for all trials. If this was not done, then it could happen that one or more impostor models had to be loaded from disk storage for the evaluation of every new test feature vector. Because disk access is slow, this would eliminate any speed gains that the T-BAGMM provides.

It was also required that no memory paging (disk swapping) be allowed, because all of the hosts that participated in the distribution were used for normal desktop work by other users. The constant swapping of data between main memory and disk storage would cause a computer to become unresponsive.

7.9.3 Results

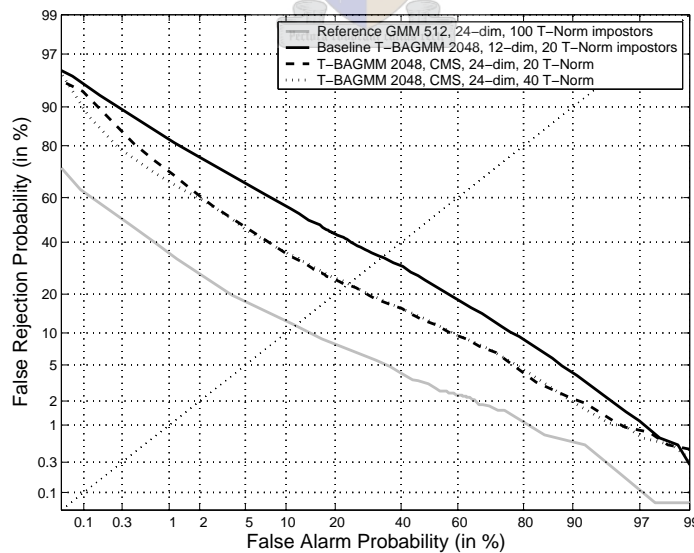


Figure 7.28: DET curve showing the influence of the number of T-Norm impostors using CMS and Δ -features.

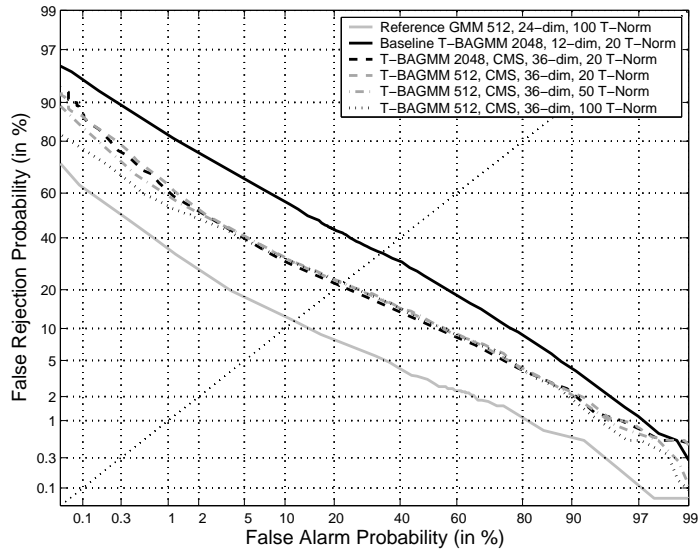


Figure 7.29: DET curve showing the influence of the number of T-Norm impostors using CMS and both Δ - and $\Delta\Delta$ -features.

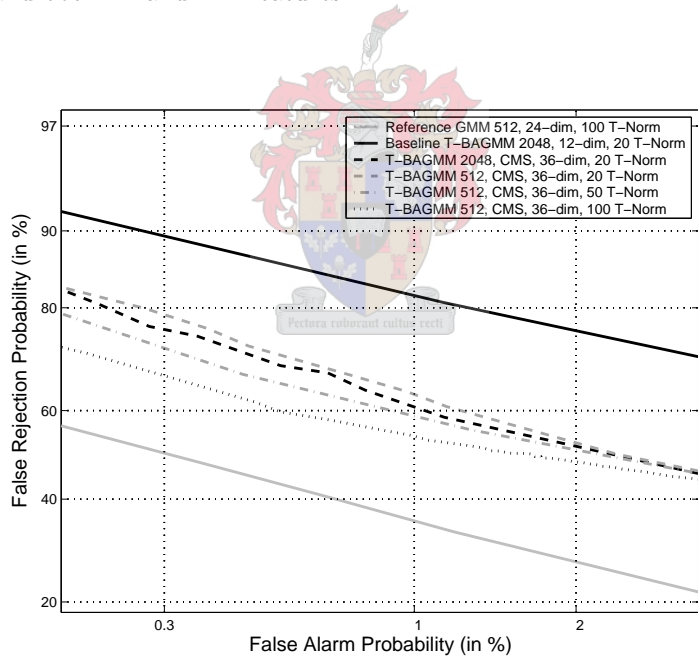


Figure 7.30: DET curve close-up showing the influence of the number of T-Norm impostors at low FAR.

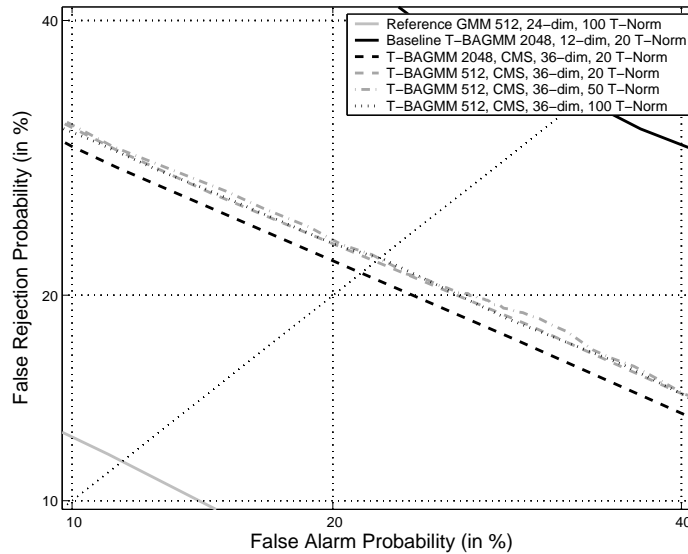


Figure 7.31: DET curve close-up showing the influence of the number of T-Norm impostors at the EER.

7.9.4 Interpretation

Firstly, it is seen that 512-component models produce slightly lower performance than 2048-component models. This was already observed in the low FAR region before CMS and dynamic features were applied. Now that CMS and dynamic features have been included, there is also a performance loss at the EER when fewer mixture components are used. Recall that in Figure 7.18 there was no perceptible difference. The difference here is very small, though.

Secondly, there is virtually no difference in performance around the EER as the number of impostor models increase. But, in the low FAR region, there is a definite increase in performance. Using up-to 100 impostor models therefore seems like a good idea when the system will operate in the low FAR region. This "tilting" of the DET curve is discussed by other researchers [29]. A full system evaluation for 2048 components per model and 100 impostor models could not be performed, because it would require too much reading from disk storage. An alternative strategy for managing the memory usage might help to alleviate the problem, but such a strategy could not be implemented in the allocated time.

7.10 Influence of the UBM Data Set Size

7.10.1 Motivation

The entire effort of this thesis is based on the correct processing of actual data. Therefore, it seems sensible to examine the effects of certain data-centred parameters. Training data are the only data that can be controlled to some extent. Although more thorough investigation is desired, only a simple preliminary study could be made to determine the effect of different numbers of training speakers in the UBM data set. This parameter was chosen, because intuition suggests that having more speakers in the UBM training data set should provide a more general UBM and hopefully better trained models giving better performance. Another parameter that might have been investigated, is the length of training speech samples for each speaker in the UBM data set, as well as for each speaker in the T-Norm impostor set.

7.10.2 Setup and Execution

For this experiment, the baseline T-BAGMM system was used with CMS and both Δ - and $\Delta\Delta$ -features included. Recall that the UBM of the baseline system was trained with speech from 100 male and 100 female speakers (a total of 200). For comparison, another evaluation was performed with a UBM that was trained with 200 male and 200 female speakers (total of 400). The same comparison was also made for two other systems: both using 512-component models, but one also using 100 T-Norm impostors instead of 20.

7.10.3 Results

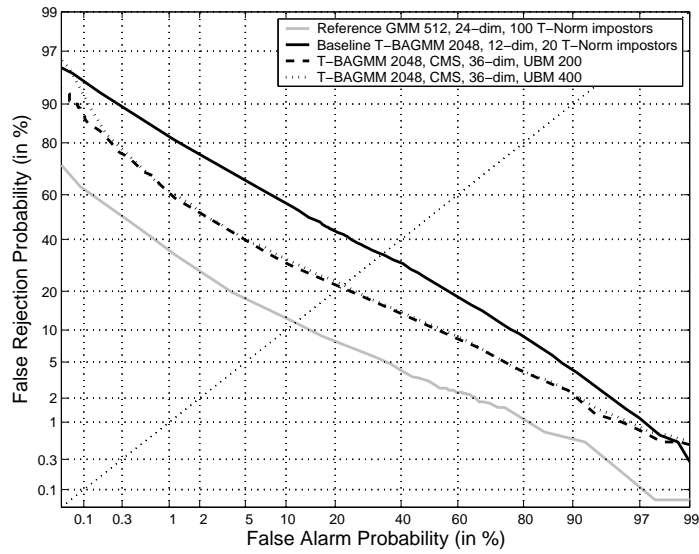


Figure 7.32: DET curve showing the difference between a system using 200 UBM speakers and one using 400 UBM speakers. Here, 2048 mixture components are used with 20 T-Norm impostors.

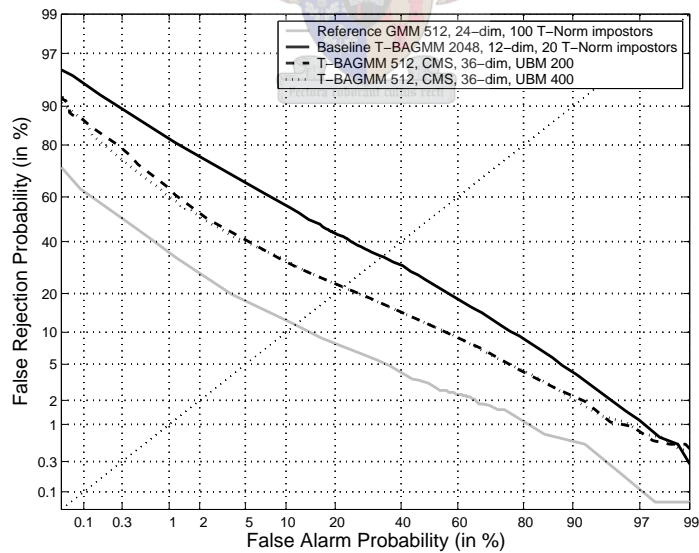


Figure 7.33: DET curve showing the difference between a system using 200 UBM speakers and one using 400 UBM speakers. Here, 512 mixture components are used with 20 T-Norm impostors.

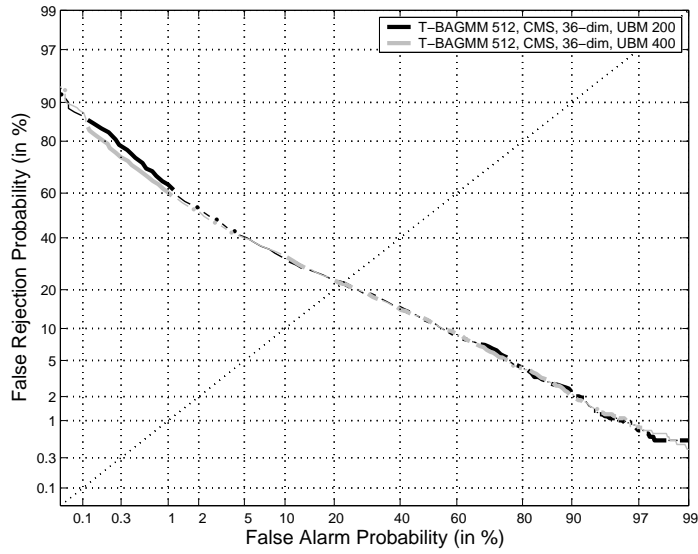


Figure 7.34: DET curve showing the statistical significance of differences between the performance of the T-BAGMM system using 200 or 400 UBM speakers with 512 mixture components. A significance level of 5 % ($\alpha = 0.05$) was used for this figure. Areas with thick lines show significant difference and areas with thin lines indicate were differences are doubtful.

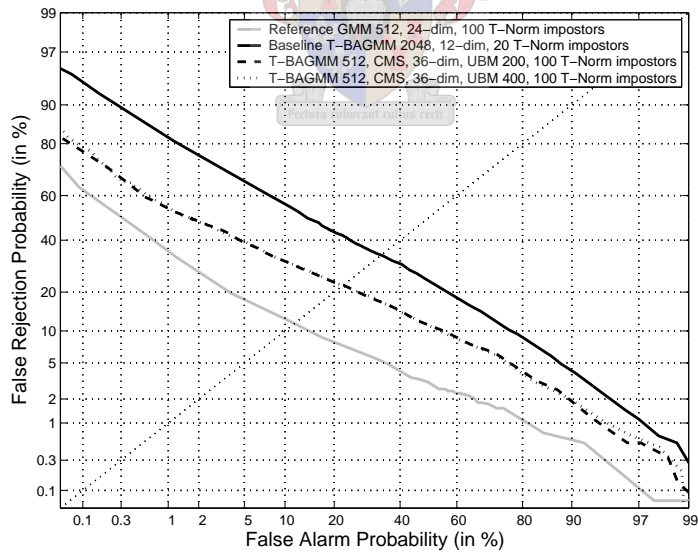


Figure 7.35: DET curve showing the difference between a system using 200 UBM speakers and one using 400 UBM speakers. Here, 512 mixture components are used with 100 T-Norm impostors.

7.10.4 Interpretation

It is fairly obvious that there is no practical difference between training a UBM with the speech of 200 people or 400 people. According to other research [6], there should be no difference between the system performance when using a UBM trained with one hour of speech and one trained with six hours of speech. For the results presented here, 60 seconds of speech were used per person. With 200 speakers, the total is 200 minutes, or 3 hours and 20 minutes. For 400 speakers, the total is six hours and 40 minutes. These results therefore seem to correlate well with the observations of others.

Some improvement in performance is observed in the low FAR region for a system with 512-component models, 20 T-Norm impostors and a UBM trained with 400 speakers. The McNemar test shows that this difference is significant. However, the cause of this effect is not known yet. It might be a good idea to conduct a much more thorough investigation into the effects of the UBM training data parameters.

7.11 MAP Adaptation for both Means and Covariance Matrices

7.11.1 Motivation

MAP adaptation can be used to train GMM speaker models by adapting any combination of means, covariance matrices and mixture weights from the UBM. It is now commonly known that the best performance is achieved when adapting with regard to only the means of mixture components. However, it was considered that this might be due to the use of the popular UBM-based component selection for speed increase [40]. Because the UBM is used to select components that should be evaluated for speaker models, it seemed that adaptation with regard to more parameters than the mean would cause the speaker models to become substantially different from the UBM. This would then cause the UBM-based selection to select inappropriate components and cause performance degradation.

The T-BAGMM is not as dependent on the UBM for selecting components with high scores. Speaker models only inherit their tree structure from the UBM. It was thought that the T-BAGMM might therefore show an improved performance when models were adapted with regard to the means and the covariance matrices. This section presents the result of such an investigation.

7.11.2 Setup and Execution

For this experiment, the baseline T-BAGMM system was used with CMS and both Δ - and $\Delta\Delta$ -features included. The alternative cepstral feature selection was used, although it was seen in Section 7.8 that this provides no significant advantage. Recall that speaker models of the baseline T-BAGMM system are trained using mean-only MAP adaptation with a relevance factor of 16.

An alternative system was configured which is identical to the one mentioned above, except that speaker models were trained using MAP adaptation with regard to both the mean and covariance matrix. Two tests were made with this system, where the one test used models adapted with a relevance factor of 16 and another with a relevance factor of 72. A higher relevance factor indicates that more of the UBM characteristics are retained in the speaker models.

7.11.3 Results

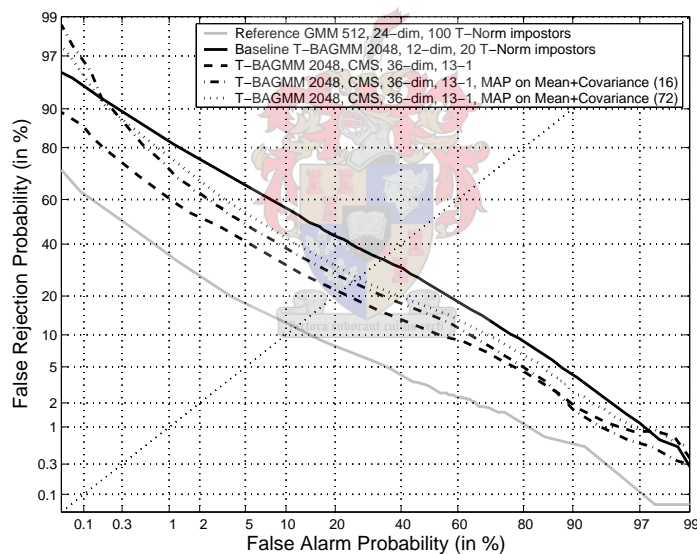


Figure 7.36: DET curve showing the difference between a system using mean-only MAP adaptation and one using MAP adaptation with regard to both the mean and the covariance matrix. The second system was tested with a relevance factor of 16 and again with a relevance factor of 72, as shown in parentheses.

7.11.4 Interpretation

The higher relevance factor definitely degrades performance. But more importantly, these results confirm again that mean-only MAP adaptation produces better performance. However, this may only indicate that the T-BAGMM

speaker models are not as independent on the UBM as originally thought. It is possible that the models that were adapted with regard to more parameters do not fit well into the tree structure that was inherited from the UBM.

On the other hand, it may just as well indicate that the cause lies somewhere else. It might be due to the limited amount of adaptation data, although this is unlikely. At least it is now known that the T-BAGMM behaves similar to the GMM in the aspect of MAP adaptation as well.

7.12 Summary

The best performance that was obtained by this research, is still far off from that of the reference system from DataVoice. It seems as though the front-end processing plays a very important role in this respect. A large factor is most likely the difference between using CMS and the superior short-time Gaussianisation. Another factor that may contribute to the performance difference, is the specific way in which development data are selected and applied.

However, this research aimed to increase the execution speed by means of the T-BAGMM, rather than to focus on factors that provide better performance. The results show that speed improvement measurements correspond well to theoretical predictions. For 2048-component models, the speed improvement is even better than predicted, reducing the GMM execution time by a factor of about 113! A possible explanation for this unexpected result was given, although further investigation may shed light on the true cause. It also seems that there is very little performance loss. Performance is even slightly better in the low FAR region, which is where systems typically operate for applications such as access control. No reason could yet be found for this behaviour, but it seems to resemble the effect that T-Norm has in rotating the DET curve counter-clockwise.

The T-BAGMM was successfully used to execute a fairly large number of experiments. Although these experiments are not exhaustive, they covered all the components that are used to construct a verification system: front-end processing, modelling, evaluation and data usage. The results can be used to make informed decisions about the construction of an acoustic speaker verification system. Without the speed improvement, the completion of all these experiments would not have been possible in the short time allocated for this research.

The most important investigation provided detailed results that characterise the T-BAGMM speed improvement. These characteristics can be used to determine whether the use of the T-BAGMM will be beneficial to any given

application. It can further be used to see how much of an improvement in speed should be observed. With the accompanying accuracy measurements, it is possible to select a good trade-off between speed and performance.

The investigations performed for this thesis are similar to that of previous research by some other researchers. This thesis served as a good platform to evaluate the various techniques on newly released data that were provided for the 2004 NIST SRE. The experiments were also conducted according to conditions that were defined for the 2004 NIST SRE. The results therefore provide information on how the corresponding techniques perform under these challenging conditions.

The investigations were performed for the various techniques on a single set of data under the same conditions. This gives a researcher the ability to observe how much different techniques alter performance relative to each other. For example, it is very clear that some form of channel compensation such as CMS is a very important requirement for a high-performance system.

It was not possible to perform a thorough T-Norm investigation for T-BAGMMs with 2048 mixture components. This was due to a shortage of computer memory. A few suggestions for an alternative memory management strategy are given here. Some memory could be freed by keeping only one target model in main memory at any given time. The trials are arranged in such a way that all trials using a given model will follow each other in a sequence. However, this will only allow about a handful of extra impostor models to be loaded for most cases.

A totally different strategy might load only one impostor model at any given time into main memory. This model will be evaluated for all of the test feature vectors in a test utterance and the results stored in main memory. This will then be performed for each impostor model. The final normalisation procedure can then simply recall the results that were stored in memory. This method will only be successful if the amount of memory required to store such a result matrix is small enough.

This method requires that all the impostor models must be loaded from disk for each trial. This will, of course, cause the speed to drop. Depending on the size of this result matrix, it might even be possible to evaluate a single model for more than one test utterance while that model is kept in memory. This will increase the speed of execution slightly, because less disk reads are required.

Chapter 8

Conclusions and Recommendations

8.1 Tree-based Adaptive Gaussian Mixture Model

This thesis presented the development of a new, time-efficient version of the Gaussian mixture model (GMM), called the tree-based adaptive Gaussian mixture model (T-BAGMM). Although other approaches exist for improving the execution speed of GMM evaluation, this new model has a few unique characteristics and possible advantages.

Popular approaches try to minimise the number of mixture components that need to be evaluated. They do this by finding those components that contribute the most to the model likelihood in an efficient manner. Some of these methods use tree-structures to perform the search. The T-BAGMM goes one step further by including components with lower contribution when they are evaluated as part of the search. This does not reduce execution speed significantly, and provides a smoother approximation to the regular GMM. The T-BAGMM effectively provides a multi-resolution GMM that models different regions in feature space with different degrees of detail, depending on where a particular test feature vector is located. The implementation of the T-BAGMM in this thesis also includes a parameter, called the node score beam width, that can be used to adjust the speed-accuracy trade-off. Detailed discussions about the T-BAGMM structure, related algorithms and implementation were provided.

The T-BAGMM was successfully implemented and tested under difficult conditions, and it proved to have been worth the effort. High speed gains for both training and evaluation were indeed obtained, although the loss due to overhead is more pronounced for lower mixture component counts. A thorough

investigation was performed to obtain speed improvement characteristics for the T-BAGMM. Improvement curves were obtained for both training and testing, with regard to the number of mixture components and the value of the node score beam width. These curves can be used to determine whether and how much of a speed improvement will be seen for any given application.

Verification accuracy is very similar to that of the regular GMM. This new model also provided a few positive surprises. Firstly, it seems as though better accuracies are produced at the low false alarm operating regions. This is especially interesting, because many practical systems in applications such as access control operate in those regions. Some speculation about the cause was given. However, during the course of performing the research for this thesis, no proper explanation could be found for this effect. It may be worthwhile to investigate this phenomenon further.

Secondly, it was observed that the speed improvement is better than the rough theoretical prediction when models have 2048 mixture components. A possible explanation was given, but it is uncertain whether it is accurate. The time measurements were also not especially accurate and may have contributed further to this effect, but it is doubtful that it could have been the only factor. If this matter must be investigated further, it should be done with different data sets and more accurate predictions and time measurements.

A curious effect with regard to speed was observed when training speaker models with MAP adaptation. It seems as though the execution speed increased for subsequent EM iterations. This was not yet verified for all cases. It may not necessarily lead to some important discovery, but it is probably a still good idea to investigate this matter further.

Apart from the benefits that were shown in this thesis, the T-BAGMM also allows the use of other techniques such as structural maximum *a-posteriori* (SMAP) adaptation [8]. This technique has already shown positive performance improvements for other tree-based GMMs in speaker verification [9]. An initial investigation into SMAP for the T-BAGMM has been performed during this research. There were differences between the MAP and SMAP formulas for covariance matrices in the particular literature. It was unknown whether it would be possible to re-use implementations of the MAP formulas for SMAP, but it was suspected to be so. A derivation was made to consolidate the MAP and SMAP formulas and it was found that they are identical, except for their context. Unfortunately, this derivation could not be reproduced here.

8.2 The Acoustic Speaker Verification System

In conjunction with the introduction of the new T-BAGMM, this thesis also fully described the construction of a complete acoustic text-independent speaker verification system. The best performance obtained by this research is still far away from that of a typical, modern high-performance system. However, this research aimed to increase the execution speed by means of the T-BAGMM, rather than focus on factors that provide better performance. The results testify that this has been done successfully.

It was shown how the likelihood ratio is used to make verification decisions. The likelihood ratio uses likelihoods that must be calculated for the target speaker and alternative (impostor) speakers. Theoretical models can be used to approximate these likelihoods, and descriptions of one such model, the GMM, as well as a faster version, the T-BAGMM, were provided. The GMM is suited well for acoustic speaker recognition. Some detail was given with regard to front-end processing for extracting useful information from speech signals.

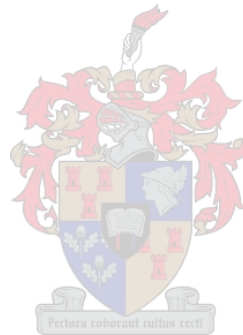
Even though other performance factors received little attention, some valuable experiments were performed. Basic investigations were made for each of the main components in the verification system. The influence of cepstral mean subtraction (CMS), dynamic features and cepstral feature selection were shown for the front-end processing part. The influence of the mixture component count, maximum *a-posteriori* (MAP) adaptation for different parameters, and data set size for the universal background mode (UBM) were shown for the speaker modelling part. The influence of the number of impostor speakers used by test normalisation (T-Norm) was shown for the evaluation part.

Although these experiments were merely less thorough repetitions of investigations that were previously made by other researchers, they provide a unique perspective. All of the experiments that were presented in this thesis were performed on newly released data that were used for the 2004 National Institute of Standard and Technology (NIST) speaker recognition evaluation (SRE). Not only did the NIST SRE present challenging conditions, but it is now possible to observe how these different techniques influence performance relative to each other. It is especially clear that a large improvement is provided by a channel compensation technique such as CMS. It is therefore obvious that the results can be used to make informed decisions about the construction of an acoustic speaker verification system.

It was mentioned that there is still much improvement to be made in order to achieve performance results that are comparable to current high-performance systems. For further investigations, front-end processing would require the most

attention, because it seems to have the greatest potential for providing a big improvement. Special consideration should be given to apply short-time Gaussianisation as an alternative to CMS. It would also be a good idea to give more attention to noise contamination.

Memory requirements became an important concern when the number of T-Norm impostors were increased. For a proper investigation, or if 2048-component models are required, a better memory management strategy must be implemented. It would also help to reduce the memory overhead of models as much as possible.



Bibliography

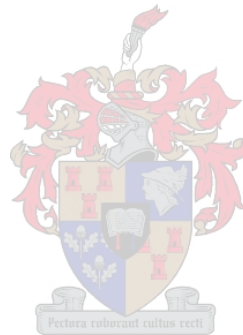
- [1] Reynolds, D.A.: An Overview of Automatic Speaker Recognition Technology. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '02)*, vol. 4, pp. 4072–4075, 2002.
- [2] Reynolds, D.A. & Rose, R.C.: Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models. *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 1, pp. 72–83, 1995.
- [3] Duda, R.O., Hart, P.E. & Stork, D.G.: *Pattern Classification*. 2nd edn. Wiley, 2001.
- [4] Alder, M.D.: *Principles of Pattern Classification: Statistical, Neural Net and Syntactic Methods of Getting Robots to See and Hear*. University of Western Australia Centre for Intelligent Information Processing Systems, 1994.
- [5] Gauvain, J.-L. & Lee, C.-H.: Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains. *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 2, pp. 291–298, 1994.
- [6] Reynolds, D.A., Quatieri, T.F. & Dunn, R.B.: Speaker Verification Using Adapted Gaussian Mixture Models. *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19–41, 2000.
- [7] Watanabe, T., Shinoda, K., Takagi, K. & Iso, K.-I.: High Speed Speech Recognition using Tree-structured Probability Function. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '95)*, vol. 1, pp. 556–559, 1995.
- [8] Shinoda, K. & Lee, C.-H.: A Structural Bayes Approach to Speaker Adaptation. *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 3, pp. 276–287, 2001.

- [9] Liu, M., Chang, E. & Dai, B.: Hierarchical Gaussian Mixture Model for Speaker Verification. *7th International Conference on Spoken Language Processing (ICSLP 2002)*, pp. 1353–1356, 2002.
- [10] Xiang, B. & Berger, T.: Efficient Text-Independent Speaker Verification with Structural Gaussian Mixture Models and Neural Network. *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 5, pp. 447–456, 2003.
- [11] Rosenberg, A.E. & Sambur, M.R.: New Techniques for Automatic Speaker Verification. *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 23, no. 2, pp. 169–176, 1975.
- [12] Freedman, D., Pisani, R. & Purves, R.: *Statistics*. W. W. Norton & Company, 1978.
- [13] Peebles Jr, P.Z.: *Probability, Random Variables and Random Signal Principles*. 4th edn. McGraw-Hill, 2001.
- [14] Rabiner, L. & Schafer, R.W.: *Digital Processing of Speech Signals*. Prentice-Hall, 1978.
- [15] Smith, S.W.: *The Scientist and Engineer's Guide to Digital Signal Processing*. 2nd edn. California Technical Publishing, 1999.
- [16] Proakis, J.G. & Manolakis, D.G.: *Digital Signal Processing: Principles, Algorithms, and Applications*. 3rd edn. Prentice-Hall, 1996.
- [17] Atal, B.: Automatic Recognition of Speakers from their Voices. *Proceedings of the IEEE*, vol. 64, no. 4, pp. 460–475, 1976.
- [18] Davis, S.B. & Mermelstein, P.: Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-28, no. 5, pp. 357–366, 1980.
- [19] Furui, S.: Cepstral Analysis Technique for Automatic Speaker Verification. *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-29, no. 2, pp. 254–272, 1981.
- [20] Tierney, J.: A Study of LPC Analysis of Speech in Additive Noise. *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-28, no. 4, pp. 389–397, 1980.

- [21] Ramaswamy, G.N., Navrátil, J., Chaudhari, U.V. & Zilca, R.D.: The IBM System for the NIST-2002 Cellular Speaker Verification Evaluation. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '03)*, vol. 2, pp. 61–64, 2003.
- [22] Childers, D.G., Skinner, D.P. & Kemerait, R.C.: The Cepstrum: A Guide to Processing. *Proceedings of the IEEE*, vol. 65, no. 10, pp. 1428–1443, 1977.
- [23] Mammone, R.J., Zang, X. & Pamachandran, R.P.: Robust Speaker Recognition. *Signal Processing Magazine*, pp. 47–60, September 1996.
- [24] da Silva, D.G., Apolinário, J.A. & de Lima, C.B.: On the Effect of the Language in CMS Channel Normalization. *International Telecommunications Symposium*, September 2002.
- [25] Xiang, B., Chaudhari, U.V., Navrátil, J., Ramaswamy, G.N. & Gopinath, R.A.: Short-time Gaussianization for Robust Speaker Verification. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '02)*, vol. 1, pp. I-681–I-684, 2002.
- [26] Liu, C.-S., Wang, H.-C. & Lee, C.-H.: Speaker Verification Using Normalized Log-Likelihood Score. *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 1, pp. 56–60, 1996.
- [27] Rosenberg, A. & Parthasarathy, S.: Speaker Background Models for Connected Digit Password Speaker Verification. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '96)*, vol. 1, pp. 81–84, 1996.
- [28] Auckenthaler, R., Carey, M. & Lloyd-Thomas, H.: Score Normalization for Text-Independent Speaker Verification Systems. *Digital Signal Processing*, vol. 10, no. 1-3, pp. 42–54, 2000.
- [29] Navrátil, J. & Ramaswamy, G.N.: The Awe and Mystery of T-Norm. *Proceedings of the 8th European Conference on Speech Communication and Technology (EUROSPEECH)*, pp. 2009–2012, 2003.
- [30] Martin, A., Doddington, T.K., Ordowski, M. & Przybocki, M.: The DET Curve in Assessment of Detection Task Performance. *Proceedings of the 5th European Conference on Speech Communication and Technology (EUROSPEECH)*, pp. 1895–1898, 1997.

- [31] Gillick, L. & Cox, S.J.: Some Statistical Issues in the Comparison of Speech Recognition Algorithms. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '89)*, vol. 1, pp. 532–535, 1989.
- [32] Burton, D.K.: Text-Dependent Speaker Verification Using Vector Quantization Source Coding. *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-35, no. 2, pp. 133–143, 1987.
- [33] Naik, J.M., Netsch, L.P. & Doddington, G.R.: Speaker Verification Over Long Distance Telephone Lines. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '89)*, vol. 1, pp. 524–527, 1989.
- [34] Savic, M. & Gupta, S.K.: Variable parameter speaker verification system based on hidden Markov modeling. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '90)*, vol. 1, pp. 281–284, 1990.
- [35] Tishby, N.Z.: On the Application of Mixture AR Hidden Markov Models to Text Independent Speaker Recognition. *IEEE Transactions on Signal Processing*, vol. 39, no. 3, pp. 563–570, 1991.
- [36] Carey, M.J., Parris, E.S., Bennett, S.J. & Lloyd-Thomas, H.: A Comparison of Model Estimation Techniques for Speaker Verification. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '97)*, vol. 2, pp. 1083–1086, 1997.
- [37] Farrell, K.R., Ramachandran, R.P. & Mammone, R.J.: An Analysis of Data Fusion Methods for Speaker Verification. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '98)*, vol. 2, pp. 1129–1132, 1998.
- [38] Moon, T.K.: The Expectation-Maximization Algorithm. *Signal Processing Magazine*, pp. 47–60, November 1996.
- [39] Bilmes, J.A.: A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Tech. Rep. TR-97-021, International Computer Science Institute, 1947 Center St., Suite 600, Berkeley, California, 94704-1198, April 1997. Available at: <http://www.icsi.berkeley.edu/techreports/1997.html> (November 2004).

- [40] du Preez, J.A.: Thesis-related discussions. Stellenbosch University.
- [41] Goodrich, M.T. & Tamassia, R.: *Data Structures and Algorithms in Java*. 2nd edn. Wiley, 2001.
- [42] Niesler, T.: DSP 823 course notes. Unpublished, 2001. Stellenbosch University E&E Engineering Department.
- [43] Martin, A.: The NIST Year 2004 Speaker Recognition Evaluation Plan. 2004. Available at: <http://www.nist.gov/speech/tests/spk/2004/>.
- [44] du Toit, I.: Non-acoustic Speaker Recognition. Master's thesis, 2004. Stellenbosch University.



Appendix A

Derivations and Justifications

A.1 The Likelihood Ratio and Test Normalisation

This section shows that it is equivalent to use log-likelihoods instead of log-likelihood ratios for normalisation by T-Norm. Start with the log-likelihood ratio (the score)

$$s_0 = \ln p(\mathbf{X}|H_0) - \ln p(\mathbf{X}|H_1). \quad (\text{A.1})$$

H_0 is the hypothesis that the utterance \mathbf{X} was created by the target speaker S_0 and H_1 is the hypothesis that \mathbf{X} was created by an impostor. For T-Norm, N additional hypotheses H_n state that \mathbf{X} was created by some non-target speaker S_n . This results in the non-target scores

$$s_n = \ln p(\mathbf{X}|H_n) - \ln p(\mathbf{X}|H_1),$$

where $n = 2, \dots, N + 1$. Using maximum-likelihood estimation, the mean μ_I of the non-target scores can be estimated by

$$\begin{aligned} \mu_I &= \frac{1}{N} \sum_{n=1}^N s_n \\ &= \frac{1}{N} \sum_{n=1}^N (\ln p(\mathbf{X}|H_n) - \ln p(\mathbf{X}|H_1)) \\ &= \frac{1}{N} \left(\sum_{n=1}^N \ln p(\mathbf{X}|H_n) - N \ln p(\mathbf{X}|H_1) \right) \\ &= \frac{1}{N} \sum_{n=1}^N \ln p(\mathbf{X}|H_n) - \ln p(\mathbf{X}|H_1). \end{aligned} \quad (\text{A.2})$$

The variance σ_I^2 of the non-target scores can in the same way be estimated by

$$\begin{aligned}
\sigma_I^2 &= \frac{1}{N} \sum_{n=1}^N (s_n - \mu_I)^2 \\
&= \frac{1}{N} \sum_{n=1}^N \left(\ln p(\mathbf{X}|H_n) - \ln p(\mathbf{X}|H_1) - \frac{1}{N} \sum_{n=1}^N \ln p(\mathbf{X}|H_n) + \ln p(\mathbf{X}|H_1) \right)^2 \\
&= \frac{1}{N} \sum_{n=1}^N \left(\ln p(\mathbf{X}|H_n) - \frac{1}{N} \sum_{n=1}^N \ln p(\mathbf{X}|H_n) \right)^2. \tag{A.3}
\end{aligned}$$

Let

$$\mu_{IL} = \frac{1}{N} \sum_{n=1}^N \ln p(\mathbf{X}|H_n)$$

be the mean estimate of the non-target log-likelihoods so that

$$\mu_I = \mu_{IL} - \ln p(\mathbf{X}|H_1).$$

Then, eq. (A.3) becomes

$$\sigma_I^2 = \frac{1}{N} \sum_{n=1}^N (\ln p(\mathbf{X}|H_n) - \mu_{IL})^2 = \sigma_{IL}^2, \tag{A.4}$$

which is simply the variance estimate of the non-target log-likelihoods.

Substitution of eq. (A.1), eq. (A.2) and the square root of eq. (A.4) into eq. (5.7) results in the normalised score

$$\begin{aligned}
s' &= \frac{s_0 - \mu_I}{\sigma_I} \\
&= \frac{\ln p(\mathbf{X}|H_0) - \ln p(\mathbf{X}|H_1) - \frac{1}{N} \sum_{n=1}^N \ln p(\mathbf{X}|H_n) + \ln p(\mathbf{X}|H_1)}{\sigma_I} \\
&= \frac{\ln p(\mathbf{X}|H_0) - \frac{1}{N} \sum_{n=1}^N \ln p(\mathbf{X}|H_n)}{\sigma_I} \\
&= \frac{\ln p(\mathbf{X}|H_0) - \mu_{IL}}{\sigma_{IL}}.
\end{aligned}$$

This shows that by using only the log-likelihoods instead of the log-likelihood ratios, the exact same resulting score is obtained.

A.2 Intuitive Derivation of MAP Formulas

The formal derivation of the MAP re-estimation formulas is quite involved. But, during this research, a much more compact, intuitive derivation was produced based on how MAP adaptation behaves. The derivation makes use of the on-line re-estimation formulas, the derivation of which is shown in Section A.3.

Look at all the parts of the ML re-estimation formulas. There is a sequence of N observed training feature vectors \mathbf{x}_n . To calculate the ML mean estimate $\tilde{\boldsymbol{\mu}}_k$ of mixture component k , this sequence is passed to the formula:

$$\tilde{\boldsymbol{\mu}}_k = \frac{\sum_{n=1}^N c_{kn} \mathbf{x}_n}{c_k},$$

where c_{kn} is the responsibility of mixture component k for vector \mathbf{x}_n , or

$$c_{kn} = p(k|\mathbf{x}_n, \lambda^{[i]}) = \frac{w_k^{[i]} p(\mathbf{x}_n | \lambda_k^{[i]})}{\sum_{j=1}^K w_j^{[i]} p(\mathbf{x}_n | \lambda_j^{[i]})}$$

and

$$c_k = \sum_{n=1}^N c_{kn}$$

can be called the probability count for the N training feature vectors. The summation of the training vectors can be represented by the symbol S , so that

$$S = \sum_{n=1}^N c_{kn} \mathbf{x}_n$$

and hence $\tilde{\boldsymbol{\mu}}_k = S/c_k$. To calculate the corresponding ML estimate for the covariance matrix $\tilde{\Sigma}_k$, the following on-line estimation formula can be used

$$\tilde{\Sigma}_k = \frac{\sum_{n=1}^N c_{kn} \mathbf{x}_n \mathbf{x}_n^T}{c_k} - \tilde{\boldsymbol{\mu}}_k \tilde{\boldsymbol{\mu}}_k^T.$$

Here, the symbol R can be defined, so that

$$R = \sum_{n=1}^N c_{kn} \mathbf{x}_n \mathbf{x}_n^T$$

and hence $\tilde{\Sigma}_k = R/c_k - \tilde{\boldsymbol{\mu}}_k \tilde{\boldsymbol{\mu}}_k^T$.

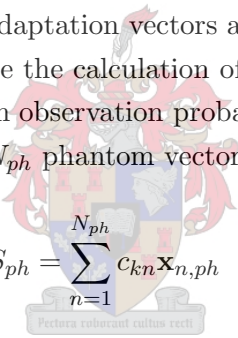
The derivation presented here is based on these points:

- MAP uses prior PDF parameters from which new parameters are adapted,
- the prior parameters are multiplied by a relevance factor to determine the magnitude of its influence, and
- the MAP re-estimation look very similar to the ML re-estimation formulas.

This derivation supposes that the relevance factor acts like a phantom observation probability count. This represents any number of phantom (or imaginary) training feature vectors. The idea is that these phantom vectors can be placed in a sequence followed by the actual adaptation (training) vectors and passed to the on-line ML estimation formulas to produce the final model parameters.

The phantom vectors will be passed to the on-line estimation formulas to produce an intermediate set of parameters, which is defined to be the prior PDF parameters. The actual adaptation vectors are then passed to the on-line estimation formulas to complete the calculation of the final model parameters.

Define τ_k to be the phantom observation probability count of mixture component k for some number of N_{ph} phantom vectors $\mathbf{x}_{n,ph}$. Let S_{ph} be



$$S_{ph} = \sum_{n=1}^{N_{ph}} c_{kn} \mathbf{x}_{n,ph}$$

so that the prior mean is calculated as $\boldsymbol{\mu}_{kp} = S_{ph}/\tau_k$. Note that $\boldsymbol{\mu}_{kp}$ and τ_k are known or chosen values and S_{ph} is unknown. For the covariance matrix, let R_{ph} be

$$R_{ph} = \sum_{n=1}^N c_{kn} \mathbf{x}_{n,ph} \mathbf{x}_{n,ph}^T$$

so that the prior covariance matrix is calculated as $\Sigma_{kp} = R_{ph}/\tau_k - \boldsymbol{\mu}_{kp} \boldsymbol{\mu}_{kp}^T$. Here also R_{ph} is the only unknown.

Now, the MAP estimate is simply the combination of the ML estimate that produced the prior parameters and the ML estimate of the adaptation data. In other words, the MAP adapted mean is

$$\hat{\boldsymbol{\mu}}_k = \frac{S_{ph} + S}{\tau_k + c_k} = \frac{\tau_k \boldsymbol{\mu}_{kp} + \sum_{n=1}^N c_{kn} \mathbf{x}_n}{\tau_k + c_k}.$$

This is exactly the same formula that is obtained by the formal derivation. The summation of the probability counts τ_k and c_k is simply the probability count of

the combined sequence of phantom and actual vectors. S_{ph} is written in terms of the known (or chosen) variables. Note here that the combination of S_{ph} and S can now be written as

$$\begin{aligned} S_{ph} + S &= (\tau_k + c_k) \hat{\boldsymbol{\mu}}_k \\ \tau_k \boldsymbol{\mu}_{kp} + c_k \tilde{\boldsymbol{\mu}}_k &= (\tau_k + c_k) \hat{\boldsymbol{\mu}}_k. \end{aligned}$$

This form will be used for deriving the MAP estimation formula for covariance matrices.

The MAP adapted covariance matrix is

$$\begin{aligned} \hat{\Sigma}_k &= \frac{R_{ph} + R}{\tau_k + c_k} - \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T \\ &= \frac{R_{ph} + R - (\tau_k + c_k) \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T}{\tau_k + c_k} \\ &= \frac{R_{ph} + R - (\tau_k + c_k) \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T + -(\tau_k + c_k) \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T + (\tau_k + c_k) \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T}{\tau_k + c_k} \\ &= \frac{R_{ph} + R - (\tau_k \boldsymbol{\mu}_{kp} + c_k \tilde{\boldsymbol{\mu}}_k) \hat{\boldsymbol{\mu}}_k^T - \hat{\boldsymbol{\mu}}_k (\tau_k \boldsymbol{\mu}_{kp} + c_k \tilde{\boldsymbol{\mu}}_k)^T + (\tau_k + c_k) \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T}{\tau_k + c_k} \\ &= (R_{ph} + R - \tau_k \boldsymbol{\mu}_{kp} \hat{\boldsymbol{\mu}}_k^T - c_k \tilde{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T - \tau_k \hat{\boldsymbol{\mu}}_k \boldsymbol{\mu}_{kp}^T - c_k \hat{\boldsymbol{\mu}}_k \tilde{\boldsymbol{\mu}}_k^T + \tau_k \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T + \\ &\quad + c_k \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T) (\tau_k + c_k)^{-1} \\ &= \left(R_{ph} + \sum_{n=1}^N c_{kn} \mathbf{x}_n \mathbf{x}_n^T - \tau_k \boldsymbol{\mu}_{kp} \hat{\boldsymbol{\mu}}_k^T - \sum_{n=1}^N c_{kn} \mathbf{x}_n \hat{\boldsymbol{\mu}}_k^T - \tau_k \hat{\boldsymbol{\mu}}_k \boldsymbol{\mu}_{kp}^T - \right. \\ &\quad \left. - \sum_{n=1}^N c_{kn} \hat{\boldsymbol{\mu}}_k \mathbf{x}_n^T + \tau_k \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T + \sum_{n=1}^N c_{kn} \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T \right) (\tau_k + c_k)^{-1} \\ &= \left(R_{ph} - \tau_k \boldsymbol{\mu}_{kp} \boldsymbol{\mu}_{kp}^T + \sum_{n=1}^N c_{kn} (\mathbf{x}_n \mathbf{x}_n^T - \mathbf{x}_n \hat{\boldsymbol{\mu}}_k^T - \hat{\boldsymbol{\mu}}_k \mathbf{x}_n^T + \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T) + \right. \\ &\quad \left. + \tau_k \boldsymbol{\mu}_{kp} \hat{\boldsymbol{\mu}}_k^T - \tau_k \hat{\boldsymbol{\mu}}_k \boldsymbol{\mu}_{kp}^T - \tau_k \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T + \tau_k \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T \right) (\tau_k + c_k)^{-1} \\ &= \left((R_{ph} - \tau_k \boldsymbol{\mu}_{kp} \boldsymbol{\mu}_{kp}^T) + \sum_{n=1}^N c_{kn} (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k) (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k)^T + \right. \\ &\quad \left. + \tau_k (\boldsymbol{\mu}_{kp} \hat{\boldsymbol{\mu}}_k^T - \hat{\boldsymbol{\mu}}_k \boldsymbol{\mu}_{kp}^T + \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T) \right) (\tau_k + c_k)^{-1} \\ &= \frac{\tau_k \Sigma_{kp} + \sum_{n=1}^N c_{kn} (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k) (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k)^T + \tau_k (\boldsymbol{\mu}_{kp} - \hat{\boldsymbol{\mu}}_k) (\boldsymbol{\mu}_{kp} - \hat{\boldsymbol{\mu}}_k)^T}{\tau_k + c_k} \end{aligned}$$

This is the same formula that is obtained by formal derivation.

A.3 On-line Re-estimation Formulas

In some cases it is desirable to obtain new estimates for model parameters as soon as any given training feature vector is available. This is called on-line training. For off-line training, all training vectors must be available at the same time. The off-line re-estimation formulas given in Section 3.5.2 can be converted to be suitable for on-line training. First, look at the component responsibilities:

$$p(k|\mathbf{x}_n, \lambda^{[i]}) = \frac{w_k^{[i]} p(\mathbf{x}_n | \lambda_k^{[i]})}{\sum_{m=1}^K w_m^{[i]} p(\mathbf{x}_n | \lambda_m^{[i]})}. \quad (\text{A.5})$$

This formula does not depend on more than one feature vector and does not need to be converted. Secondly, consider the weights:

$$w_k^{[i+1]} = \frac{1}{N} \sum_{n=1}^N c_{kn}. \quad (\text{A.6})$$

This formula is only dependant on the number of observed feature vectors. The summation can be incremented with each new training vector. Again, no conversion is needed here. The same applies to the formula for updating the means:

$$\boldsymbol{\mu}_k^{[i+1]} = \frac{\sum_{n=1}^N c_{kn} \mathbf{x}_n}{\sum_{n=1}^N c_{kn}}. \quad (\text{A.7})$$

When looking at the covariance matrices:

$$\Sigma_k^{[i+1]} = \frac{\sum_{n=1}^N c_{kn} (\mathbf{x}_n - \boldsymbol{\mu}_k^{[i+1]}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{[i+1]})^T}{\sum_{n=1}^N c_{kn}} \quad (\text{A.8})$$

it can be observed that the fully updated mean estimate is required to complete this calculation. However, the updated mean requires that all training vectors must have been observed already. This formula needs to be converted so that it is only dependant on training vectors that are available. When multiplying

the factors in the numerator, the following derivation can be made:

$$\begin{aligned}
\Sigma_k^{[i+1]} &= \frac{1}{\sum_{n=1}^N c_{kn}} \left(\sum_{n=1}^N c_{kn} \left(\mathbf{x}_n \mathbf{x}_n^T - \mathbf{x}_n \boldsymbol{\mu}_k^{T[i+1]} - \boldsymbol{\mu}_k^{[i+1]} \mathbf{x}_n^T + \boldsymbol{\mu}_k^{[i+1]} \boldsymbol{\mu}_k^{T[i+1]} \right) \right) \\
&= \frac{1}{\sum_{n=1}^N c_{kn}} \left(\sum_{n=1}^N c_{kn} \mathbf{x}_n \mathbf{x}_n^T - \sum_{n=1}^N c_{kn} \mathbf{x}_n \boldsymbol{\mu}_k^{T[i+1]} - \right. \\
&\quad \left. - \sum_{n=1}^N c_{kn} \boldsymbol{\mu}_k^{[i+1]} \mathbf{x}_n^T + \sum_{n=1}^N c_{kn} \boldsymbol{\mu}_k^{[i+1]} \boldsymbol{\mu}_k^{T[i+1]} \right) \\
&= \frac{1}{\sum_{n=1}^N c_{kn}} \left(\sum_{n=1}^N c_{kn} \mathbf{x}_n \mathbf{x}_n^T - \left(\sum_{n=1}^N c_{kn} \mathbf{x}_n \right) \boldsymbol{\mu}_k^{T[i+1]} - \right. \\
&\quad \left. - \boldsymbol{\mu}_k^{[i+1]} \left(\sum_{n=1}^N c_{kn} \mathbf{x}_n^T \right) + \left(\sum_{n=1}^N c_{kn} \right) \boldsymbol{\mu}_k^{[i+1]} \boldsymbol{\mu}_k^{T[i+1]} \right) \\
&= \frac{\sum_{n=1}^N c_{kn} \mathbf{x}_n \mathbf{x}_n^T}{\sum_{n=1}^N c_{kn}} - \left(\frac{\sum_{n=1}^N c_{kn} \mathbf{x}_n}{\sum_{n=1}^N c_{kn}} \right) \boldsymbol{\mu}_k^{T[i+1]} - \\
&\quad - \boldsymbol{\mu}_k^{[i+1]} \left(\frac{\sum_{n=1}^N c_{kn} \mathbf{x}_n^T}{\sum_{n=1}^N c_{kn}} \right) + \frac{\sum_{n=1}^N c_{kn}}{\sum_{n=1}^N c_{kn}} \boldsymbol{\mu}_k^{[i+1]} \boldsymbol{\mu}_k^{T[i+1]} \\
&= \frac{\sum_{n=1}^N c_{kn} \mathbf{x}_n \mathbf{x}_n^T}{\sum_{n=1}^N c_{kn}} - \boldsymbol{\mu}_k^{[i+1]} \boldsymbol{\mu}_k^{T[i+1]} - \boldsymbol{\mu}_k^{[i+1]} \boldsymbol{\mu}_k^{T[i+1]} + \boldsymbol{\mu}_k^{[i+1]} \boldsymbol{\mu}_k^{T[i+1]} \\
&= \frac{\sum_{n=1}^N c_{kn} \mathbf{x}_n \mathbf{x}_n^T}{\sum_{n=1}^N c_{kn}} - \boldsymbol{\mu}_k^{[i+1]} \boldsymbol{\mu}_k^{T[i+1]}.
\end{aligned}$$

This form of the equation is only dependant on the available training vectors. As each new training vector is added, a new mean estimate can be made to complete this calculation. Once all the training vectors are included, the estimates are exactly the same as those obtained with off-line estimation. In effect, only the statistics $\sum_{n=1}^N c_{kn}$, $\sum_{n=1}^N c_{kn} \mathbf{x}_n$ and $\sum_{n=1}^N c_{kn} \mathbf{x}_n \mathbf{x}_n^T$ need to be accumulated as training vectors are added. The final parameter estimates then needs to be calculated only when all the training vectors have been processed. Intermediate estimates can be calculated at any time, though.