HIDDEN MARKOV MODELS AND ARTIFICIAL NEURAL NETWORKS FOR SPEECH AND SPEAKER RECOGNITION

THÈSE N° (1998)

PRÉSENTÉE AU DÉPARTEMENT D'ÉLECTRICITÉ

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR 'ES SCIENCES TECHNIQUES

PAR

JEAN HENNEBERT

Ingénieur Civil Electricien
diplômé de la Faculté Polytechnique de Mons,
de nationalité belge

presentée au jury:

Prof. Martin Hasler, directeur de thèse
Prof. Hervé Bourlard, corapporteur
Prof. Christian Wellekens, corapporteur
Prof. Frédéric Bimbot, corapporteur

Lausanne, EPFL
1998

Hidden Markov Models and Artificial Neural Networks for Speech and Speaker Recognition

October 16, 1998

I believe that the first objective of a thesis is to learn how to do research, and this implies working by yourself, developping your own ideas and discovering your mistakes.

I also believe that doing research is trying and trying again to open new doors. If you do not close this door, if you discover a wall behind it or if you don't explore entirely the room thus unveiled, it does not matter. The important, vital gesture, is to open the door.

Jean Hennebert, October 16, 1998

Version abrégée

Cette thèse aborde l'étude de systèmes de reconnaissance de la parole et du locuteur dans lesquels des réseaux de neurones artificiels (ANN) sont utilisés à la place d'outils de modélisation classiques tels que certaines composantes des modèles de Markov cachés (HMM).

Trois approches sont analysées. La première est originale et concerne l'utilisation des cartes auto-organisatrices de Kohonen en combinaison avec des HMMs discrets pour la reconnaissance de mots isolés. La deuxième concerne les systèmes hybrides ANN/HMM pour la reconnaissance de la parole. Le système proposé n'est pas original en soi et a déjà fait l'objet de nombreuses publications. Néanmoins, un nouveau formalisme théorique est avancé, ainsi qu'une nouvelle façon d'entraîner le système. La dernière partie est quand à elle originale et concerne la mise au point d'une nouvelle approche de vérification du locuteur basée sur un concept segmental.

D'une manière plus détaillée, ce rapport de thèse est organisé de la façon suivante :

Chapitre 1: Introduction Une vue d'ensemble des différentes techniques de reconnaissance de la parole et du locuteur est présentée. Une classification des différentes applications de ces systèmes est donnée, de telle sorte à positionner ce travail dans son contexte. Quelques applications commerciales sont également mentionnées.

Chapitre 2: Fondements théoriques Ce chapitre décrit de façon claire la théorie nécessaire à la compréhension des systèmes de reconnaissance de parole et du locuteur. Classification statistique, paramétrisation, modèles de Markov cachés et réseaux de neurones artificiels sont introduits.

Chapitre 3: Cartes auto-organisatrices et HMM discrets. Le quantificateur vectoriel d'un HMM discret, au lieu d'être entraîné par un algorithme qui minimise sa distorsion, est entaîné à l'aide de réseaux de neurones artificiels non-supervisés : les cartes auto-organisatrices de Kohonen (SOM). L'approche SOM est comparée avec deux algorithmes classiques : LBG et k-means. Une analyse détaillée du fonctionnement de ces trois algorithmes permet d'avoir une meilleure compréhension de leurs fonctionnalités. Des résultats experimentaux sont reportés sur une tâche de reconnaissance de mots isolés où l'approche SOM obtient de meilleures performances de manière significative et consistante par rapport aux autres algorithmes.

Chapitre 4: Réseaux de neurones multi-couches et HMMs continus. Les HMMs continus utilisent de manière générale une modélisation multigaussienne des densités de probabiliés. L'idée est d'utiliser des réseaux de neurones multi-couches à la place des multigaussiennes. Ce genre de systèmes, dénommés "hybrides" dans la littérature, ont déjà été étudiés lors de nombreux travaux. L'analyse effectuée dans ce chapitre, en plus de valider les conclusions des travaux antérieurs, propose un nouveau formalisme théorique pour décrire le comportement de ces systèmes. A la lumière de ce développement théorique, un nouvel algorithme d'entraînement du réseau de neurones multi-couches est proposé. Cet algoritme, dérivé de l'entraînement "avant-arrière" des HMMs est analysé et comparé à la méthode d'entraînement classique sur une tâche de reconnaissance de mots isolés.

Chapitre 5: Une approche segmentale pour la vérication du locuteur indépendente du texte. Une nouvelle approche segmentale est proposée comme alternative aux approches globales de vérification du locuteur indépendente du texte. L'idée est de retrouver les avantages des techniques dépendentes du texte en segmentant automatiquement le signal de parole en unités

acoustiques élémentaires. La modélisation du locuteur se fait alors individuellement sur ces unités qui sont définies dans des régions plus restreintes de l'espace d'entrée. Le système ainsi obtenu est moins sensibles aux variabilités des conditions de test et d'entraînement. Un système segmental basé sur la décomposition temporelle, la quantification vectorielle et les réseaux de neurones discriminants multi-couches est construit, analysé et évalué sur une base de données standard. L'approche segmentale donne effectivement de meilleures performances que les approches globales dans le cas où les conditions de test et d'entraînement sont différentes.

Chapitre 6: Conclusions. Des conclusions générales sur les diffŕents systèmes présentés dans ce travail sont données.

Abstract

In this thesis, we are concerned with the two fields of automatic speech recognition (ASR) and automatic speaker recognition (ASkR) in telephony. More precisely, we are interested in systems based on hidden Markov models (HMMs) in which artificial neural networks (ANNs) are used in place of more classical tools.

This work is dedicated to the analysis of three approaches. The first one, mainly original, concerns the use of Self-Organizing Maps in discrete HMMs for isolated word speech recognition. The second approach concerns continuous hybrid HMM/ANN systems, extensively studied in previous research work. The system is not original in its form but its analysis permitted to bring a new theoretical framework and to introduce some extensions regarding the way the system is trained. The last part concerns the implementation of a new ANN segmental approach for text-independent speaker verification.

In a more detailed way, this thesis report is organized as follows :

Chapter 1: Introduction This introductory chapter gives a global view of the ASR, ASkR and ANNs fields. A classification of the tasks is given in order to place, in their context, problems that are addressed in this work . A range of commercial applications is also given.

Chapter 2: Fundamentals Chapter 2 is dedicated to the description of fundamentals needed to understand how speech and speaker recognition systems work. Statistical pattern recognition, feature extraction, hidden Markov models and neural networks are introduced.

Chapter 3: Self-Organizing Maps and Discrete HMMs. The vector quantizer, instead of being designed with a classical minimum distortion algorithm, is designed with an unsupervised self-organizing neural network: Kohonen Self-Organizing Map (SOM). SOM is compared with two baseline algorithms, namely LBG and k-means. A theoretical analysis of these algorithms will give us a better understanding of their differences. Experiments are reported on an isolated word tasks in which the SOM approach gives performance improvements when compared to the other algorithms.

Chapter 4: Feedforward ANNs and Continuous HMMs. Instead of using multi-Gaussian probability density function estimators, a feedforward ANN is used to generate local hypotheses, leading to the so-called **hybrid HMM-ANN systems**. Many studies have already been done in this direction in which advantages of hybrid systems have been underlined. In addition to the validation of this approach, a new theoretical formalism which describes the behaviour of hybrid systems is presented. In the light of this formalism, a new algorithm to train the ANN is proposed. The algorithm is derived from the forward-backward training algorithm and presents advantages when few training data are available.

Chapter 5: A segmental approach for speaker verification. A segmental approach is proposed as an alternative to global modeling techniques as GMM. The general idea of the segmental approach is to recover the advantages of text-dependent techniques through an automatic segmentation of the speech material into sub-word units. Speakers are then modeled on these units that are more precise in the acoustic vector space and less sensitive to variabilities. A segmental system based on temporal decomposition, automatic clustering and a set of discriminant ANNs is proposed, analysed and evaluated on a standard speaker verification task. When used on a standard speaker

verification task, the segmental approach shows more robustness than global approaches, in the case of mismatched conditions between training and testing.

Chapter 6: Conclusions. General conclusions on the different systems investigated in this work are drawn.

Contents

Version abrégée iii

Abstract v

Acknowledgements xi

1 Introduction 1
- 1.1 Outline of the report . . . 2
- 1.2 Automatic Speech Recognition . . . 2
  - 1.2.1 Task Classification . . . 3
  - 1.2.2 Applications . . . 4
- 1.3 Automatic Speaker Recognition . . . 5
  - 1.3.1 Task classification . . . 5
  - 1.3.2 Applications . . . 6
- 1.4 Artificial Neural Networks . . . 6
  - 1.4.1 Classifications . . . 7
  - 1.4.2 Applications . . . 7

Bibliography 9

2 Fundamentals 11
- 2.1 Speech Signal . . . 12
- 2.2 Statistical Pattern Recognition . . . 14
- 2.3 Feature extraction . . . 15
- 2.4 Hidden Markov Models . . . 20
  - 2.4.1 HMMs hypothesis . . . 21
  - 2.4.2 Forward and backward recursions . . . 24
  - 2.4.3 Parameter reestimation . . . 26
  - 2.4.4 Viterbi criterion . . . 27
  - 2.4.5 Probability density functions . . . 27
- 2.5 Multilayer Perceptrons . . . 29
  - 2.5.1 Propagation . . . 31
  - 2.5.2 Backpropagation . . . 31
  - 2.5.3 Posterior probability estimation . . . 34
  - 2.5.4 Input Scaling . . . 34
  - 2.5.5 Weight and bias initialization . . . 35

Bibliography 38

3 Self-Organizing Maps and Discrete HMMs **39**
3.1 System description . . . 40
3.1.1 Vector quantization . . . 40
3.1.2 Lloyd Algorithms . . . 41
3.1.3 Self-Organizing Maps of Kohonen . . . 42
3.1.4 Discrete HMMs . . . 44
3.2 Experiments . . . 44
3.2.1 HIM Database . . . 45
3.2.2 HER database . . . 47
3.3 Discussion and extensions of the system . . . 47
3.3.1 Fast nearest neighbour search . . . 47
3.3.2 Supervised VQ . . . 48
3.3.3 Topological information of SOM . . . 50
3.4 Conclusion . . . 50

Bibliography **53**

4 Feedforward ANNs and Continuous HMMs **55**
4.1 Original Viterbi trained HMM/ANN system . . . 56
4.1.1 Motivations . . . 56
4.1.2 Viterbi decoding and training . . . 56
4.1.3 System advantages . . . 57
4.1.4 System drawbacks . . . 59
4.2 New formulation . . . 59
4.2.1 Estimation of Global Posteriors . . . 60
4.2.2 Discussion and system extension . . . 63
4.3 Experiments . . . 65
4.3.1 HER database . . . 65
4.3.2 Phonebook database . . . 66
4.4 Conclusions . . . 71

Bibliography **73**

5 A segmental approach for speaker verification **75**
5.1 Baseline text-independent speaker verification . . . 76
5.1.1 Pattern classification . . . 77
5.1.2 Thresholding . . . 78
5.2 Preliminary experiments . . . 80
5.2.1 System description . . . 80
5.2.2 Experimental setup . . . 81
5.2.3 Results . . . 82
5.2.4 Discussion . . . 85
5.3 The segmental system . . . 85
5.4 Experiences and results . . . 88
5.4.1 Task description . . . 88
5.4.2 Global system results . . . 89
5.4.3 Segmental system results . . . 89
5.5 Conclusions . . . 95
5.5.1 Text-dependent preliminary experiments . . . 95
5.5.2 Text-independent segmental system . . . 95

Bibliography **98**

6 Conclusions **99**
6.1 Self-Organizing Maps and Discrete HMMs . . . 99
6.2 Feedforward ANNs and Continuous HMMs . . . 100
6.3 Segmental approach for speaker verification . . . 101

A Databases **105**
A.1 Phonebook . . . 105
A.1.1 Common features to all the experiments . . . 105
A.2 HER . . . 107
A.3 HIM . . . 107
A.4 NIST'1998 : Switchboard . . . 107
A.4.1 Training set . . . 109
A.4.2 Test set . . . 109

B Backpropagation and Kolmogorov's theorem **111**
B.1 Stochastic backpropagation algorithm . . . 111
B.2 MLP inversion . . . 113
B.3 MLP and Bayes theory . . . 114
B.4 Kolmogorov's theorem . . . 115

C Review of other ANN approaches for ASR and ASkR **117**
C.1 Dumb multilayer perceptron for isolated word recognition . . . 117
C.2 Time-delay neural network for recognition . . . 118
C.3 Hidden control neural network for recognition . . . 119

Bibliography **123**

General bibliography **136**

Index **140**

Curriculum Vitæ **141**

List of publications **143**

Acknowledgements

This thesis was conducted in the Electricity Department of the Swiss Federal Institute of Technology, Lausanne and is the achievement of three years of research in the area of speech processing and neural networks. I am greatly indebted to my thesis supervisor Prof. Martin Hasler for accepting me in his stimulating Circuits and Systems group, for the material support he provided me and for his patient guidance in various domains of research and scientific writing. The trust and freedom he gave me, made me love my work.

I also wish to thank the official referees of this thesis. I thank Prof. Fred Gardiol for having presided both the private and the public defenses. Boundless gratitude goes out to Prof. Hervé Bourlard who shared with me his scientific vision and enthusiasm about neural networks and speech recognition; I am also very grateful for the support he gave me for organizing my visit at UC Berkeley. I had also the chance to have in my jury Prof. Christian Wellekens and Prof. Frédéric Bimbot who provided me with many useful and friendly comments about the thesis.

I will always remember with pleasure and gratitude the entire personnel from CIRC for creating a friendly and supportive atmosphere in which to work during the years of this study. I will be forever grateful to Dr Hervé Dedieu who, more than being an excellent scientific advisor, was also a big brother, always ready to help and support me in my everyday life; I will miss the friendly Pelote Basque tournaments we had together in the CIRC corridor. I benefited much from the constant energy and thirst for knowledge of Dr Dijana Petrovska without whom the speaker verification part of the thesis would not have been accomplished; I wish to thank her for the fruitful collaboration and for the constant support she gave me for the thesis. I had the great pleasure to work with Arnaud Robert whose scientific curiosity and tennistic abilities will remain an example for me; I also sincerely thank him for the help he gave me putting the thesis manuscript together. Thanks a lot to Prof. Patrick Thiran for his friendly encouragements and for his useful LaTeX expertise. I also wish to thank Dr Bertrand Dutoit and Oscar Defeo for their constant availability and their great support in administering the computers. Sincere thanks goes out to Christiane Good for running all the administrative matters with a unbounded effectiveness.

I had also the priceless chance to meet some key people who made me go a step further in the long path of the thesis. I express my gratitude to Prof. Nelson Morgan who let me spend six months in his realization group at the International Computer Science Institute, UC Berkeley; I have undoubtedly benefited from his great competence and expertise in the field of hybrid systems. I warmly thank Dr Gérard Chollet for the sharp feedback and advice he always provided me in my work and also for his fruitful collaboration in the speaker verification part of the thesis. I had the chance to take part into many different projects and I have enjoyed working together with highly competent collaborators: Dominique Genoud, Christophe Ris, Hakan Melin, Jan Černoký, Vincent Fontaine, Warner Warren and all the people in the Elisa consortium .

I was very fortunate to be surrounded by a wonderful family and great friends who never let me down in the crucial periods of the thesis. I am thinking particularly to my girlfriend Paola who always comforted me with her presence and beloved attentions and to my brother Guy who did not hesitate to travel from Belgium to give me a helpful hand at the end of the thesis. As well, I express my gratitude to my friends le Gros Emeka and his brother Ugo for their constant phone support, to Albert who lend me his sharp correcting eyes reading parts of my prose, and of course to Sandrine, Gianluca, Olivier(s), Mad Ben, Fabian, Laurent(s) and all the others who have helped me more than they know.

Lastly but most importantly, I thank my wonderful family for the love and patience they always gave me in this and all my projects.

Chapter 1

Introduction

Victory usually goes to those green enough to underestimate the monumental hurdles they are facing.

Richard Feynman, Physicist

It has been a long standing technical challenge to let machines acquire and expand certain human abilities. Maybe the most difficult tasks, apart from the higher level functions of the brain, are speech and image recognition. Modern computers outperform the human brain by orders of magnitude as far as the execution speed of elementary operations are concerned. On the other hand, they are far behind when it comes to deal with more complex perceptive tasks. It is therefore more than natural that scientists and engineers sought inspiration from biology in order to build more elaborate machines. One approach is to imitate the human brain by Artificial Neural Networks (ANNs). Of course, ANNs are only very poor imitations of biological neural networks.

The growing importance of telecommunication has urged the need for automated services in which speech and speaker recognition have become key techniques. Neural networks are often considered to be a computational analog of the parallel, distributed processing in biological systems; the connection strengths (weights) between the network processors (neurons) are roughly analogous to the strength of synapses in organic systems. In this work, neural networks will be tackled from a purely mathematical standpoint, dropping quickly any connections with biology. Amongst other advantages which are described later in the text, ANNs can handle highly nonlinear problems, making them superior to classical linear approaches and often making them achieve better performances than would analytical models, in practical applications. For example, systems based on ANNs could offer solutions to solve the difficult problem of dealing with speech in the telephone environment.

In this thesis, we shall be concerned with the two fields of automatic speech recognition (ASR) and automatic speaker recognition (ASkR) in telephony. More precisely, hidden Markov models (HMMs) in which artificial neural networks (ANNs) are used in place of more classical tools are analysed. The scope is limited to the analysis of feature extraction algorithms, of acoustic/speaker modeling and of isolated word decoding.

Several speech and speaker recognition systems based on ANNs are presented in this work. Some are completely original, some are the continuation of previous research work and some are the results

of collaborations with colleagues. Each system is presented in self-contained chapters that could, if necessary, be read separately. A detailed overview of the thesis report is given in the next section.

1.1 Outline of the report

Chapter 1: Introduction This introductory chapter gives a global picture of the ASR, ASkR and ANNs fields. A classification of the tasks is given in order to place in their context problems that are addressed in this work . A range of potential and commercialised applications is also presented.

Chapter 2: Fundamentals Chapter 2 is dedicated to describe fundamental theory which is needed to understand how speech and speaker recognition systems work. Statistical pattern recognition, feature extraction, hidden Markov models and neural networks are introduced in this part.

Chapters 3 and 4 describe two speech recognition systems in which ANNs are used in place of classical and widely used components. Chapters 5 presents a new segmental speaker recognition systems based on ANNs. Each of these chapters can be read independently. They all have their own introductions, motivations, descriptions and bibliographies. Results and comparisons with state-of-the-art systems are reported in each case. Performances are evaluated on different databases which are described in appendix A.

Automatic Speech Recognition

Chapter 3: Self-Organizing Maps and Discrete HMMs. The vector quantizer, instead of being designed with a classical minimum distortion algorithm (Kmeans), is designed with an unsupervised self-organizing neural network (Kohonen Self-Organizing Map).

Chapter 4: Feedforward ANNs and Continuous HMMs. Instead of using multi-Gaussian probability density function estimators, a feedforward ANN is used to generate local hypotheses, leading to the so-called **hybrid HMM-ANN systems**. Many studies have already been done in this direction in which advantages of hybrid systems have been underlined. In addition to the validation of this approach, a new theoretical formalism that describes the behaviour of hybrid systems is presented. In the light of this formalism, a new algorithm to train the ANN is presented. The algorithm is derived from the forward-backward training algorithm and presents advantages when few training data are available to build the system.

Automatic Speaker Verification

Chapter 5: A segmental approach for speaker verification. A segmental approach is proposed as an alternative to global modelling techniques as GMM. The general idea of the segmental approach is to recover the advantages of text-dependent techniques through an automatic segmentation of the speech material into sub-word units. Speakers are then modeled on these units that are more precise in the acoustic vector space and less sensitive to variabilities. A segmental system based on temporal decomposition, automatic clustering and a set of discriminant ANNs is proposed, analysed and evaluated on a standard speaker verification task. The segmental approach shows better robustness than global approaches in the case of mismatched conditions between training and testing.

1.2 Automatic Speech Recognition

The goal of speech recognition is to bring a voice based interface enabling the communication between men and machines. Ultimately, the voice input should be natural (including hesitations, grammatical errors, ...) and could be degraded by different factors like additional noise, channel distorsions, echos, ... Today, this task is far from being achieved, but some restricted applications working in well defined environment with up-to-date technologies are made available. During the last few years, a lot of improvements have

been carried out and, given the obvious industrial interest, several ASR systems are now proposed to the public.

More formally, an ASR system automatically transcribes the discourse from the speech signal. In other words, the system has to derive a sequence of symbols from a stream of acoustic informations. Those recognized symbols can be sentences, words or other representations of what has been said. The more general task of automatic speech understanding, which includes the extraction of meaning or production of answer, is out-of-the-scope of the work reported here.

The approaches described in this report (and the ones of most of up-to-date technologies) are essentially based on statistical pattern classification. In such a framework, sentences, words, or any other speech units (phonemes, ...) are depicted in terms of statistical models which attempt to characterize the temporal and frequential features of the speech signal.

1.2.1 Task Classification

The ASR applications are usually classified according to some aspects of the speech input, making the system easy or difficult to realize :

Amount of speakers The more speakers you allow in the system, the more variability you get in the characteristics of the signal and the least robust is the recognizer. Usually, the system is said to be *speaker-dependant* if there is only one speaker accessing the system. A speaker dependent system is easier to implement and offers better performance in terms of recognition accuracy because of the lower variability of the speech features. The variability affecting speaker-dependent systems is due to the slow variation of the voice feature of the talker and is reffered as *intra-speaker* variabilty. Intra-speaker variability can be due to many factors (age, stress, sickness,...). However, many applications require to be available for any speaker (e.g. most telephone applications). The system is then said to be *speaker-independent* , working with a potentially infinite number of speakers. Speaker independent systems are mainly affected by *inter-speaker* variability due to the fact that people have different vocal tract and different way of using it. Since we are dealing with statistical systems, independancy is simply obtained by training the systems on a large amount of different speakers covering as many speaking styles as possible. An intermediate system is a *multi-speaker* system that works well only on a limited set of speakers on which the system has been trained on.

Both speaker-dependent and speaker independent systems can be *adapted* (in either supervised or non-supervised way) to track or to particularize the system to the talker characteristics.

Speaking rate The complexity of the task increases as the fluency and spontaneousity of the speech increases:

- isolated word recognition : words are surrounded by periods of silence allowing each word to be processed separately.
- continuous speech recognition : the word boundaries are no longer well defined, hence, the recognizer must deal with co-articulation (the pronunciation of a word is affected by the phonetic context) and with insertion and deletion of keywords.
- spontaneous speech recognition : the level of complexity can deeply change whether we try to decode read text or spontaneous conversations including hesitations, grammatical errors, disfluencies ...
- keyword spotting : in this case, some pre-defined words are detected in any utterance and out of vocabulary words are rejected.

Vocabulary size and complexity The vocabulary size and its complexity are also important factors. Generally, if the vocabulary size increases, the confusionability increases and the recognizer performs less efficiently[1].

[1]Small vocabularies can be difficult to handle if they are highly confusionable (e.g. alphabet, phoneme recognition).

Difficulties of the environment A lot of variables can significantly affect the recognizer peformance: environmental noise such as stationary or non-stationary additive noise (car noise, factory noise, ...), twisted acoustics and correlated noise like reverberation, non-linear distortion, frequency band of the transmission channel, modification of the speaking style due to stress, emotion, Lombard effect, speaking rate, breathing noise, ... An important problem for current systems is the presence of multiple and possibly competing[2] speech sources.

1.2.2 Applications

Efforts in improving the speech recognition technology coupled with the incredible advances in computer performances and the cost of such systems are almost simultaneously enabling many practical applications.

ASR systems are now already believed to change the way we will interact with computers (Gross, Judge, Port, and Wildstrom 1998) (Comerford, Makhoul, and Schwartz 1997). Today's voice dictation machines are good examples. Dragon system (1990) is able to understand 30,000 words; Kurzweil AI's system is a 50,000 word system for medical application; IBM's system (1992) is able to recognize 20,000 words. All these systems are *speaker dependent applications* meaning that they must be trained by the user, either by speaking a prescribed series of sentences, either by letting the system adapt to his voice during an initial period of dictations. Speaking rates of more than 50 words per minute can be achieved and word error-rates are as low as 3 to 5 % (Bahl and al. 1989) (Roe and Wilpon 1993). Eastern non-alphabetic languages based on ideograps transcription like Mandarin Chinese should benefit from such voice dictation systems because of the inherent difficulty of entering ideograms into computers (Lee 1997b). Today, for the input of Chinese characters into computers, more than 200 different methods have been developed, based on mapping from keyboards originally designed for alphabetic languages to these Chinese characters. These methods are either too slow (phonetic symbol input) or too complex (radical input). The Chinese community, with a quarter of the world's population, represents a huge market for voice dictation and about 100 research groups with thousands of people, mainly located in Asia, are currently focusing on this topic.

ASR systems are also being introduced into the *telecommunication* field at increasing pace which both aim to cut the cost of services (a computer can replace thousand of human attendants in the case of directory assistance) or to generate new services such as banking (Nippon Telephone) or Stock Exchange quotation retrieval (Bell Northern Research (BNR). Due to the variabilities introduced by telecommunication channels and to the speaker independent characteristics of the applications, the vocabulary and the way the speech is input are more constrained in order to reach viable performances. Neverheless domains of application are multiple and with high potential :

- automatic information service
- voice dialing, free hand phones, ...
- command and control of database consultation service
- voice banking
- booking/purchase services by phone

In order to give an idea of the present capabilities of today's ASR through a telephone line, we can report the field trials made by the BNR with a new stock quotation service based on an Hidden Markov Model (HMM) recognizer. The BNR has introduced this service since mid-1992. Using the 2000 company names of the New-York stock exchange, entered in phonetic form, callers can obtain the current price of a stock simply by speaking the name of the stock. The experimental system which is freely accessible is called thousands of times a day, and callers are obviously able to obtain the quotation they want (Lennig 1992) (Roe and Wilpon 1993).

Other applications based on ASR systems have been introduced in many different area :

[2] Automatic annotation of broadcasted speech like tv or radio show

- web voice-controlled browsers
- form fill-in by voice
- text and data encoding
- command and control of manufactoring process
- help to the disabled
- learning of a language, ...

1.3 Automatic Speaker Recognition

Biometric person authentication can be achieved through the speech signal which carries informations both on the geometrical vocal tract configuration and on the way the person is talking (prosody, coarticulation, phoneme rate, ...). ASkR is a good authentication alternative in comparison to more "penetrating" biometrics as finger prints, retina and ADN analysis. Nevertheless, the speech signal is inherently variable and requires estimation of averaged values of some relevant parameters. This is accomplished during one or more *enrolment* sessions. In practice, the speech features are always affected by the state of the speaker, the background noise, the microphone and the communication channel, which implies that recording few enrolment sessions will yield most of the time to biased estimates of the parameters. The length and the number of enrolment sessions will then condition the quality of the estimates and therefore the performances of the system. For the user's convenience, enrolment should be as short as possible. For the system, enrolment should be long enough to cover most of the variabilities.

1.3.1 Task classification

ASkR aims at getting informations about the identity of the speaker analysing the speech signal (and only the speech signal[3]) coming out of his vocal tract. ASkR is usually split into two distinct technologies, yet closely related. The first technology is *speaker verification* (SV) in which the goal is to verify the identity of a claimed speaker. The second technology is *speaker identification* (SI) in which the goal is to match the voice pattern of a speaker to previously stored patterns of a set of speakers. For tutorials papers on SV and SI, see (Doddington 1985) (O'Shaugnessy 1986) (Naik 1990).

Often, research focusses on SV, mainly because at present, SI is still not reliable in the case of a large speaker set. Another reason is the small number of potential real-life applications of SI.

Tasks can also be classified following the strategies used to design the system. There are roughly three strategies which differ in the type of text user must speak in order to get recognized :

1. **Text-dependent**
   - *System selected passwords :* An a priori fixed phrase is imposed by the system. Generally, the training and testing material is based on words taken from a limited lexicon. *Personal Identification Numbers* (PIN) systems, for example, generate different sequences of digits for each user, allowing to accomplish the identity claim and verification at the same time. More flexibility can be achieved when using sub-word models.
   - *User selected passwords:* The user can decide on the password. This is generally more difficult to solve since the system cannot rely on knowing what the speaker must say.
2. **Text-Prompted :** The sequence of words is, a priori, not known anymore by the user. Instead, the system will prompt him to utter a random sequence of words. This strategy achieves a higher level of security in preventing lawbreakers of using user's pre-recorded speech signal to enter the system. Text-prompted systems can generate a rich set of prompts if based on sub-word modeling.

[3]A new speaker authentication technology, known as *verbal information verification* (Lee 1997a) which aims at authenticating speakers by verifying the content of the spoken utterance against the user's personal profile, is not investigated here.

3. **Text-independent :** The user is free to say anything he wants (and sometimes he can be very funny) and no a priori knowledge of what he is going to utter can be used. The obvious advantage of text-independent AskR is that the user is not constrained to remember any password. On the other hand, the vulnerability of the system is increased since any recording of the user's voice might break into the system. Performance of text-independent ASkR is generally of a lower level than what can be obtained with the two previous categories in which a priori knowledge of what the speaker must say can be exploited.

In the previous enumeration, ASkR systems are more or less classified from the most constrained to the less constrained or from the most efficient to the less efficient in terms of recognition accuracy.

1.3.2 Applications

Obviously, ASkR has less direct applications than ASR do. We will focus here on SV since most of the potential applications are in this domain. A good survey article of SV systems in commercial applications is given in (Boves 1998).

- **Phone banking** requires the authentication of the person when sensitive transactions are processed. In most of the current systems, a verbal authentication is used in which the user is queried for passwords or personal informations. These informations can easily be reproduced by lawbreakers while reproducing voice is maybe less obvious, especially for text-prompted systems. Field tests have been conducted by Ubilab (from Union Bank of Switzerland) and USA City Corp but no known real application is actually working.

- **Calling card** services could also reduce fraud with SV systems. Sprint Foncard in USA is already offering a calling card PIN-based system to verify the identity of the caller. High charges and limited voice dialing is leading to poor success of the Sprint service. No verification rates were ever published by Sprint.

- **Teleshopping** services are a major source of unauthorised use of credit cards and could also benefit of SV systems.

- **Home incarceration** and parole/probation monitoring is a less known but interesting application of SV systems. Several experiments with home incarceration have already started around the world, motivated by the fact that jails are overcrowded and cost a lot of money. Furthermore, threre is an increasing awereness of the negative effects of emprisonment. SV can provide an attractive way to monitor outmates who are asked to call for report at some time of the day. The Speak-EZ SV product commercialized by T-Netix is already used in several states in USA for home incarceration purposes. Recently, Lernout and Hauspie have started a beta test of password-based SV in Belgium in the framework of a home incarceration programme.

- **Access control** systems could use SV in combination with the usual mechanisms (key or badge) to improve security at relatively low cost. Several companies are offering systems for access control : Voice Strategies in Troy MI, International Electronics in Canton MA and Keyware in Europe.

1.4 Artificial Neural Networks

A neural network can be seen as a set of simple units named *neurons* interconnected to give an oriented architecture. The generated network has *inputs* and *outputs*, and the system variables are the *weights* of the connections between neurons. A neuron's function is to gather information via a weighted sum of its inputs. A nonlinear *transfer* or *activation* function sends the results to the subsequent units until the results reach the output units. A neural network is then simply a graphical notation for an equation, one that states how output values are generated from input values. The main characteristic of such networks is their ability to *learn* - to fix their weights - from examples, which a computer can do automatically.

ANNs offer many theoretical and practical advantages :

- **Development speedup**. ANNs speed the development of new applications, relaxing the need to develop and program special analytical models, a laborious and time-consuming process.
- **Good performances**. ANNs can handle highly nonlinear problems, making them vastly superior to classical linear approaches and often making them achieve better performances than do analytical models in practical applications.
- **Online adaptation**. ANNs are able to adapt online, coping with process fluctuations and improving their internal knowledge on the field.
- **Exploitation speedup**. ANNs, with their repetitive structures, are well suited to speed up software and hardware implementations (parallelism, VLSI design, ...).

1.4.1 Classifications

The learning algorithms of ANNs are often classified in two categories, related to how the connection weights are chosen to perform a specific task.

- **Supervised learning :** Here weights are tuned on the basis of direct comparison of the output of the network with known correct answers. The training data is a set of network inputs coupled with the corresponding target outputs.
- **Unsupervised learning :** The network is expected to discover some underlying organization of the training data and to tune automatically its weights to output categories. No informations on correct output is available and the training data is a set of network inputs.

1.4.2 Applications

In recent years, there has been a lot of theoretical and experimental research that has installed neural networks as useful technology for many practical applications. The list of real-world commercialized applications of neural networks is already impressive. For example, they are used in about half the optical character recognition systems on the market today, including the Newton portable digital assistant from Apple (Yaeger 1996). They are used for steel manufacturing at Sollac, France (Pican, Alexandre, and Bresson 1996) and Siemens, Germany (Schlang, Poppe, and Gramchow 1996) to preset the rolling and bending forces of steel temper mills. They are used for computer virus recognition as part of the IBM Anti-Virus software package (Tesauro, Kephart, and Sorkin 1996).

In this thesis, we will focus on the use of artificial neural networks in speech and speaker recognition. There are numerous attempts in this direction reported in the literature and a review of some of them is reported in appendix C of this report. Other survey articles are (IEEE-TSAP 1994), (Lippmann 1989). and for more introductory information on ANNs in general, see (Hertz, Krogh, and Palmer 1991).

Bibliography

Bahl, R. and al. (1989, May). Large vocabulary natural language continuous speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 465–468.

Boves, L. (1998). Commercial applications of speaker verification: Overview and critical success factors. In *Speaker Recognition and Its Commercial and Forensic Applications (RLA2C)*, Avignon, France, pp. 150–159.

Comerford, R., J. Makhoul, and R. Schwartz (1997, December). The voice of the computer is heard in the land (and it listens too. *IEEE Spectrum 34*(12).

Doddington, G. (1985, November). Speaker recognition - identifying people by their voices. *Proceedings of the IEEE 73*(11), 1651–1664.

Gross, N., P. Judge, O. Port, and S. Wildstrom (1998, February 23). Let's talk. *Business Week*, 45–56.

Hertz, J., A. Krogh, and R. G. Palmer (1991). *Introduction to the theory of Neural Computation*. Santa Fe Institute Studies in the Sciences of Complexity. Addison Wesley.

IEEE-TSAP (1994, January). Special issue on neural networks for speech. *IEEE Transactions on Speech and Audio Processing 2*(1).

Lee, C.-H. (1997a). A unified statistical hypothesis testing approach to speaker verification and verbal information verification. In *Speech Technology in the Public Telephone Network, Where Are We Today ?*, Rhodes, pp. 63–72.

Lee, L.-S. (1997b, July). Voice dictation of mandarin chinese. *IEEE Signal Processing Magazine 14*(4), 63–101.

Lennig, e. a. (1992, October). Automated bilingual directory assistance trial in bell canada. In *Proceedings of the first IEEE workshop on Interactive Voice Technology for Telecommunications Applications*, Piscataway, N. J.

Lippmann (1989). Review of neural networks for speech recognition. *Neural Computation 1*(1), 1–38.

Naik, J. M. (1990, January). Speaker verification: A tutorial. *IEEE Communications Magazine 28*(1), 42–48.

O'Shaugnessy, D. (1986, October). Speaker recognition. *IEEE Trans. Acoust., Speech, Signal Processing*, 4–17.

Pican, N., F. Alexandre, and P. Bresson (1996, February). Artificial neural networks for the presetting of a steel temper mill. *IEEE Expert Intelligent Systems and Their Applications 11*(1), 22–27.

Roe, B. and J. G. Wilpon (1993, November). Whither speech recognition: The next 25 years. *IEEE Communications Magazine 31*(11), 54–62.

Schlang, M., T. Poppe, and O. Gramchow (1996, August). Neural network for steel manufacturing. *IEEE Expert Intelligent Systems and Their Applications 11*(4), 8–9.

Tesauro, G., J. Kephart, and G. Sorkin (1996, August). Neural networks for computer virus recognition. *IEEE Expert Intelligent Systems and Their Applications 11*(4), 5–6.

Yaeger, L. (1996, August). Neural networks provide robust character recognition for newton pda. *IEEE Expert Intelligent Systems and Their Applications 11*(4), 10–12.

Chapter 2

Fundamentals

You think you know when you learn, are more sure when you can write, even more when you can teach, but certain when you can program.

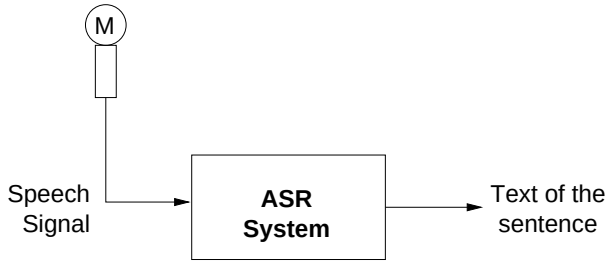
Alan Perlis

The speech signal is produced by a system, generally a human being. Our goal is to build up a processing system that classifies some events occuring in the speech signal. For speech recognition, the objective is to give the correct sequence of words as output (fig 2.1, a). For speaker verification, the goal is to decide wether the speech signal belongs to a target speaker or to another set of different speakers.

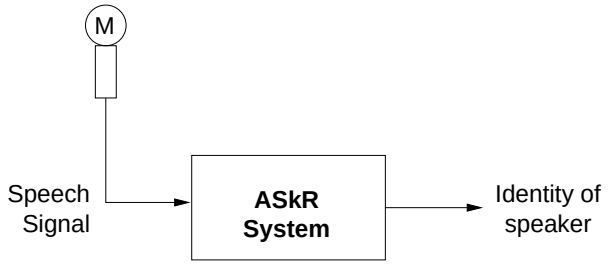
In both cases, we are facing a *pattern classification* problem in which we would like to build *classification rules* that tell us which *patterns* to output for every possible *observation*. The classification rules will have to satisfy some criterions which are generally tied to the minimization of some cost of mis-classifying patterns.

The approaches reported in this text are related to modeling procedures. The scheme of any modeling procedure is twofold. First a structure for the model has to be chosen in such a way that it reflects some underlying structure of the physical system. Some *a priori knowledge* about the system can generally be exploited at this point. Second, a so-called *learning* or *adaptative* or *data-driven* algorithm uses a set of training data to tune the model parameters in order to meet a criterion. The term *statistical modeling* is used to refer to this learning phase.

We will start this chapter with a description of the speech signal and underline difficulties that automatic recognizers are facing when dealing with it. Starting from this, we briefly introduce nomenclature and theory of statistical pattern recognition. Notions of training and testing data are reviewed. Feature extraction principles are then introduced and the widely used lpc-cepstral representation is presented. Hidden Markov models (HMMs) are then described with the set of hypothesis necessary to make them tractable from a computational point of view. A section is also dedicated to describe particular problems that are encountered in speaker verification (error analysis and thresholding). Finally, we close this chapter with some fundamental theory on neural networks.



a)



b)

Figure 2.1: Automatic Speech Recognition system (a) and Automatic Speaker Recognition system (b)

2.1 Speech Signal

The human speech signal is the result of the execution of neuromuscular commands that shape the vocal tract and causes vocal cords to vibrate or not vibrate. Roughly, the vocal tract moves from one position to another such that a proper sequence of speech sound is created by the talker. An acoustic wave is propagated on a communication channel and arrives to the listener who decodes it and retrieves the message.

Figure 2.2-a gives an example of a speech waveform recorded over a telephone line. The speech signal is said to be slowly time varying or quasi-stationary because when examined over short time windows (figure 2.2-b), its characteristics are fairly stationary ($5-100$ msec) while over long periods (figure 2.2-a), the signal is non-stationary (> 200 msec), reflecting the different speech sounds being spoken.

In sections 1.2.1 and 1.3.1, a classification of the ASR and ASkR tasks has been given and the difficulties inherent to the speech signal were already introduced. We sum-up here more formally the features that make the speech signal difficult to deal with.

(Morgan and Bourlard 1995) underlined five difficulties :

1. **High redundancy :** the speech signal is short-term redundant (see figure 2.2, b). This redundancy, surely helpful for humans, does not help the classification of speech units, except in getting an averaged estimate of some noisy features.

2. **Signal degradation :** additional noise and convolutional noise degrade the signal: environment, channel transmittance, microphone characteristic, room reverberation, etc. Experiments reported in this thesis are carried on telephone speech signals which are degraded by bandwidth limitation (100-3800 Hz) and by channel and microphone variabilities.

3. **Temporal and frequency variabilities :** speech units are produced with variable speed and frequencies. Variabilities are said to be intra-speaker when due to differences of the state of the

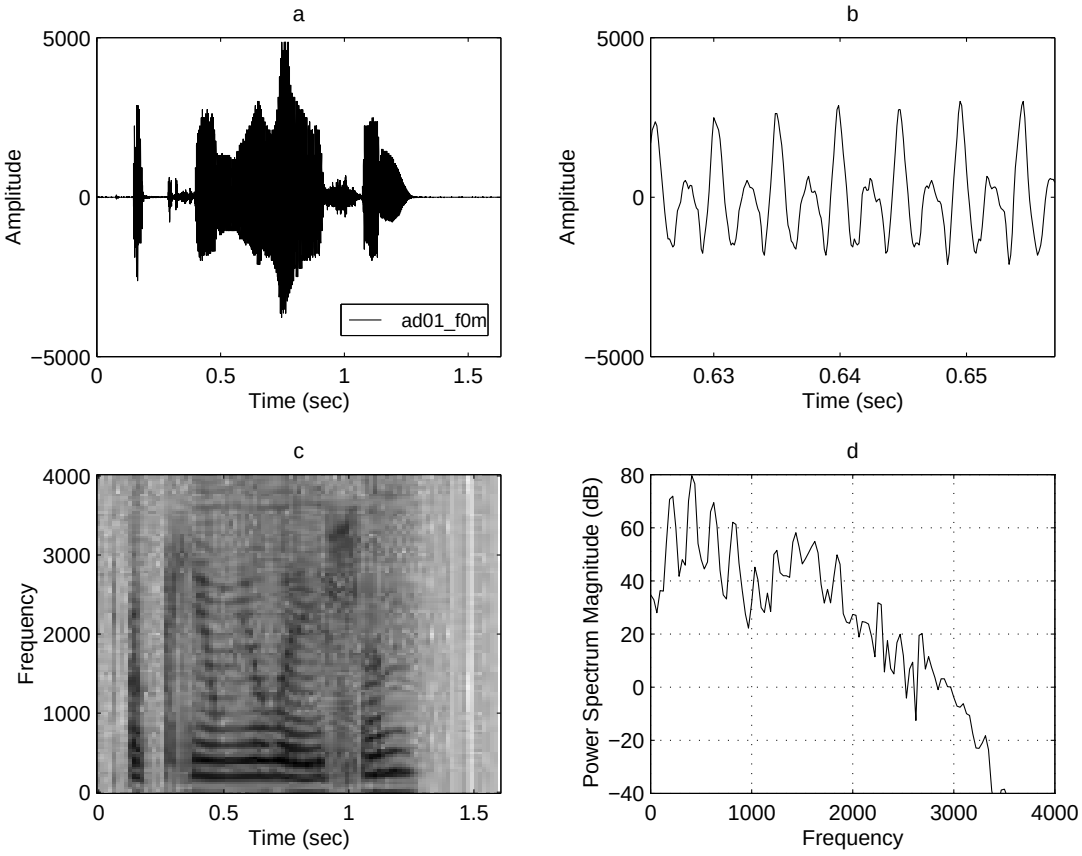


Figure 2.2: Speech signal of the word *accumulation* taken from the Phonebook database. (a) : waveform, (b) partial waveform, (c) narrowband spectrogram of (a), (d) power spectrum magnitude of (b).

speaker (emotional, health, ...). Variabilities are said to be inter-speaker when due to differences between speakers (sex, weight, ...).

4. **Coarticulation :** the inertia of the vocal tract makes the realisation of speech units context-dependent, i.e. the waveform of a particular speech unit depends to the preceding and the following speech units.

5. **User :** the behaviour of the talker is most of the time unpredictable. He might use out-of-vocabulary words. He might hesitate, producing filled and/or empty pauses. He might produce all kind of sounds that are not part of the message, e.g. lip smack, cough, grunt, throat clear, tongue click, loud breath, laugh, loud sigh. He might utter word fragments or insert/delete whole words, etc.

Two more features of the speech signal can also be sources of difficulties:

6. **Multiple speech sources :** in the case of spontaneous speech, multiple and often competing speech sources can overlap in time.

7. **Language ambiguity:** the language itself can be ambiguous because of homophones.

2.2 Statistical Pattern Recognition

When doing pattern recognition [1], we are faced with a decision problem about the *state of nature* M of an observation, taking into account some evidence or *pattern* $\mathbf{x}$ about the observation. Let us define some variables:

- M the random variable associated with the state of nature. M takes discrete values in the case of a finite number of classes.
- $\mathcal{M} = \{M_0, M_1, \cdots, M_{\mathcal{I}}\}$ the set of states of nature or set of *classes/categories* with $\mathcal{I}$ the total number of possible classes.
- $\mathbf{x}$ a pattern determined from the acoustic signal. $\mathbf{x}$ is also a random variable taking continuous or discrete values.

Our aim in doing pattern classification is to minimize the amount of classification errors [2]. Bayes decision theory tells us that the optimal procedure is the one that chooses the class that has the highest *posterior probability* given the evidence :

$$M_{j_{opt}} = \underset{j}{argmax} P(M_j|\mathbf{x}) \tag{2.1}$$

In the case of statistical pattern recognition, an estimate of the posterior probability is built from data and from assumed prior knowledge. The performance of the classifier depends on how well the posterior probability is estimated and therefore on the validity of the assumed prior knowledge. In any case, the lower bound of classification errors is defined by the true form of the posterior probability and none of its estimates will beat this value.

Generally speaking, a prior knowledge is any piece of information that we know about the input data. In the particular case of speech processing, prior knowledge is often derived studying the way humans do generate and recognize speech. Prior knowledge can come as well from characteristics of the transmission channel or from usual assumptions such as the form of the probability density functions.

Two key problems have to be solved to design a statistical pattern recognizer:

[1] (Duda and Hart 1973) wrote a very good textbook on pattern recognition and we refer to it for a more thorough treatment.

[2] A more general framework for the theory is to minimize the overall risk of taking an action α_i. It implies the computation of (1) losses $\lambda(\alpha_i|M_j)$ incurred for taking action α_i when the state of nature is M_j and (2) posteriors probability $P(M_j|\mathbf{x})$ (Duda and Hart 1973). Our discussion will stick to the minimization of the amount of classification errors since it can be easily extended to the minimization of the overall risk.

- **How to determine a good pattern x from the acoustic signal ?** The pattern is computed with a so-called *feature extractor* which should optimally transform the raw input into something easilly classified. Prior knowledge is usually used at this stage to design the feature extractor.

- **How to build a good posterior probability estimator ?** The first issue is the modeling part in which a mathematical model has to be chosen in order to estimate the posterior probability of equation 2.1. The form of the model is generally chosen making some assumptions on the probability distributions or trying to include prior knowledge of the problem. The second issue is to tune the model parameters to fit the estimate onto the real posterior probability distribution. This is generally accomplished with so-called data-driven or learning algorithms which go through a set of patterns x. This set of data is called the *training data set*. Of course, it should be rich enough to represent all the possible realisations of the random variable M in order to build good estimates.

Learning can be achieved through different procedures. In *supervised learning* , class labels for each pattern in the training set are provided and the learning algorithm seeks to reduce the error on these patterns. In *reinforcement learning* or *learning with a critic*, only the feedback that a tentative categorization is correct or not is given. In *unsupervised learning*, there is no explicit a priori classes and the system forms clusters or "natural grouping" of the input patterns. In this text, we are concerned with supervised and unsupervised learning, both applied to well spread probability estimators and to neural networks which can be seen, as it will be shown further, as a familly of estimators with their accompanying learning schemes.

The quality of the pattern recognizer is often estimated on an independent set of patterns for which the correct classification is known. This set is called the *test data set* and serves to compute the classification accuracy. A third set of data called the *cross-validation* data set is often used to avoid over-training of the classifyer which can occur when the number of parameters in the model is too large with respect to the number of training patterns. Over-training is detected when accuracy is increased on the training set while decreasing on the cross-validation set.

Although the ASR and ASkR systems could be seen as one processing block which takes as input the speech waveform and gives as output the recognized entities as illustrated on figure 2.1, there is generally a break-down of the mapping into the initial *pre-processing* or *feature analysis* stage, followed by the parametrized *classifier* model itself. As well, for many systems and particularly ASR and ASkR systems, the outputs of the classifier also undergo *post-processing* to convert them into a corrected form. Figures 2.3 and 2.4 illustrate the usual breakdown of ASR and ASkR systems into the more detailed stages. As shown on figure 2.4, the scheme of the ASkR system is similar to the one of the ASR system. Techniques are also very close, except for the post-processing stage.

2.3 Feature extraction

A good introductory paper on feature extraction in state-of-the-art speech processing systems has been written by (Picone 1993). Notations and figures have been inspired from (Rabiner and Juang 1993). This section is meant to introduce principles of speech feature extraction and to describe the widely used **linear predictive coding (LPC) cepstral** analysis.

The distinction between feature extraction and pattern classification is not always clear cut. The feature extractor should optimally transform the raw input into something easily classifiable. On the other hand, the classifier should be good enough to cope with potential incorrect assumptions made in the feature extraction[3].

In the case of the speech signal, the feature extractor will first have to deal with the long-term non-stationarity. For this reason, the speech signal is usually cut into frames of about 10-30 msec and feature extraction is performed on each piece of the waveform[4]. Secondly, the feature extraction algorithm has to

[3]Often the feature extraction can be regarded as a fixed transformation of the input which does not generally imply any tuning of the transformation parameters or does not use any feedback of the latter stage of the system. For this reason, feature extraction is also called the *front-end* part of the system.

[4]The frame length has also a lower bound limit which is dictated by the algorithm.

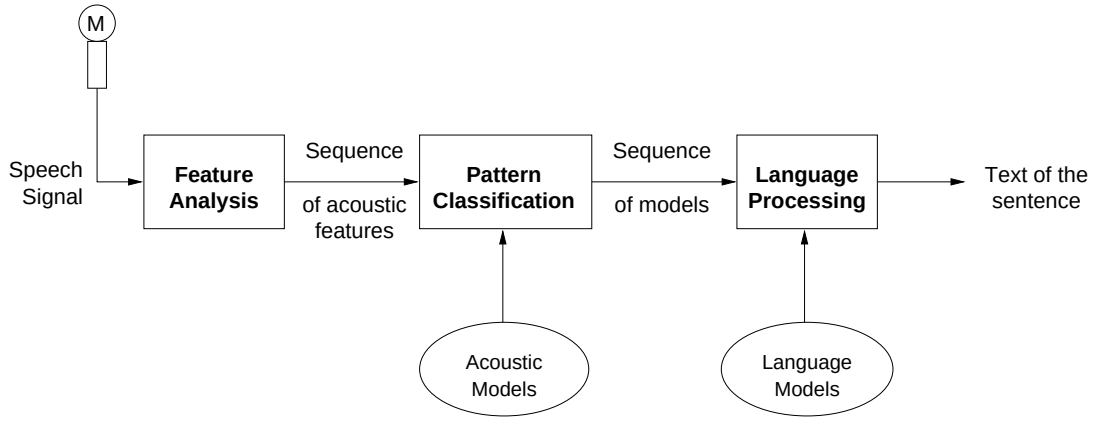


Figure 2.3: Breakdown of the ASR recognition system. **Pre-processing** : a feature analysis transforms the speech waveform in a better representation for classification. **Pattern matching** : a pattern classification block matches acoustic features on acoustic models and outputs a set of winner model hypothesis. **Post-processing** : languages models are applied to discard wrong hypothesis.

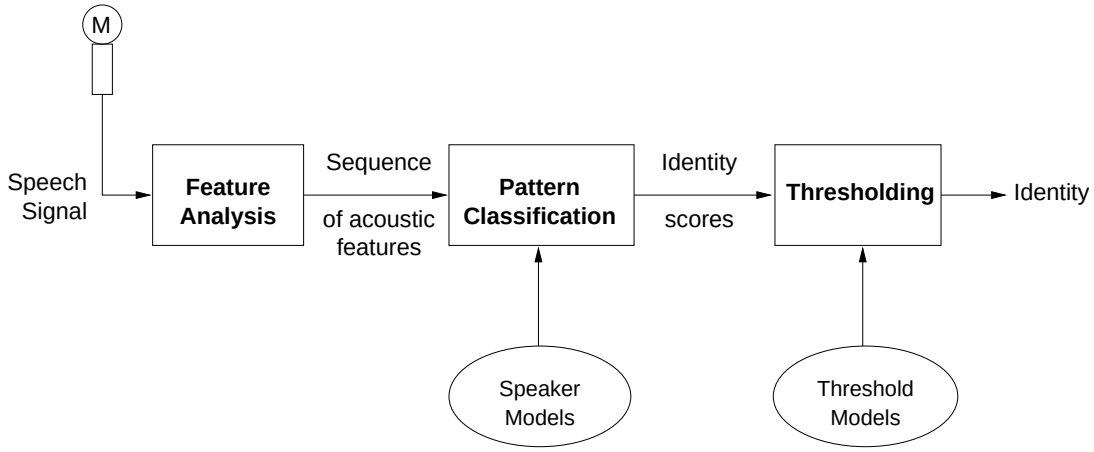


Figure 2.4: Breakdown of the ASkR system. **Pre-processing** : a feature analysis transforms the speech waveform in a better representation for classification. **Pattern matching** : pattern classification matches acoustic features with speaker models and outputs identity scores. **Post-processing** : a thresholding is applied on the identity scores and a identity decision is taken.

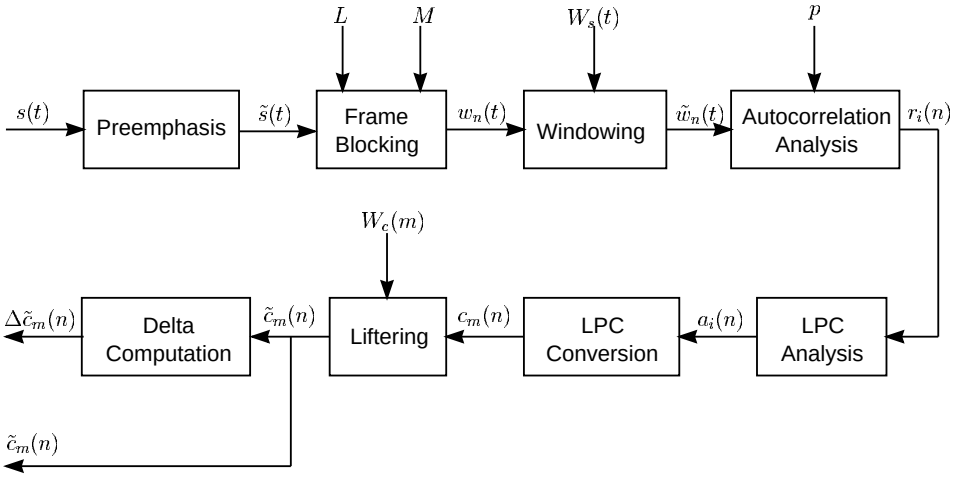


Figure 2.5: LPC-cepstrum extraction, after Rabiner.

cope with the short-term redundancy so that a reduced and relevant acoustic information is extracted. For this purpose, the representation of the waveform is generally swapped from the temporal domain to the frequency domain, in which the short-term temporal periodicity is represented by higher energy values at the frequency corresponding to the period. Thirdly, feature extraction should smooth out possible degradations incurred by the signal when transmitted on the communication channel. For example, we are concerned in this work with telephone speech for which the limited bandwidth and the channel variability need some special treatment. Finally, feature extraction should map the speech representation into a form which is compatible with the statistical classification tools in the remainder of the processing chain. Some classification algorithms will, for example, require a decorrelation of the features.

A usual feature extraction of telephone speech is the so-called **linear predictive coding (LPC) cepstral** analysis. This algorithm is widely used in the field and most of the experiments reported in this text are based on it. Consequently, more details on LPC-cepstral analysis are given here. Without going into all the details of LPC modeling, the basic idea behind it is that a given speech sample $s(t)$ at time t can be approximated as a linear combination of the past p samples $s(t) \approx a_1 s(t-1) + \ldots + a_p s(t-p)$, that correspond to idealize the vocal tract by a all-pole system excited either by a quasiperiodic pulse train (for voiced speech sounds), either by a random noise source (for unvoiced sounds).

A block diagram of a complete cepstral analysis module is given in figure 2.5. The steps in the digital processing are as follows (Rabiner and Juang 1993):

1. Pre-emphasis. The signal is passed through a first-order digital network $\tilde{s}(t) = s(t) - as(t-1)$ [5]. This operation spectrally flattens the signal and avoids some finite precision effects latter in the processing. In our experiments, the pre-emphasis factor a is set equal to 1.

2. Blocking into frames. Sections of L samples are used as single frames. Consecutive frames are spaced M samples apart. In our experiments, we use $L = 240$ corresponding to 30 ms of speech and $M = 80$ corresponding to 10 ms frame spacing (or 20 ms frame overlap). The 30 ms length is actually a trade-off between too long frames falling on non-stationary part of signal and too short frames disabling any spectral analysis. We denote the n^{th} frame of speech by $w_n(t)$:

$$w_n(t) = \tilde{s}(Mn + t) \quad t = 0, \ldots, L-1 \quad n = 0, \ldots, N-1 \tag{2.2}$$

3. Frame windowing. Each frame is multiplied by a L sample window $W_s(t)$ so as to minimize border

[5] t is discrete time

effects.

$$\tilde{w}_n(t) = w_n(t)W_s(t) \qquad 0 \leq t \leq L-1 \tag{2.3}$$

A Hamming window (see figure 2.6 left) is used in our experiments :

$$W_s(t) = 0.54 - 0.46cos(\frac{2\pi t}{N-1}) \qquad 0 \leq t \leq L-1 \tag{2.4}$$

4. Autocorrelation analysis. Each windowed set of speech samples is autocorrelated to give a set of $p+1$ coefficients where p is the order of the desired LPC analysis. We use here $p = 10$.

$$r_i(n) = \sum_{t=0}^{L-1-i} \tilde{w}_n(t)\tilde{w}_n(t+i) \qquad i = 0, \ldots, p \tag{2.5}$$

5. LPC/Cepstral analysis. A vector of LPC coefficients $a_i(n), 1 \leq i \leq p$ is computed from the autocorrelation vector using a recursion method (see (Rabiner and Juang 1993) for more details). A cepstral vector $c_m(n)$ is then computed from the LPC coefficients up to the D^{th} component with $m = 0, \ldots, D$ and $D > p$.

$$c_0 = ln\sigma^2 \tag{2.6}$$

$$c_m = a_m + \sum_{k=1}^{m-1} c_k a_{m-k} \qquad 1 \leq m \leq p \tag{2.7}$$

$$c_m = \sum_{k=1}^{m-1} c_k a_{m-k} \qquad m > p \tag{2.8}$$

$$\tag{2.9}$$

where σ^2 is the gain term in the LPC all-pole model. Typical value of D with telephone speech is 12. The cepstral coefficients in the low index range correspond to the slow variation of the spectral envelope. A reconstruction of the spectrum from only 12 cepstral coefficients leads to a smoothed image of the spectrum, leaving apart the details. The pitch is generally lost stopping the cepstral index at 12, which is a good operation for speaker independent systems.

6. Liftering. The D-coefficients cepstral vector is weighted by a "raised-sine" window $W_c(m)$ (see figure 2.6 right):

$$W_c(m) = 1 + \frac{D}{2} sin\frac{\pi m}{D} \qquad m = 0, \ldots, D \tag{2.10}$$

This operation is called *liftering* because it can be viewed as a filtering occuring in the cepstral domain. The form of the window is illustrated in figure 2.6. This kind of weighting improves the quality of the recognizer. Cepstral coefficients in the low index are known to be sensitive to spectral slope which can be variable with telephone speech). High order coefficients are also more sensitive to noise, justifying the overall shape of the liftering window. (Juang, Rabiner, and Wilpon 1987) have also shown that doing a liftering with a function as in equation 2.10 can smooth the peaks in the gain spectrum of the all-pole LPC system. The liftering then does not change the structure of the peaks corresponding to formants of the speech signal but reduces the sensitivity of the system regarding their amplitude and position [6].

[6] Theoretically, liftering should not affect recognizers that perform a normalization such as gaussian classifiers or some kind of feed-forward neural networks like MLP since the first layer can perform this operation. Therefore, results should not show any difference but it seems not to be the case. There is a undiscussed improvement when liftering is used for discrete HMMs in which the vector quantizer uses an Euclidian distance.

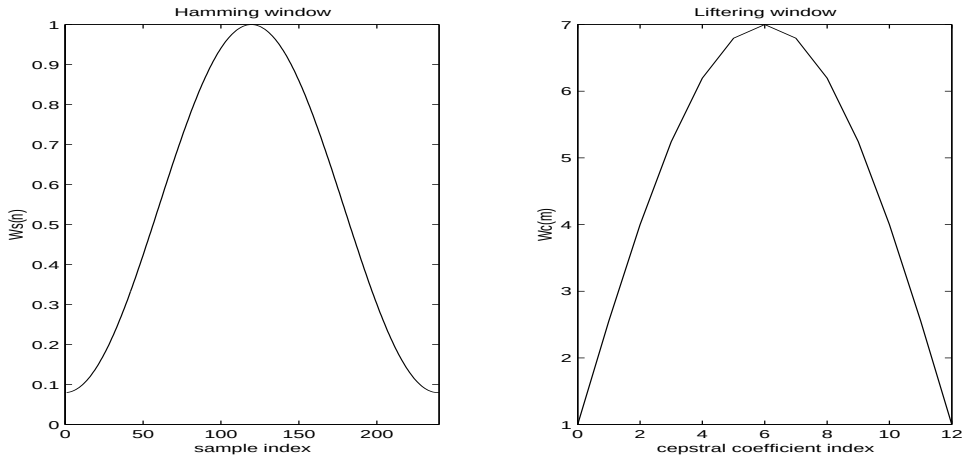


Figure 2.6: Hamming window, $L = 240$ (left) and liftering window, $D = 12$ (right)

7. Delta coefficients. Often, the weighted cepstral vector is completed with its time derivative. There are two reasons for this. First, it gives an extra information regarding the dynamic evolution of the cepstral vector. Second, it is a way to increases the 30 ms context of the ceptral vector. Because of the noise on the cepstral representation, the time derivative of the sequence of weighted cepstral vectors is approximated by a first order orthogonal polynomial fit over a finite length window of $(2K + 1)$ frames centered on the current vector. We use here $K = 2$ corresponding to a five frame window (70 ms of context). The so-called delta cepstrum vector is computed as:

$$\Delta c_m(n) = \sum_{k=-K}^{K} k c_m(n-k) \tag{2.11}$$

In equation 2.11, $\Delta c_m(n)$ is often multiplied by a gain term G chosen to make the variances of $c_m(n)$ and $\Delta c_m(n)$ equal. As explained later on, this normalization is useful when using a unique stream of data feeding the probability density functions estimators (kmeans initialization of multi-gaussian system) or the vector quantizer or the neural network. As we treat each stream of vectors (cepstral and delta-cepstral) independently in our systems, this normalization is not used, except when indicated. In case of normalization, a value of $G = 0.375$ is generally used.

8. Delta-delta coefficients. Second-order cepstral time derivatives have been shown to be useful when incorporated in ASR systems (Lee, Giachin, Rabiner, Pieraccini, and Rosenberg 1992) (Ney 1990) and especially on noisy and stressed speech (Hanson and Applebaum 1990) presenting Lombard effect[7]. Usually, the delta-delta cepstrum is evaluated as the time derivatives of the delta cepstrum as in 2.11 but with a smaller value of K :

$$\Delta\Delta c_m(n) = \Delta c_m(n+1) - \Delta c_m(n-1) \tag{2.12}$$

Larger values of K lead to delta-delta cepstral parameters somewhat too unstable because computed on a too broad window falling on non-stationarities of the speech signal.

9. Log-energy. Other temporal features such as the delta log-energy and delta-delta log-energy can also complete the cepstral vector. The log-energy for the n^{th} frame is computed as $e(n) = log(\sum_{t=0}^{L} w_n(t)^2)$ with $0 \leq tleqN - 1$. Similarly, $\Delta e(n)$ and $\Delta\Delta e(n)$ are computed as described in eq. 2.11 and 2.12.

[7]Lombart effect is related to modifications of the articulation due to the environment influence. For example, when a talker speaks in a environment with a masking noise, it has been reported that the first formant of a vowel increases while the second formant decreases. The difficulty with Lombart effect is a lack of understanding as to how to quantify it.

Based on the computations described above, the acoustic waveform is transformed by the feature extraction algorithm into a sequence of vectors $X = \{x_1, \ldots, x_n, \ldots, x_N\}$ in which a particular vector x_n has the form

$$x_n = (c_1(n), \ldots, c_D(n)) \tag{2.13}$$

In equation 2.13, cepstral coefficients are often appended to their time derivatives (first and second) and to the log-energy:

$$\begin{aligned} x_n &= (c_1(n), \ldots, c_D(n), \Delta c_1(n), \ldots, \Delta c_D(n), \Delta\Delta c_1(n), \ldots, \Delta\Delta c_D(n), e(n), \Delta e(n), \Delta\Delta e(n)) \\ x_n &= (c_n, \Delta c_n, \Delta\Delta c_n, e(n), \Delta e(n), \Delta\Delta e(n)) \end{aligned} \tag{2.14}$$

In the remaining of the text, we shall assume that feature vectors x_n takes the form described in equation 2.13 or 2.14.

2.4 Hidden Markov Models

This section is dedicated to the particular problem of modeling speech units. Notations and mathematical developments were inspired from (Bourlard and Morgan 1994). As explained in section 2.3, a speech pattern is a sequence of acoustic vectors $X = \{x_1, x_2, \cdots, x_N\}$ of length N, resulting from the feature extraction. The temporal quasi-stationary nature of the speech signal prescribes some specifications that the model will have to fulfill. As explained earlier in section 2.1, the vocal tract moves from one position to the next such that a proper sequence of speech sounds is produced by the talker. The model should then be made up of states tied to each possible speech sound. Transitions between these states should be possible to illustrate the sequential production of speech sounds. The model should also be able to cope with the temporal variability associated to the realisation of the speech signal. These three prescriptions are fulfilled by a familly of models called hidden Markov models (HMMs).

A HMM is a stochastic automaton with a stochastic output process attached to each state (see Fig. 2.7). Thus we have two concurrent stochastic processes : an underlying (hidden) Markov process modeling the temporal structure of speech and a set of state output processes modeling the stationary character of the speech signal. For more detailed information about HMMs used in speech recognition, several texts can be recommended such as (Rabiner 1989; Rabiner and Juang 1993; Bourlard and Morgan 1994; Waibel and Lee 1990).

Ideally, there would be a unique HMM model M_i for each sentence. This is of course unfeasible for many applications. Often, large vocabulary systems will require a hierarchical modeling scheme with the computation of smaller sub-word units like phonemes so that any sentence model can be built from those sub-word models. The sub-word unit most commonly used is the phone. Any given sentence can be associated to the corresponding HMM built from the concatenation of smaller HMM themselves associated to phones. When working with sub-words units, the parameters are automatically shared between word and sentence models. Phone models can be *context-dependent* or *context-independent* when they depend or not on the surrounding phone context. See (Rabiner and Juang 1993) for a more detailled discussion on subword units.

As seen in section 2.2, Bayes decision theory is a statistical approach to pattern classification and we shall use this approach to describe HMM's functioning. During testing time the optimal way to classify an input pattern is to choose the class which gives the highest posterior probability (eq. 2.1).

$$j = \arg\max_i P(M_i|X) \tag{2.15}$$

During training[8], the aim is to discover the set of HMM parameters Θ which minimizes the number of mis-classifications. In other words and following the Bayes criterion, training should maximize the

[8]We are here in the framework of *supervised learning* since we have at our disposal speech waveforms which are labelled in terms of speech units.

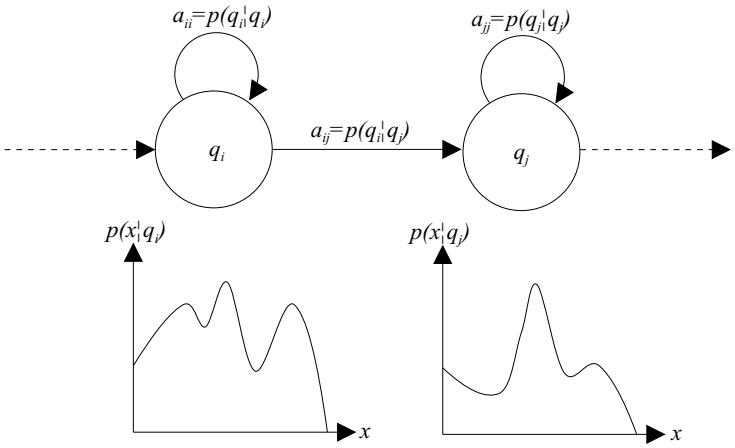


Figure 2.7: Left-to-right HMM with self loop

posterior probability of the correct model given the evidence measure (sequence of acoustic vectors). This parameter optimization criterion is called *Maximum A Posteriori* (MAP) criterion.

$$\arg\max_{\Theta} \prod_{j=1}^{J} P(M_{\omega_j}|X_j, \Theta) \tag{2.16}$$

where ω_j is index of the model corresponding to the lexical content of the training utterance j. Θ is the set of parameters of all the models. If the parameters are not shared between models, the posterior probability of the model can be noted as $P(M_i|X, \Theta_i)$. Usually, a small set of speech units (phonemes or succession of one or two phonemes) is used to model larger part of the speech signal, e.g. words and sentences, and therefore, parameters are generally shared between models. During recognition time, the parameter set Θ is generally fixed[9] and can be dropped from the equations.

2.4.1 HMMs hypothesis

Hypothesis 1 *The speech signal is assumed to be produced by a Hidden Markov Model M built up from a set of K states* $Q = \{q_0, q_1, \cdots, q_K\}$.

As illustrated in figure 2.7, a HMM is a stochastic automaton consisting of a set of states and possible transitions between states. Ideally, each state is associated to an acoustic unit that is modelled by some statistical functions tied to the state. A HMM is associated with a specific word or sentence and obtained by concatenating elementary states. If states are associated to phonemes, a relative low number of states can generate very large vocabularies. As already explained, the nature of the speech signal is suggesting the topology of the HMM used to model speech :

1. The HMM topology is generally left-to-right to illustrate the fact that words and sentences are formed of a priori succession of acoustic units. A HMM is then made up of L states $q_l \in Q$ and transitions between those states. Since a particular acoustic unit may occur several times in a word or sentence, the same state may also occur several times.

2. Self-transition state loops are needed to cope with temporal variability of speech unit realisations.

Equations 2.16 and 2.15 have to be re-formulated in terms of HMMs parameters.

[9]Excepted for adaptative systems

$$P(M_i|X,\Theta) = \frac{p(X|M_i,\Theta)P(M_i|\Theta)}{p(X|\Theta)} \tag{2.17}$$

Equation 2.17 splits the posterior probability of the model into :

- $p(X|M_i,\Theta)$ which is the probability of the acoustic sequence given a model and its parameters. This probability is called the *likelihood* in the Bayes theory.
- $P(M_i|\Theta)$ which is the probability of the model given the parameter set. Generally, this probability is modelled with a different set of parameters $\Theta^\star$ since it represents the probability of a sentence that can be estimated independently of what is happening at the acoustic level. This probability is called the *prior* probability of the model, since it represents the knowledge we would have if no acoustic information X was available.
- $p(X|\Theta)$ which is the probability of the acoustic vector sequence given the parameter set, without making any hypothesis regarding the model.

Equation 2.17 puts in evidence the so-called language model $P(M_i|\Theta) = P(M_i|\Theta^\star)$ and acoustic model $\frac{p(X|M_i,\Theta)}{p(X|\Theta)}$, which are generally modelled and trained apart.

To a particular acoustic vector sequence X of length N corresponds an HMM in which a set of state paths are permitted. A particular state path is an ordered sequence $Q_1^N = \{q^0, q^1, \ldots, q^N, q^{N+1}\}$ in which q^n represents the HMM state at time n. Initial and final states $q^0 = q_I$ and $q^{N+1}q_F$ are non-emitting states, defined to initialize recursions. If we let q_k^n means that specific state q_k is visited at time n, the likelihood $p(X|M_i,\Theta)$ can be written as :

$$P(X|M_i,\Theta_i) \;=\; \sum_{\ell_1=1}^{L} \ldots \sum_{\ell_N=1}^{L} P(X, q_{\ell_1}^1, \ldots, q_{\ell_N}^N|M_i,\Theta_i) \tag{2.18}$$

which expresses the likelihood as the sum, over all possible paths of length N in the hidden process, of the probability of the observable process along a particular path. Eq. 2.18 assumes that each state can be reached from any state, which is most of the time not true. Anyway, the formalism holds since a forbidden state sequence has a null probability in Eq. 2.18 [10]. If we consider a particular state sequence $q_{\ell_1}^1, \ldots, q_{\ell_N}^N$ in equation 2.18, we can write

$$\begin{aligned} P(X, q_{\ell_1}^1, \ldots, q_{\ell_N}^N|M_i) \;&=\; P(X, q_{\ell_1}^1, M_i)P(X, q_{\ell_2}^2|q_{\ell_1}^1|M_i)\ldots \\ &\quad P(X, q_{\ell_N}^N|q_{\ell_1}^1, \ldots, q_{\ell_{N-1}}^{N-1}, M_i) \end{aligned} \tag{2.19}$$

Hypothesis 2 *First order Markov process (independence property of the hidden process).*

The *first order Markov process* hypothesis can be interpreted as follows : knowing the state occupancy at time $(n-1)$, the future state occupancy $(n \rightarrow N)$ is independent of the past $(1 \rightarrow n-2)$. Equation 2.19 can then be re-written as :

$$\begin{aligned} P(X, q_{\ell_1}^1, \ldots, q_{\ell_N}^N|M_i) \;&\simeq\; P(X, q_{\ell_1}^1|M_i)P(X, q_{\ell_2}^2|q_{\ell_1}^1, M_i)\ldots \\ &\quad P(X, q_{\ell_N}^N|q_{\ell_{N-1}}^{N-1}, M_i) \end{aligned} \tag{2.20}$$

Hypothesis 3 *Observation independence. The sub-sequence of observations $X_{n-c}^{n+d} = \{x_{n-c}, \ldots, x_n, \ldots, x_{n+d}\}$ at time n depends only on the state $q_{\ell_n}^n$ visited at time n.*

[10] This fact will even become clearer when transition probabilities will be introduce later on. In case of forbidden state transition, the corresponding state transition probability is simply set to 0.

Equation 2.20 can then be re-written as :

$$P(X, q_{\ell_1}^1, \dots, q_{\ell_N}^N | M_i) \simeq \prod_{n=1}^{N} P(X_{n-c}^{n+d}, q_{\ell_n}^n | q_{\ell_{n-1}}^{n-1}, M_i) \tag{2.21}$$

If we consider the non-stationary nature of the speech signal over long periods, hypothesis 3 is not so far from the reality. Indeed, it tells us that what is observed before time $n-c$ and after time $n+d$ is not related to what is happening in state $q_{\ell_n}^n$ visited at time n, which is reasonable because states are tied to acoustic events that span over limited time windows.

The sub-sequence X_{n-c}^{n+d} context is in practice often limited in size with $c = d = 0$, giving $X_{n-c}^{n+d} = x_n$. This is an almost necessary step to make discrete or multigaussian systems tractable because the emission on transition probabilities of equation 2.23 are multivariate probability density functions that are difficult to compute if the dimension of the vector X_{n-c}^{n+d} is too high [11]. On the other hand, x_n is often complemented with the so-called delta and delta-delta coefficients (eq. 2.13), which can be seen as an indirect way to incorporate a larger context without increasing too much the vector dimension. Starting from these considerations and for sake of clarity in the notations, we will assume in the rest of the text that the context is limited to one frame, setting $X_{n-c}^{n+d} = x_n$, even if this is not completely true when delta coefficients are included. Equation 2.21 becomes

$$P(X, q_{\ell_1}^1, \dots, q_{\ell_N}^N | M_i) \simeq \prod_{n=1}^{N} P(x_n, q_{\ell_n}^n | q_{\ell_{n-1}}^{n-1}, M_i) \tag{2.22}$$

If we come back to the expression of $P(X|M_i)$ in equation 2.18 in which we apply equation 2.22, we get

$$\begin{aligned} P(X|M_i) &= \sum_{\ell_1=1}^{L} \dots \sum_{\ell_N=1}^{L} \prod_{n=1}^{N} P(x_n, q_{\ell_n}^n | q_{\ell_{n-1}}^{n-1}, M_i) \\ &= \sum_{\ell_1=1}^{L} \dots \sum_{\ell_N=1}^{L} \prod_{n=1}^{N} \underbrace{P(x_n | q_{\ell_n}^n, q_{\ell_{n-1}}^{n-1}, M_i)}_{em.\ on\ trans.\ probs} \underbrace{P(q_{\ell_n}^n | q_{\ell_{n-1}}^{n-1}, M_i)}_{trans.\ probs} \end{aligned} \tag{2.23}$$

in which

- $P(x_n | q_{\ell_n}^n, q_{\ell_{n-1}}^{n-1}, M_i)$ are defined as the *emission on transition probabilities* and represents the probability to observe a feature vector x_n at time n when transiting from state $q_{\ell_n}^n$ to state $q_{\ell_{n-1}}^{n-1}$.
- $P(q_{\ell_n}^n | q_{\ell_{n-1}}^{n-1}, M_i)$ is defined as the *transition probability* and represents the probability to go from state $q_{\ell_{n-1}}^{n-1}$ to state $q_{\ell_n}^n$ at time n.

Hypothesis 4 *The current acoustic vector x_n at time n depends only to the current state $q_{\ell_n}^n$.*

Although not necessary, this hypothesis is often done in order to limit the number of parameters. Equation 2.23 becomes :

$$P(X|M_i) = \sum_{\ell_1=1}^{L} \dots \sum_{\ell_N=1}^{L} \prod_{n=1}^{N} \underbrace{P(x_n | q_{\ell_n}^n, M_i)}_{em.\ probs} \underbrace{P(q_{\ell_n}^n | q_{\ell_{n-1}}^{n-1}, M_i)}_{trans.\ probs} \tag{2.24}$$

[11] Multivariate probability density functions taking as input high dimensional vectors are uneasy to estimate because larger amount of training data are required and because probability values are getting very small. leading to problems of computer precision.

$P(x_n|q^n_{\ell_n}, M_i)$ is the so-called *emission probability* and represents the probability to observe a feature vector in time n when visiting state $q^n_{\ell_n}$. In comparison with the previous equation 2.23, this formulation implies that emission probability density functions are tied to states instead of being tied to transitions. Since there are generally more transitions than states in a HMM, there is a subsequent reduction of the number of probability density functions with this last formulation.

Hypothesis 5 *Independence of acoustical pieces of information in the acoustic vector x_n.*

As explained in section 2.3, the acoustic vector x_n includes cepstral coefficients c_n, sometimes appended to their derivatives Δc_n and $\Delta\Delta c_n$, and to the log-energy e_n. In the case of discrete or continuous multigaussian HMMs (explained later), it is often preferable to make the assumption that these different pieces of information are independent.

$$\begin{aligned} P(X|M_i) &= \sum_{\ell_1=1}^{L} \dots \sum_{\ell_N=1}^{L} \prod_{n=1}^{N} P(c_n|q^n_{\ell_n}, M_i) P(\Delta c_n|q^n_{\ell_n}, M_i) \\ & \quad P(\Delta\Delta c_n|q^n_{\ell_n}, M_i) P(q^n_{\ell_n}|q^{n-1}_{\ell_{n-1}}, M_i) \end{aligned} \tag{2.25}$$

2.4.2 Forward and backward recursions

The direct computation of the likelihood $P(X|M_i)$ as in Eq. 2.25, i.e. summing over all possible state sequence, is for most practical situation infeasible, even with small values of L and N. Indeed, assuming a fully connected HMM (*ergodic* HMM), i.e. a HMM in which a state can be reached by any state, there are L^N possible state sequences. For example, if $L = 8$ and $N = 100$, about 10^{90} terms of Eq. 2.25 have to be computed. Fortunately, an efficient dynamic programming recursion has been formulated to compute Eq. 2.25. Let us define the *forward probability* by:

$$\alpha_n(\ell) = p(X^n_1, q^n_\ell|M_i) \tag{2.26}$$

that is the joint probability of observing the partial vector sequence $X^n_1 = \{x_1, \dots, x_n\}$ and having arrived in state l at time n, given the HMM M_i. The forward probability equals the likelihood $P(X|M_i)$ of Eq. 2.25 where $\alpha_{N+1}(F) = p(X^N_1|M_i)$ for the final state q_F. Assuming the hypothesis that have been stated before, the following recursion can be written as :

$$\alpha_n(\ell) = \left[\sum_{k=1}^{L} \alpha_{n-1}(k) p(q_\ell|q_k)\right] p(x_n|q_\ell) \tag{2.27}$$

The forward recursion is computed from a lattice as described in figure 2.8 where each point corresponds to a an acoustic vector x_n on the X-axis and to a state q_l on the Y-axis. The forward recursion propagates from left to right to compute the $\alpha_n(l)$ probabilities. The calculation of $P(X|M_i)$ requires about L^2N operations (assuming a fully ergodic HMM and known emission probabilities) which is much less than the exhaustive computation over all possible paths.

In a similar manner, we can consider a *backward probability*

$$\beta_n(\ell) = p(X^N_{n+1}|q^n_\ell, M_i) \tag{2.28}$$

$$= \sum_k [\beta_{n+1}(k) P(q_k|q_\ell) p(x_{n+1}|q_k)] \tag{2.29}$$

that is, the probability of the partial observation sequence from $n+1$ to the end, given state l at time n and te model M_i. Again, we can solve backward probabilities inductively as in Eq. 2.29. Backward probabilities will be used later on to derive the HMM parameters reestimation formulae.

Continuous speech recognition with large vocabulary can be achieved using the stack decoding algorithm (Jelinek 1969; Bahl, Jelinek, and Mercer 1983). It is derived from the A^* algorithm (Nilsson 1980) and implements a modification of the forward algorithm leading to a best-first search strategy .

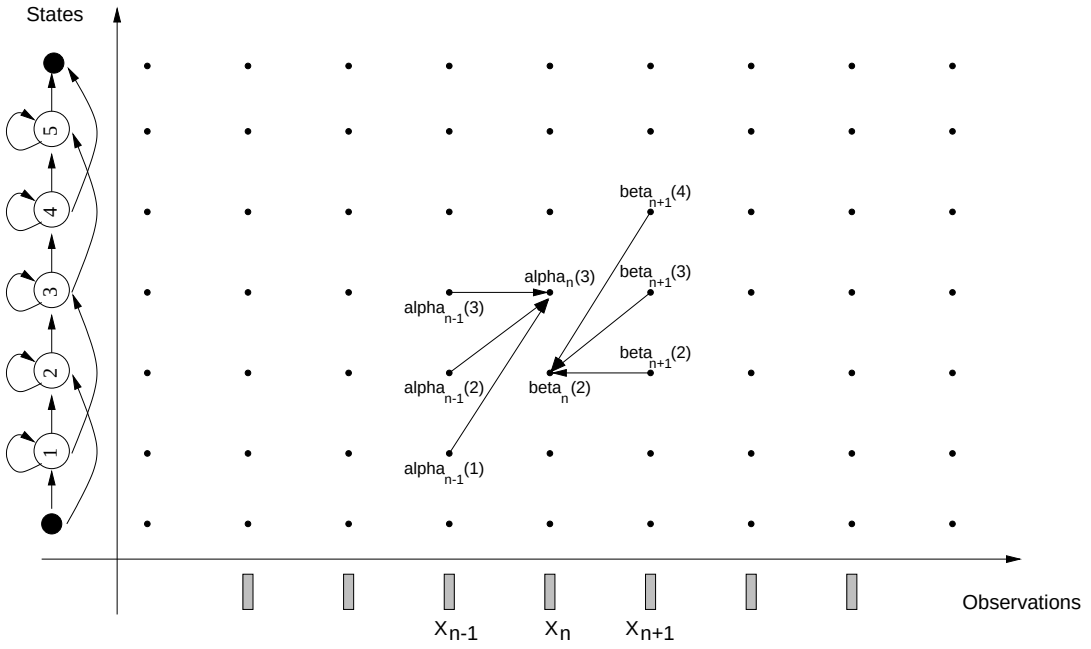


Figure 2.8: Forward and backward recursion. The HMM topology is left-right but permits in this case to skip the next state and to go to the second next state. The forward recursion propagates from left to right to compute the α probabilities. For example, the lattice point corresponding to $\alpha_n(3)$ can only be reached from states q_1, q_2 and q_3, and the recursion in this point is $\alpha_n(3) = [\alpha_{n-1}(1)p(q_3|q_1) + \alpha_{n-1}(2)p(q_3|q_2) + \alpha_{n-1}(3)p(q_3|q_3)]\, p(x_n|q_3)$. The backward probabilities are computed in a similar manner but propagating the recursion from right to left. Any HMM topology can be taken into account, the beam of forward/backward connections getting larger with more intricated topologies.

2.4.3 Parameter reestimation

The HMM parameters are the transition probabilities and the emission probabilities of Eq. 2.24. We are here in the case of supervised learning because we know what has been said in the speech signal and therefore, we know to which HMM it corresponds[12].

An iterative procedure known as the Baum-Welch method (or forward-backward or Expectation-Maximisation (EM)) can be used to locally maximize the likelihood $P(X|M_i)$. Let us define the following probabilities :

- $\gamma_n(k) = P(q_k^n|X_1^N, M_i)$ the probability of being in state q_k at time n given the entire observation sequence and the model.
- $\xi_n(k,l) = P(q_k^n, q_l^{n+1}|X_1^N, M_i)$ the probability of being in state k at time n and state j at time $n+1$, given the entire observation sequence and the model.

It is easily shown that $\gamma_n(k)$ and $\xi_n(k,l)$ can be expressed in terms of the forward $\alpha_n(k)$ and backward $\beta_n(k)$ probabilities as follows :

$$\gamma_n(k) = \frac{\alpha_n(k)\beta_n(k)}{\sum_{\ell=1}^{L}\alpha_n(\ell)\beta_n(\ell)} \tag{2.30}$$

$$\xi_n(k,l) = \frac{\alpha_n(k)p(q_l|q_k)p(x_{n+1}|q_l)\beta_{n+1}(k)}{\sum_{k=1}^{L}\sum_{\ell=1}^{L}\alpha_n(k)p(q_l|q_k)p(x_{n+1}|q_l)\beta_{n+1}(k)} \tag{2.31}$$

$$\tag{2.32}$$

$\gamma_n(k)$ are defined for each point of the lattice of figure 2.8 and can be viewed as a *membership* measure of observation x_n in state q_k. They can also be interpreted as *soft segmentation* (against the hard segmentation of the Viterbi criterion) of the vector sequence into states. If we sum $\gamma_n(k)$ over time index n, we get the expected (over time) number of times that state q_k is visited. In a similar manner, if we sum $\xi_n(k,l)$ over time, we get the expected number of times a transition from state k to state l is taken.

- Emission probabilities are reestimated for each state q_k as it is usually done in statistical inference, i.e. deriving parameter values from a population of measure vectors, excepted that now each vector x_n will be weighted by its corresponding $\gamma_n(k)$ value, indicating "how much" it belongs to state q_k.
- The reestimated transition probabilities are equal to the expected number of transitions from state k to state l, divided by the expected number of time state k is visited :

$$\overline{p}(q_l|q_k) = \frac{\sum_{n=1}^{N}\xi_n(k,l)}{\sum_{n=1}^{N}\gamma_n(k)} \tag{2.33}$$

The reestimation procedure is as follows :

1. Compute $\gamma_n(k)$ and $\xi_n(k,l)$ for all feature vectors and states in the training database.
2. Reestimate transition and emission probabilities.
3. Iterate in 1 until the likelihood score $P(X|M_i)$ accumulated for each training utterance in the database saturates.

[12] Equivalently, for speaker recognition systems, we know during training who is speaking and therefore we can choose the corresponding model to train.

2.4.4 Viterbi criterion

The likelihood $P(X|M_i)$, when computed with the forward recursion, takes into account all the possible state sequences through the HMM. The Viterbi criterion considers only the most probable path to estimate $P(X|M_i)$. Therefore, it can be viewed as an approximation of the forward method. Eq. 2.18 becomes :

$$\overline{P}(X|M_i) = \max_{\ell_1,\dots,\ell_N} P(X, q^1_{\ell_1}, \dots, q^N_{\ell_N}|M_i) \tag{2.34}$$

and the forward Viterbi recursion is :

$$\begin{aligned} \overline{\alpha}_n(\ell) &= p(X^n_1, q^n_\ell|M_i) \\ &= \max_k \left[\alpha_{n-1}(k)p(q_\ell|q_k)\right] p(x_n|q_\ell) \end{aligned} \tag{2.35}$$

The induction in Eq. 2.35 can be used to retrieve the best path trough the set of states, keeping track of the argument k that maximized Eq. 2.35 :

$$\psi_n(\ell) = \arg\max_k \left[\ \alpha_{n-1}(k)p(q_\ell|q_k)\right] \tag{2.36}$$

Viterbi recursion can easily be implemented in the log domain where no multiplications are needed to compute 2.35. The Viterbi criterion is often used to compute a hard *segmentation* of the acoustic vector sequence and hence to define automatically phoneme stretches on the speech signal. The term *alignment* is also used to express the fact that the acoustic vector sequence is aligned to the the state sequence. An example of alignment is given in Fig. 2.9.

Since the best path trough the set of states can be known with Eq. 2.36, the probability $P(q^n_k|X^N_1, M_i)$ of being in state q_k at time n given the entire observation sequence and the model is either 1 or 0, depending if state q_k is visited or not at time n. Reestimation following the Viterbi criterion can then be performed as explained in section2.4.3, setting the $\gamma_n(k)$ values to 1 or 0 looking at the best path. Training using the Viterbi criterion is sometimes known as the *segmental k-means* algorithm (Rabiner, Wilpon, and Soong 1989).

For continuous speech recognition, a full Viterbi search can be time consuming. One solution is to prune out candidates that are less likely (for which the accumulated score is below a threshold) for each incoming frame. This technique is called the *beam search* (Ney, Mergel, Noll, and Paeseler 1987).

2.4.5 Probability density functions

Gaussian Mixtures

There are several ways to model the emission probability density function $P(x_n|q_k, M_i)$ of equation 2.24. One way is to make the assumption that the distribution of an acoustic vector can be approximated by a weighted sum of gaussian mixtures :

$$P(x_n|q_k, M_i) = \sum_{j=1}^{J} w_{kj}\mathcal{N}(x_n, \mu_{kj}, \Sigma_{kj}) \tag{2.37}$$

in which J is the number of gaussian mixtures in the probability density function and w_{kj} is the weight for mixture j in state k. Each mixture is parametrized by its mean vector μ_{kj} and its covariance matrix Σ_{kj}. $\mathcal{N}$ has the form :

$$\mathcal{N}(x_N, \mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp\left(-\frac{1}{2}(x_n - \mu)^T\Sigma^{-1}_k(x_n - \mu)\right) \tag{2.38}$$

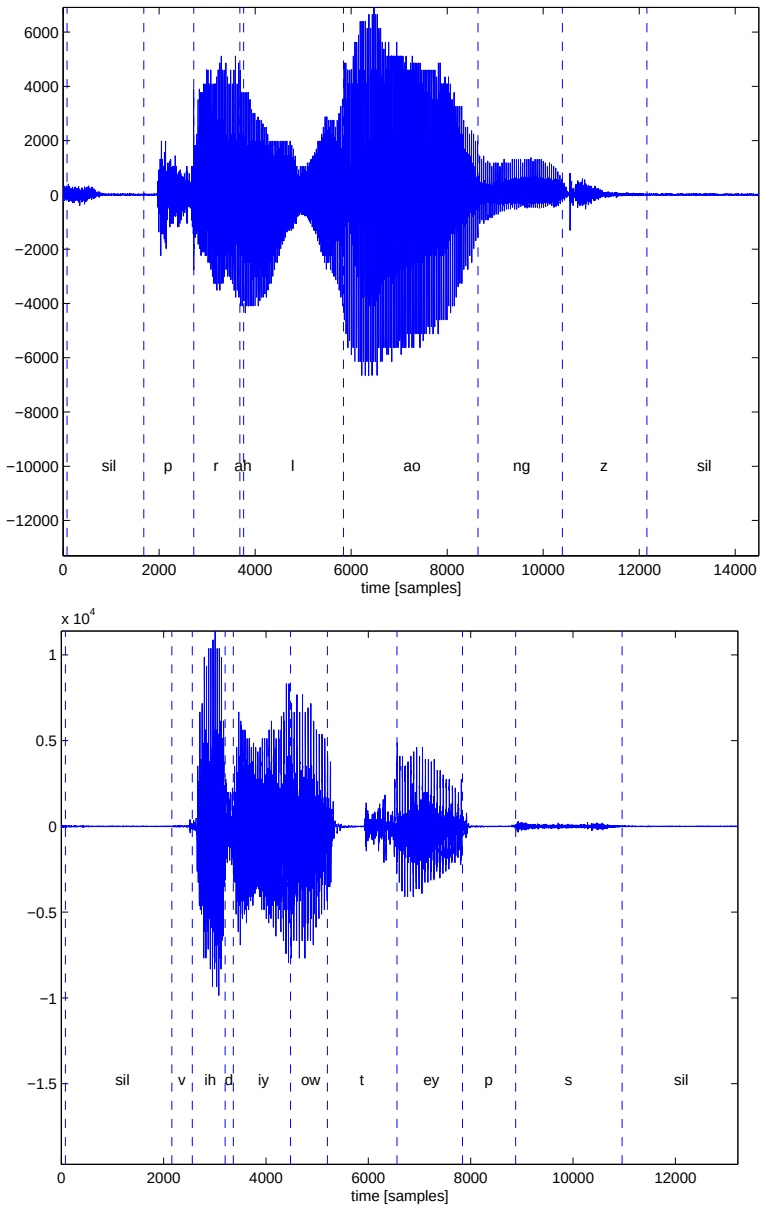


Figure 2.9: Example of Viterbi alignment from the Phonebook database. a) `prolongs` b) `videotapes`.

Since the feature extraction process generally provides us with uncorrelated vector coefficients [13], the covariance matrix can be considered as diagonal and equation 2.38 can be re-written :

$$\mathcal{N}(x_n, \mu, \Sigma) = \prod_{i=1}^{D} \frac{1}{\sqrt{2\pi\sigma_i^2}} exp\left(-\frac{(x_n i - \mu_i)^2}{2}\right) \tag{2.39}$$

Discrete features

Another way to model emission probabilities is to use discretized features. Each acoustic vector x_n is mapped to a discrete value y_{i_n} selected in a predetermined set $\mathcal{Y} = \{y_1, \ldots, y_i, \ldots, y_I\}$. The prototype vector is referred to as *codeword* or *centroid* vector and $\mathcal{Y}$ is referred to as the *codebook*. I is the amount of centroids in the codebook. The emission probabilities of equation 2.24 are modelled by histograms of probabilities $p(y_i|q_k)$ with $1 \leq i \leq I$ tied to each state q_k. Probability values are directly stored in tables wich are looked up at recognition time, leading to very fast processing.

Duration modeling

In case of Viterbi criterion, we can compute state durations (state occupancies) counting the number of frames aligned to the state. For a particular word or utterance u_i, the duration $d_{u_i}(q_k)$ in state q_k is given by :

$$d_{u_i}(q_k) = \sum \frac{N_{q_k}}{N_{u_i}} \tag{2.40}$$

where N_{q_k} is the number of frames aligned in state q_k and where N_{u_i} is the total number of frames in word u_i. If we accumulate these durations over the training database, we can compute histogram *duration* probabilities for each state q_k. At recognition time, it is possible to include duration modeling to enhance the performance of the system. Any duration probability value can be modelled using a HMM topology as illustrated in Fig. 2.10 b, setting transition probabilities from a state to the final state equal to the corresponding duration probability. Another easier way to model durations is to use the so-called *minimum duration modeling* by forcing the frame sequence to stay a given number of time in a state. In this optic, a topology as illustrate in Fig. 2.10 c can be used. Experiments have shown that setting the number of state repetition equal to the expected duration divided by two leads to good performances. Duration modeling is worthwile, especially when using mono-state CIP. Minimum duration actually does not change a lot the score of the model corresponding to the correct transcription but penalizes the incorrect models that tend to skip quickly the CIP giving bad scores. Minimum duration forces the sequence to stay in these incorrect states for a while, penalizing the global score.

2.5 Multilayer Perceptrons

Good introductory books for ANN are (Hertz, Krogh, and Palmer 1991) and (Bishop 1995). An artificial neuron is the building block of a neural network. Its behaviour, very loosely based on the brain's nerve cell, is illustrated on figure 2.11 : neurons receive inputs x_i via weighted links w_i from other neurons, that are summed and processed according to the neuron activation function $f(a)$. Signals are then passed on to other neurons. The output g of a neuron is expressed as :

$$g = f(a) = f(\sum_i w_i x_i) \tag{2.41}$$

There are three different types of neurons within a neural net. Input neurons receive encoded information from the external environment. Output neurons send signals out to the external environment in the

[13] This is almost the case for common feature extraction techniques like lpc-ceptra or mfcc.

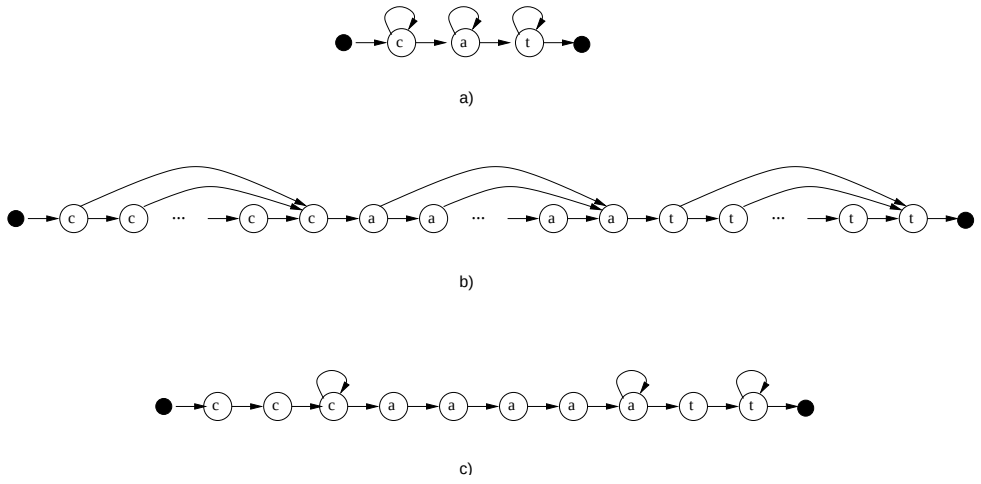


Figure 2.10: Duration modeling with HMMs. a) Topology with no duration modeling. b) Histogram type duration modeling in which transition probabilities between a state and a final phone is set equal to a particular duration probability. c) Minimum duration modeling in which a state is forced to be visited a given number of time.

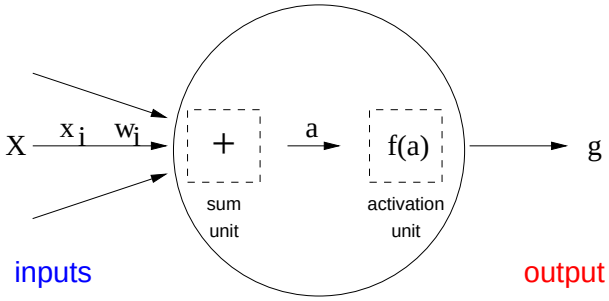


Figure 2.11: Artificial neuron or perceptron : basic processing unit which performs a non-linear operation on the weighted sum of its inputs.

form of an encoded answer to the problem presented in the input. Hidden [14] neurons allow intermediate calculations between inputs and outputs.

The activation function $f(a)$ is generally non-linear by displaying saturation when inputs are either too large or too small. The activation function can also be viewed as a decision function for which the outputs will be large when the weighted inputs are above a threshold value. The training procedure which is explained later requires a differentiable function for the activation function. Either a 0/1 or a ± 1 can be obtained with, respectively, a *sigmoid* or a *tanh* function (see figure 2.12) :

$$f(a) = \frac{1}{1+\exp(-x)} \tag{2.42}$$

$$f(a) = \tanh(a) \tag{2.43}$$

The best known neural network model which is often used as a synonym when speaking about neural networks is the *multi-layer perceptron* (MLP). MLPs are extensively used in this work and we will focus

[14] "Hidden," in neural network jargon, indicates that at run time, this neuron's activity is not apparent to the user, for whom the net behaves like a black box. The same nomenclature holds for layers of neurons, as described after.

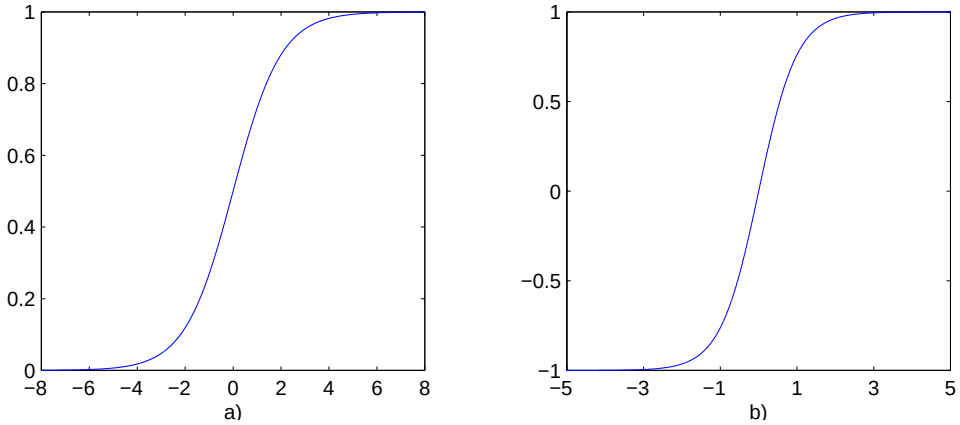


Figure 2.12: Activation functions. a) Sigmoid function $f(a) = \frac{1}{1+\exp(-x)}$. b) Hyperbolic tangent $f(a) = \tanh(a)$.

on them in this fundamental description. Nevertheless, we should underline that other neural networks exist and will be introduced when needed in the next chapters. The MLP (see figure 2.13) is composed of hierarchical layers of neurons arranged so that information flows from the input layer to the output layer of the network, i.e. no feedback connections are allowed. A layer has all its inputs connected to either a preceding layer or to the external world. A layer has all its outputs connected to either a succeeding layer or to the external world. A layer is said to be hidden if it has no input or output connections to the external world.

2.5.1 Propagation

MLPs have generally at least three layers of neurons (i.e., ones with input neurons, hidden neurons and output neurons). It has been prooved that *any* decision boundary can be implemented with these three layers. This feature is partially explained by Kolmogorov and others and is discussed in appendix B, section B.4. A neural network with one so-called hidden layer in the middle can actually be viewed as an universal approximation function. In the remaining of the text, we will consider three-layers MLP as depicted in figure 2.14. The notations are the following. Subscripts i, j and k denote respectively neurons from the input, hidden and output layers. Similarly, connection weights will be noted w_{ij} for input-to-hidden nodes and w_{jk} for hidden-to-output nodes. Activations are noted a_j for neurons on the hidden layer and a_k for neurons on the output layer. Neuron's outputs are noted g_j and g_k for hidden and output layers. Taking into account the single neuron's equation 2.41, the *propagation* function $F_k(x, w)$ of a three layers MLP (figure 2.14) is given by:

$$F_k(x,w) = f(a_k) = f\left(\sum_j g_j w_{jk}\right) = f\left(\sum_j f(a_j) w_{jk}\right) = f\left(\sum_j f\left(\sum_i x_i w_{ij}\right) w_{jk}\right) \tag{2.44}$$

We have omitted any activation bias (threshold) from Eq. 2.41 and 2.44 because they can always be treated as connections from additional neurons whose output is always 1, on all layers as depicted in figure 2.15.

2.5.2 Backpropagation

The behaviour of artificial neural networks is determined by their weight values. A common algorithm in learning of weights in multi-layer networks is the so-called *stochastic backpropagation*. As explained in

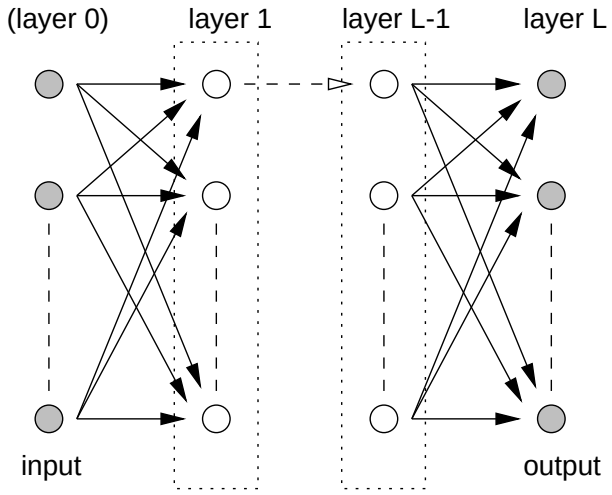


Figure 2.13: Muti-layer perceptron with L layers of neurons.

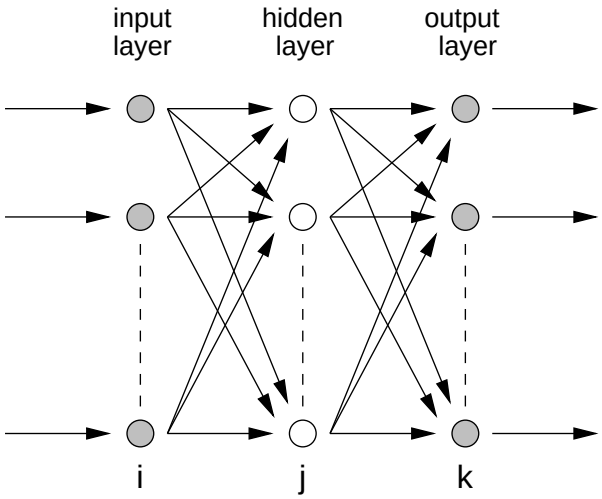


Figure 2.14: Muti-layer perceptron with 3 layers. The role of the input layer is to distribute the input features. No computation is actually performed at this stage. Hidden and output layers are computational. Subscripts i, j and k denote respectively neurons from the input, hidden and output layers. Similarly, connection weights will be noted w_{ij} for input-to-hidden nodes and w_{jk} for hidden-to-output nodes.

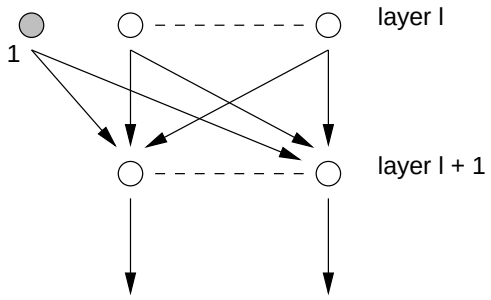


Figure 2.15: Bias can be treated as connection weights from an additional neuron whose output is clamped to 1.

(Rumelhart, Hinton, and Williams 1986), the parameters of the MLP (weight matrices) are iteratively updated via a gradient descent procedure in order to minimise an error criterion. The approach proceeds in two steps. First an input pattern x_n randomly sampled [15] in the training set is presented to the network and the output is determined. Secondly, based on an error measure between the actual output g_n and the desired *target* output t_n, the weights are adjusted to reduce the error. A common error measure for a pattern n is the sum over the outputs of the squared difference between the "target" output vector t and the actual "gotten" output vector g_k :

$$E_n = \frac{1}{2}\sum_k (t_{nk} - g_{nk})2 = \frac{1}{2}(t_n - g_n)2 \tag{2.45}$$

Other error measures can be defined but we shall use this one to explain the backpropagation algorithm. The average of Eq. 2.45 computed on the training set is called the mean square error (MSE). Equations may differ with another error measure but the principle is the same. The learning rule is based on a simple gradient descent changing the weight values in a direction that will reduce the error measure:

$$\Delta w_{ij} = -\eta \frac{\partial E_n}{\partial w_{ij}} \tag{2.46}$$

$$\Delta w_{jk} = -\eta \frac{\partial E_n}{\partial w_{jk}} \tag{2.47}$$

where η is the *learning rate* and indicates the relative size of the change in weights. Weights are updated at each presentation of one input pattern as follows :

$$w(n+1) = w(n) + \Delta w(n) \tag{2.48}$$

A simple application of the chain rule, as developped in appendix B section B.1, allows to write the weights' corrections as :

$$\Delta w_{jk} = \eta \delta_k g_j = \eta (t_k - g_k) f'(a_k) g_j \tag{2.49}$$

$$\Delta w_{ij} = \eta \delta_j g_i = \eta g_i f'(a_j) \sum_k \delta_k w_{jk} \tag{2.50}$$

where δ_k and δ_j are the *sensitivities* describing how the error changes with the neuron's activation :

[15] n represents here the index of an input pattern chosen randomly in the training set.

$$\delta_k \equiv -\frac{\partial E}{\partial a_k} = (t_k - g_k) f'(a_k) \tag{2.51}$$

$$\delta_j \equiv -\frac{\partial E}{\partial a_j} = -\frac{\partial E}{\partial g_j}\frac{\partial g_j}{\partial a_j} = f'(a_j)\sum_k \delta_k w_{jk} \tag{2.52}$$

2.5.3 Posterior probability estimation

When dealing with classification tasks, a *1-from-K* training is usually performed, i.e. MLP outputs are associated to classes and target values are set to :

$$t_k = \begin{cases} 1 & \text{if } x \in \omega_k \\ 0 & \text{otherwise} \end{cases} \tag{2.53}$$

In this case, it has been shown (Bourlard and Wellekens 1990) (Bourlard and Wellekens 1988) that a minimization of the sum-of-square error functions 2.45 makes the network approximate the posterior probabilities $P(\omega_k|x)$ of class membership conditioned on the input variables. Further discussion and mathematical proof of this can be found in appendix B section B.3. The proof is valid provided some conditions :

- The network need to be able to represent the functions $P(\omega_k|x)$ and thus, enough hidden neurons have to be used.
- An infinite amount of training data is required, which is never true in practice.
- The gradient descent has to be optimal and can't fall into local minima.
- The network has to be trained with target values set as in Eq. 2.53. If target values are set equal to the actual posterior probabilities, the proof is still valid.

The proof holds for training with other error function (including relative entropy for example). Output activation functions should be constrained to be nonnegative and less than one.

The training is said to be *discriminative* because it minimizes the likelihood of incorrect models (through the zeros of the target vector) and maximizes the likelihood of the correct model. The network attempts to model class boundaries, rather than accurate probability density functions for each class.

2.5.4 Input Scaling

By applying a linear transformation, we can arrange for the inputs to have normalized values. This is often useful during training time if input patterns have component values wich differ significantly. To do this, we consider each of the input vector value x_i independently and calculate its mean $\overline{x_i}$ and variance σ_i^2 going through the training set :

$$\overline{x_i} = \frac{1}{N}\sum_{n=1}^{N} x_i^n \tag{2.54}$$

$$\sigma_i^2 = \frac{1}{N-1}\sum_{n=1}^{N}(x_i^n - \overline{x_i})^2 \tag{2.55}$$

where N denotes the total number of training vectors. The normalized values are then given by

$$\hat{x}_i^n = \frac{x_i^n - \overline{x_i}}{\sigma_i} \tag{2.56}$$

It can be argued that such a normalization is useless in the case of MLP. Indeed, the transformation of Eq. 2.56 is linear and could be in principle performed in the first layer of the network. However, in practice, the weights are initialized randomly prior to any training, which makes this normalization favorable for some good reasons:

1. Speeded up training has been reported.
2. Problems of local minima are avoided.
3. Problems of limited precision for weights representation are avoided (in case of fixed point implementation).

2.5.5 Weight and bias initialization

The weights are usually initialized randomly, taken to have zero mean since there is no reason to prefer any specific points in weight space. The choice of the initial weight variance is set such that saturation of neurons is avoided. Indeed, $f'(a)$ can be very small in the saturation regions, which would lead to small Δw in Eq. 2.49 and 2.50, slowing down the training procedure. More complicated weight initialization techniques are discussed in (Bishop 1995).

(Bourlard and Morgan 1994) suggest the following initialization of the output neuron biases to improve convergence of the backpropagation algorithm :

$$w_{0k} = \log\left(\frac{P(\omega_k)}{1-\omega_k}\right) \tag{2.57}$$

This can be explained intuitively as follows. If we were taking decisions without any pattern measure x, we would declare winner the class that has the largest prior probability $p(\omega_k)$. This is suggesting that our network should give prior probabilities as output if not trained. Given this and using a sigmoid as activation function at the output of the MLP, biases of the output neurons should be initialized as in Eq. 2.57.

Bibliography

Bahl, L., F. Jelinek, and R. Mercer (1983, March). A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 5*(2), 179–190.

Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Clarendon Press Oxford.

Bourlard, H. and N. Morgan (1994). *Connectionist Speech Recognition*. Kluwer Academic Publishers.

Bourlard, H. and C. Wellekens (1988). Links between markov models and multilayer perceptrons. In D. Touretzky (Ed.), *Advances in Neural Information Processing*, Volume 1, San Mateo CA, pp. 502–510. Moran Kaufmann.

Bourlard, H. and C. J. Wellekens (1990, December). Links between markov models and multi-layer perceptrons. *IEEE Trans. Patt. Anal. Machine Intell. 12*(12), 1167–1178.

Duda and Hart (1973). *Pattern Classification and Scene Analysis*. Wiley and Sons.

Hanson, B. A. and T. H. Applebaum (1990, April). Robust speaker-independent word recognition using static, dynamic and acceleration features: experiments with lombard and noisy speech. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Albuquerque, New Mexico, pp. 857–860.

Hertz, J., A. Krogh, and R. G. Palmer (1991). *Introduction to the theory of Neural Computation*. Santa Fe Institute Studies in the Sciences of Complexity. Addison Wesley.

Jelinek, F. (1969). Fast sequential decoding algorithm using a stack. *IBM Journal of Research and Development 13*, 675–685.

Juang, B.-H., L. R. Rabiner, and J. G. Wilpon (1987). On the use of bandpass filtering in speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 947–954.

Lee, C.-H., E. Giachin, L. Rabiner, R. Pieraccini, and A. E. Rosenberg (1992, April). Improved acoustic modeling for large vocabulary continuous speech recognition. *Computer Speech and Language 6*(2), 103–127.

Morgan, N. and H. Bourlard (1995, May). Neural networks for statistical recognition of continuous speech. *Proceeding of the IEEE 83*(5), 742–770.

Ney, H. (1990, April). Acoustic-phonetic modeling using continuous mixture densities for the 991-word darpa speech recognition task. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Albuquerque, New Mexico, pp. 713–716.

Ney, H., D. Mergel, A. Noll, and A. Paeseler (1987, April). A data-driven organization of the dynamic programming beam search for continuous speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 833–836.

Nilsson, N. (1980). *Principles of Artificial Intelligence*. Palo Alto, CA: Tioga Publishing Co.

Picone, J. (1993, September). Signal modeling techniques in speech recognition. *Proceedings of the IEEE 81*(9), 1214–1247.

Rabiner, L. (1989, February). Tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*(2), 257–285.

Rabiner, L. and B.-H. Juang (1993). *Fundamentals of Speech Recognition*. Prentice Hall.

Rabiner, L., J. Wilpon, and F. Soong (1989). High performance connected digit recognition using hidden markov models. *IEEE Transactions Acoustics Speech Signal Processing 37*, 1214–1225.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland (Eds.), *Parallel Distributed Processing. Exploration in the Microstructure of Cognition*, Volume 1. Massachusetts Institute of Technology Press.

Waibel and Lee (Eds.) (1990). *Readings in Speech Recognition*, Volume 1. San Mateo, CA: Morgan Kaufman Publisher, Inc.

Chapter 3

Self-Organizing Maps and Discrete HMMs

I am not sure how clouds get formed. But the clouds know how to do it, and that is the important thing.

Kid's wisdom

We describe in this chapter an original and successful utilisation of unsupervised ANNs and discrete Hidden Markov modeling for speech recognition systems. Although conceptually weaker than continuous supervised systems like hybrid systems (discussed in chapter 4), the approach presented here offers performance and learning speedup advantages when compared to other (more classic) discrete systems. The unsupervised ANNs which are used here are the so-called Self-Organizing Map (SOM) of Kohonen (Kohonen 1990; Kohonen 1982). The approach has been proposed and analysed in the framework of the Himarnnet ESPRIT project P6488 where discrete systems were required for low-cost implementation of a personal voice mailbox application.

In the first section, a system description is presented. Principles of vector quantization are introduced. The k-means algorithm which is widely used to build vector quantizer is described. SOM of Kohonen are then presented and it is shown that they can be used as vector quantizers in discrete HMMs. Theoretical backgrounds of discrete HMMs are also presented.

Comparison results between the k-means and the Kohonen approaches are reported in the second section. The same recognition platform and databases are used, permitting fair comparisons between the algorithms. SOMs, although not minimizing the distorsion of the quantizer, are shown to outperform k-means. The best performance reported in this study is actually obtained when SOMs are used as a preliminary initialisation step for k-means. This combination of both algorithms performs better than either SOMs or k-means used alone. Influence of some quantization parameters is also investigated.

The last section is dedicated to discussions and extensions of the system. As a possible extension, the Learning Vector Quantization (LVQ) technique is proposed to train further the SOM vector quantizer in a supervised way.

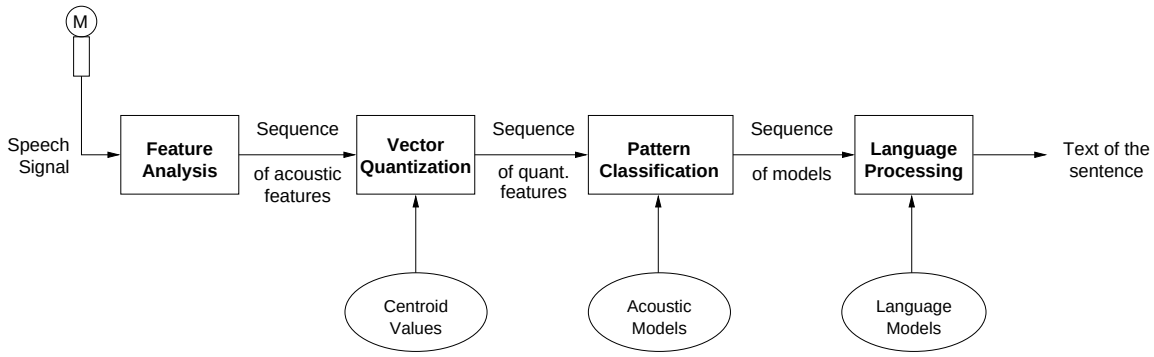


Figure 3.1: Discrete ASR system. A vector quantization is performed at the output of the feature extraction front-end, mapping the acoustic vector sequence into a centroid label vector sequence.

3.1 System description

3.1.1 Vector quantization

As illustrated in Fig. 3.1, a further step is taken in the speech recognition chain in order to use discrete HMMs. A vector quantization is performed at the output of the feature extraction front-end. The acoustic vector x_n is mapped to a discrete value y_{i_n} selected in a predetermined set $\mathcal{Y} = \{y_1, \cdots, y_i, \cdots, y_I\}$. The prototype vector is referred to as *codeword* or *centroid* vector and $\mathcal{Y}$ is referred to as the *codebook* of centroids. I is the amount of centroids in the codebook.

Vector quantization is considered as a data compression technique in the field of speech coding. In this case, the acoustic vector sequence $X = \{x_1, x_2, \cdots, x_N\}$ is replaced by a centroid vector sequence $Y = \{y_{i_1}, y_{i_2}, \cdots, y_{iN}\}$ in which only the labels i_n are transmitted to the receiver.

Vector quantization has also been increasingly applied to reduce problem complexity like pattern recognition. In the case of speech recognition based on HMMs, the emission probabilities of equation 2.24 are computed on the discrete values y_{i_n}. Discrete HMM systems are essentially developed to build up low-cost real-time systems because they are much less CPU consuming than continuous systems.

The mapping operation from a continuous space into a finite set of I codewords generates a *quantization error* or *distortion*. The distortion value must follow a subjective quantization quality. A bad quantization should be represented by a high distortion and vice-versa. In our case, we are using cepstrum feature vectors computed as explained in section 2.3. Cepstrum are naturally decorrelated and therefore, we can adopt an Euclidean distance as distorsion measure for our quantizer [1]:

$$d^2(x, y) = \sum_{d=1}^{D} [x_d - y_d]^2 = (x - y)^t (x - y) \tag{3.1}$$

Often, a weighted Euclidean distance is preferred, leading to better performances of the discrete HMM (Rabiner and Juang 1993) :

$$d_w^2(x, y) = \sum_{d=1}^{D} [w(d)x_d - w(d)y_d]^2 \tag{3.2}$$

where $w(d)$ is the *liftering* window defined in Eq. 2.10. We refer to section 2.3 for a discussion on the liftering procedure.

[1] Of course, a distorsion can be measured differently depending to what has to be considered as (dis)similar in the feature space.

Vector quantization is usually performed comparing each test vector with all centroids to determine the nearest neighbour. The input vector is then assigned to the code vector y_b with the lowest distortion :

$$y_b = \min_i d(x, y_i) \tag{3.3}$$

Vector quantizers that follow the rule of Eq. 3.3 are called *nearest neighbour quantizers* or *Voronoi* quantizers. The quality of a codebook $\mathcal{Y}$ is measured with the average distorsion :

$$D_{\mathcal{Y}} = E[d(x, y_b)] = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} d(x_n, y_b) \tag{3.4}$$

In practice, the average distorsion $D_{\mathcal{Y}}$ is estimated on a finite population. One must be careful when using the average distorsion as comparison criterion of codebook quality because the distorsion is not directly related to speech/speaker recognition error rates. As it will be shown later, a larger distorsion does not necessarily imply a larger error rate and vice-versa.

Voronoi quantizers are known to be optimal in the sense of minimizing the average distortion $D_{\mathcal{Y}}$. If we define by R_i the cell region of the space where each vector is quantized by the same code vector, we can compute a partition of the parameter space into I cells for a given codebook :

$$R_i = \{x : d(x, y_i) < d(x, y_j)\} \qquad \forall j \neq i \tag{3.5}$$

Cells R_i are called the Voronoi cells. Inversely, an optimal codebook can be computed if the Voronoi cells are given. In this case, code vectors will correspond to the mass centre of the I cells for the considered distorsion.

3.1.2 Lloyd Algorithms

Codebook creation is usually performed using algorithms based on iterative methods providing an explicit minimization of the average distorsion (Gersho and Gray 1992).

K-means algorithm

The k-means or *generalized Lloyd*[2] algorithm improves iteratively the codebook as follows :

1. Start with I centroids chosen randomly in the input space.
2. Given the codebook, compute the Voronoi cells R_i on the training points, as explained in Eq. 3.5.
3. Given the resulting partition, find the optimal codebook computing new estimates of the centroid positions in each partition R_i :

$$y_i = \frac{1}{N_i} \sum_{x_n \in R_i} x_n \tag{3.6}$$

   where N_i is the number of training points in cell R_i.
4. Iterate in 2 until the average distorsion $D_{\mathcal{Y}}$ vary relatively less than $\varepsilon\%$ from one iteration to the next.

It can be proved that each iteration 2-3 will reduce monotonically the average distorsion (Gersho and Gray 1992). It should be noted that this iterative procedure will not necessarily converge to the optimum codebook but may converge to a local minimum of the distorsion. The initial codebook is of great importance for the convergence of the algorithm and therefore for the quantizer quality.

[2] A confusion exists in the literature regarding the denomination of the algorithms. For instance, in data compression, the LBG algorithm often refers to the generalized Lloyd algorithm while it is actually an extension to this one. We are using here the nomenclature given in (Gersho and Gray 1992).

K-means with splitting

The k-means algorithm as described above may go across convergence problems due to bad initialization of the centroids. For example, if centroids are all initialized out of the training cluster, it may happen that some centroids will stay apart of the cluster, without being associated to any input vectors. The convergence actually get stuck in a local minimum and the resulting average distorsion is large. In order to have a better initialization and to avoid those problems, centroids can be sequentially included in the codebook as and when the distorsion falls below a pre-defined threshold. The procedure is continued until enough centroids are included in the codebook. For example, to design a codebook containing $I = 2^B$ centroids, the following algorithm can be applied:

1. Start with one centroid equals to the average of the training vectors.
2. Split each centroid of the codebook into two new centroids by adding small noise values.
3. Apply k-means as above until the average distorsion stabilizes.
4. Goto 2 until there are enough centroids in the codebook.

Codebooks of any size can be obtained, splitting only some chosen centroids instead of splitting the whole codebook. For example, centroids associated to the largest number of vectors or those associated to the largest cell distorsion can be split.

LBG algorithm

Linde, Buzo and Gray (LBG) developed a binary tree-structured quantization algorithm (Y. Linde 1980), which is as follows.

1. Start with one centroid equals to the average of the training vectors. This centroid is the root of the tree.
2. Split each leaf of the tree into two branches by adding small noise values.
3. Compute partitions R_{f1} and R_{f2} for each leaf of the tree, using (and only using) the training points of the father node's partition R_f. We have $R_{f1} \subseteq R_f$, $R_{f2} \subseteq R_f$ and $R_{f1} \cup R_{f2} \equiv R_f$.
4. Compute new positions for the two centroids y_{f1} and y_{f2} and iterate in 3 until the average distorsion stabilizes.
5. Goto 2 and make grow the tree until there are enough centroids in the codebook.

Vector quantization can then be performed browsing the tree starting from the root and computing step by step the closest leaves. The closest centroid is found with $\log_2(I)$ distance measure instead of I distance measure in the case of the full search. One problem with this algorithm is that the whole quantized space is not optimized at each iteration. Indeed, k-means is applied only to sub-spaces in order to guarantee to find the nearest centroid when going through the tree. As a consequence, this algorithm is very sensitive to initial conditions and results generally to higher average distorsion.

3.1.3 Self-Organizing Maps of Kohonen

Kohonen Self Organising Maps (SOM) (Kohonen 1990) were initially introduced with the purpose of producing a special mapping from a high dimensional input-space to a very low dimensional output-space while conserving the topological information of the input-space. It can also serve as vector quantizer as explained below. SOM is a neural network trained by following a non-supervised algorithm. As illustrated on figure 3.2, the network is made up of **two** layers of neurons. The input ("sensory") layer just distributes the inputs to the output layer. There is no data processing on it. The output layer has M^P neurons, arranged on a P dimensional lattice or *map*. Neurons on the output layer are characterised by

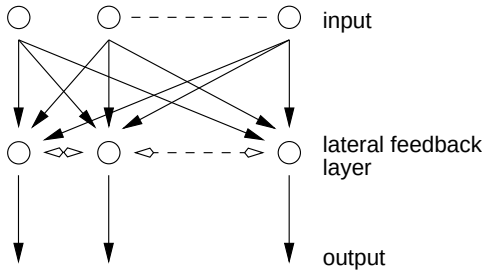


Figure 3.2: Kohonen self-organizing map. The network is made up of two layers of neurons : the **input layer** whose function is to distribute the inputs and the **output layer** with its lateral interaction which is active during training time.

- a map coordinate $\mathbf{k} = (k_1, k_2, \cdots, k_P)$ with $1 \leq k_1, k_2, \cdots, k_P \leq M$ that locates the neuron on the map.
- a synaptic weight vector $w_k = (w_k^1, w_k^2, \ldots, w_k^D)$ where D is the network's input dimension.

At time n, a D dimensional input vector $x_n = (x_n^1, x_n^2, \ldots, x_n^D)$ is presented to the network. The output response of each neuron k to input vector x_n is given by

$$g_k(n) = \|x_n - w_k\| \tag{3.7}$$

In our case, the norm $\|.\|$ is defined by Eq. 3.1 or 3.2. A winner neuron b is selected with

$$g_b(n) = \min_k \|\mathbf{x}_n - \mathbf{w}_k\| \tag{3.8}$$

During training of the network, input vectors are sampled randomly in the training set. For each training vectors, the learning proceeds in two phases :

1. The input vector x_n is presented to the network and the winner neuron $g_b(n)$ is determined with Eq. 3.8.
2. The weights are then updated according to

$$\mathbf{w}_k(n+1) = \mathbf{w}_k(n) + \eta(n)\Lambda(b-k, n)\,(\mathbf{x}_n - \mathbf{w}_k(n)) \tag{3.9}$$

   where $\eta(n)$ is the *adaptation gain* $(0 < \eta(n) < 1)$ generally decreasing with time and $\Lambda(b-k, n)$ is the *neighbourhood function.*

The neighbourhood function Λ is actually an *indirect* [3] lateral (feedback) interaction, defined on the discrete distance between the winner neuron and current neuron. Three types of neighbourhood functions are illustrated on figure 3.3. The neighbourhood function equals 1 for $k = b$ and is generally decreasing with time and distance to the winner neuron. The neighbourhood function can sometimes take negative values for neurons distant from the winner as for the Mexican hat function (fig. 3.3, c).

At each input presentation, the updating Eq. 3.9 attracts the winner neuron and its neighbours towards the input. The quantity by which the weights of neighbouring neurons are updated during learning is determined by the adaptation gain η **and** the neighbourhood function Λ.

Due to the fact that the learning algorithm "moves" the weights vectors towards the input vectors, the Kohonen network tries to map the distribution of the input vectors. In other words, the weight vectors will tend to form a discrete image of the input space, preserving its distribution.

[3] Neurons do not receive the inputs of their neighbours.

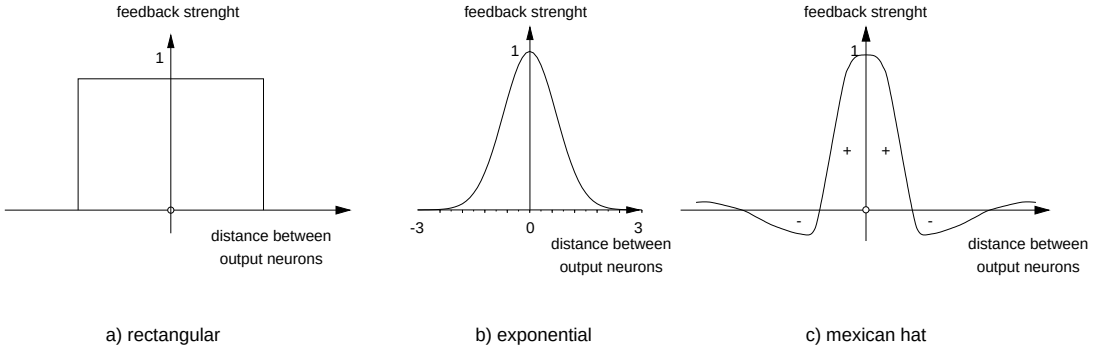


Figure 3.3: Different types of neighbourhood functions. a) Rectangular neighbourhood, b) exponential neighbourhood, c) Mexican hat neighbourhood. For the Mexican hat function, neurons close to the winner neuron on the map are driven towards the input vector while neurons far from the winner are repelled.

Synaptic weights can be used as centroids (code vectors) for vector quantization since they are of the same dimension as the input space. SOM training, from this point of view, can even be considered as a vector quantization learning. The approach is yet different to the k-means where the training criterion is clearly to decrease monotonously the average distortion. The distortion produced by a SOM quantizer after training is actually known to be larger than the distortion produced by a k-means algorithm.

As a by-product of the training, an extra piece of information is provided; namely the location of winners on the map. A topology information is kept in the coordinate space in the sense that nearby neurons are excited by nearby inputs.

3.1.4 Discrete HMMs

The vector quantization maps the acoustic vector x_n to a discrete value y_{i_n} selected in the codebook $\mathcal{Y} = \{y_1, \ldots, y_i, \ldots, y_I\}$. In the case of HMMs, the emission probabilities of equation 2.24 are directly stored in tables which are looked up when needed. Each state q_k is actually tied to a histogram of probabilities $p(y_i|q_k)$ with $1 \leq i \leq I$ as illustrated in Fig. 3.4.

HMM transition probabilities $p(q_l|q_k)$ are reestimated as in Eq. 2.33. Emission probabilities are reestimated with :

$$\overline{p}(y_i|q_k) = \frac{\text{expected number of time in state } k \text{ and observing centroid } y_i}{\text{expected number of time in state } k} \tag{3.10}$$

$$= \frac{\overline{P}(y_i, q_k)}{\overline{P}(q_k)} \tag{3.11}$$

$$= \frac{\sum_{\substack{n=1 \\ x_n \rightarrow y_i}}^{N} \gamma_n(k)}{\sum_{n=1}^{N} \gamma_n(k)} \tag{3.12}$$

where $x_n \rightarrow y_i$ indicates the input vector x_n that are quantized to centroid y_i. $\gamma_n(k)$ values are computed as in Eq. 2.30.

3.2 Experiments

All the experiments reported here are performed using the same system configurations in order to ensure reliable comparisons when changing parameters of the vector quantization algorithms :

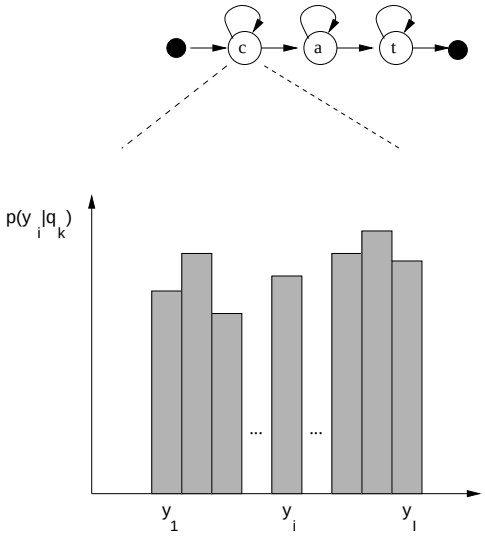


Figure 3.4: Discrete probability density functions for HMMs. A histogram of probabilities is tied to each state of the HMM.

Feature extraction : The 26 component feature vectors are composed of 12 LPC cepstrum coefficients, their first derivatives and the first and second derivatives of the log-energy. After pre-emphasis ($\alpha = 0.95$) and application of a Hamming window, ten LPC coefficients are used to compute the cepstrum. The acoustic vectors are computed every 10 ms over a 30-ms window. The algorithms are described in section 2.3.

Vector quantization : We systematically used four codebooks to quantize the acoustic vectors, considering 4 independent streams of data as in Eq. 2.25. In table of results, codebooks (*a*-*b*-*c*-*d*) means that we used 4 codebooks with a centroids for delta energy; b centroids for delta-delta energy; c centroids for cepstrum; d centroids for delta-cepstrum.

HMM : Context independent phoneme HMMs are used. Each phoneme is represented by a 3-state left-to-right model. In the case of both HIM and HER database, 42 phonemes are modelled. Strictly left-to-right topologies are used (no state skipping is permitted). Transition probabilities are fixed to 0.5 and are not reestimated during training.

Training and decoding: We used the Viterbi criterion to train (10 iterations) and decode the HMM.

3.2.1 HIM Database

We report here results obtained with the HIM database (see description in appendix A.3).

K-means

The error rates obtained with both LBG and k-means methods are presented in table 3.1. The first line gives recognition rates obtained using the classical LBG algorithm trained on 20 speakers. The codebook is split when the ratio between distortions of two successive iterations is less than 1% ($\varepsilon = 0.01$). The second line of the table is obtained by performing the k-means algorithm with binary splitting at each iteration. The training is also done on 20 speakers with $\varepsilon = 0.01$. K-means 2 and 3 are the same as k-means 1 but with respectively $\varepsilon = 0.001$ and $\varepsilon = 0.0001$.

Results of table 3.1 show that vector quantization parameters are of significant importance :

- If the codebook sizes are increased, recognition rates are improved. This is certainly due to the resolution of the probability histograms which is better with larger codebooks.

VQ method	32-32-64-64	32-32-128-128	32-32-256-256
LBG	26.9 %	23.3 %	22.1 %
k-means 1	-	20.7 %	18.6 %
k-means 2	-	19.6 %	17.5 %
k-means 3	-	-	17.8 %

Table 3.1: Error rates on the HIM corpus obtained using LBG and k-means vector quantization methods. Codebooks (*a-b-c-d*) means that we used 4 codebooks : a centroids for delta energy; b centroids for delta-delta energy; c centroids for cepstrum; d centroids for delta-cepstrum.

VQ method	32-32-121-121	32-32-256-256
Kohonen	18.7 %	15.9 %
Kohonen + k-means 1	18.4 %	15.4 %
Kohonen + k-means 2	18.5 %	15.8 %

Table 3.2: Error rates on the HIM corpus obtained using Kohonen and Kohonen+k-means vector quantization methods. Codebooks (*a-b-c-d*) means that we used 4 codebooks : a centroids for delta energy; b centroids for delta-delta energy; c centroids for cepstrum; d centroids for delta-cepstrum.

- Pure k-means algorithm outperforms the LBG algorithm. LBG is actually more sensitive to initial conditions and results generally to higher average distorsion than k-means. The better codebook quality reached by k-means leads to better recognition rates.
- Doing more k-means iterations improves the quality of the vector quantizer in terms of average distorsion while the recognition performances are not very much different.

Kohonen

Table 3.2 summarizes results obtained with the SOM vector quantizer. In the implementation, η is decreasing linearly with time and the neighbourhood function Λ is decreasing according to a negative exponential of time and distance to the winner neuron as shown in Fig. 3.2 c). The number of iterations is chosen proportionally to the number of neurons in the map. The dimension P of the map is equal to 1 for delta log-energy and delta-delta log-energy features. P is equal to 2 for cepstra and delta cepstra features. The number of neurons has been chosen in order to compare results with LBG and k-means [4].

The second and third lines of table 3.2 are obtained with a combination of both SOM and k-means algorithms. Centroids provided by Kohonen maps are used as initial values for the application of the k-means.

Results in table 3.2 are totally comparable with those of the k-means approach in table 3.1 since the same features extraction and HMM recognizer have been used. The following comments can be made :

- Average distortions produced by SOM quantizers are much larger than the distortion produced by k-means (for same codebook size). Despite its poor distortion performances, the Kohonen quantizer provides better recognition rates than in the k-means approach (table 3.1) for similar codebook sizes.
- In a similar manner as for k-means, recognition rates are improved when codebook sizes are increased.
- In terms of distortion and recognition rates, the combination of Kohonen and k-means procedures gives better results than the algorithms taken separately.
- When designing the codebook, SOM learning is much faster than k-means.

[4] Some maps have 121 neurons, corresponding to a square configuration with 11×11 neurons.

VQ method	32-32-128-128	32-32-256-256
LBG	11.0 (8.0) %	10.5 (7.3) %
k-means	9.6 (7.2) %	9.1 (6.2) %

Table 3.3: Error rates on the HER corpus obtained using LBG and k-means vector quantization methods. Codebooks (*a-b-c-d*) means that we used 4 codebooks : a centroids for delta energy; b centroids for delta-delta energy; c centroids for cepstrum; d centroids for delta-cepstrum.

VQ method	32-32-121-121	32-32-256-256
Kohonen	9.4 (6.6)	9.0 (6.2)
Kohonen + k-means	9.2 (6.6)	8.9 (6.1)

Table 3.4: Error rates on the HER corpus obtained using Kohonen and Kohonen+k-means vector quantization methods. Codebooks (*a-b-c-d*) means that we used 4 codebooks : a centroids for delta energy; b centroids for delta-delta energy; c centroids for cepstrum; d centroids for delta-cepstrum.

3.2.2 HER database

We re-did the same experiments on the HER database (see appendix A.2 for a description), using exactly the same conditions. Results with LPC-cepstrum coefficients are reported in table 3.3 for the LBG and k-means and in table 3.4 for Kohonen and Kohonen+k-means. Results with averaged coefficients (CMS) are given between parenthesis. The signal quality is better for the HER database than for the HIM database, which explains the difference of performances when comparing results from both databases.

The conclusions are exactly the same as for the HIM database.

3.3 Discussion and extensions of the system

Discrete HMMs don't require high CPU (though they require large memory) and are therefore well-suited for low-cost platform and real-time systems. We first discuss several ways to further reduce the computational load of the VQ step. Then, we refer to potential methods to improve performance of the system while preserving the speed advantage of discrete HMMs. Two methods for designing supervised VQ are presented. Finally, we refer to related studies in which particular features of SOM are advantageously exploited.

3.3.1 Fast nearest neighbour search

In real-time applications the nearest neighbour search time to solve Eq. 3.3 can be prohibitive. Reducing the search time can be achieved by different strategies :

- As explained in section 3.1.2, the LBG tree-structured quantization algorithm (Y. Linde 1980) can be used to guarantee a search time of $\log_2(I)$ through the codebook. Nevertheless, as shown in table 3.1 and 3.3, LBG degrades substantially the recognition performances.

- Several other techniques as the *k-d tree*, *triangle inequality*, ... have been developed to reduce search times without degradation of the quantization quality (Ramasubramanian and Paliwal 1988) (Cheng and Gersho 1986). The k-d tree builds a tree on an existing codebook and can be applied to either k-means or Kohonen codebooks. K-d tree is very efficient and provides search times compared to the balanced binary tree used in the LBG algorithm. Recognition experiments have been carried on with the k-d tree algorithm in (Himarnnet 1995) with k-means and Kohonen codebooks and no degradation of the recognition rates has been observed.

- Without going into refined search strategies as with k-d trees, implementation tricks of the nearest neighbour search of Eq. 3.3 can tremendously reduce the computation time when large codebooks

are used. One of the tricks is to stop the computation of the distance between the input vector and the centroid if the sub-total of Eq. 3.1 is already greater than the distance of the current winner. This shortcut works very well because cepstral values have generally (except c_0) (1) zero means and (2) variances essentially inversely proportional to the square of the coefficient index such that $Ec_n^2 \sim \frac{1}{n^2}$ [5]. The computation time has been measured to be five times lower for a codebook of 256 coefficients when this computational shortcut is used (Himarnnet 1995). A further reduction of the search time can be reached if the centroids are classified from the most probable to the less probable in the codebook.

3.3.2 Supervised VQ

Learning Vector Quantization

Learning Vector Quantization (LVQ) is a supervised technique to create codebooks for classification (Kohonen, Brna, and Chrisley 1988) (Kohonen 1990).

- During training, centroids are assigned to class labels and are moved iteratively in the input space in such a way that the amount of classification errors is reduced. The general principle of this iterative algorithm is to present a labelled training vector and to move the closest centroid according to its label. Several LVQ training procedures, referred as LVQ1, LVQ2 and LVQ3 are proposed in (Kohonen 1990). They have different convergence behaviour and we refer to (Kohonen 1990) for further discussion. LVQ3 seems to be the more robust training scheme amongst them.

- During classification, the nearest neighbour search performs the classification, assigning the input vector to the class label of the closest centroid. More than one centroid can share the same class-label and therefore, the region of the input space associated to a class is formed of a set of Voronoi regions.

LVQ does not position centroids in the input space so that they mimic the distribution. Instead, it aims at minimizing the number of mis-classifications. This classification method is then very attractive because class boundaries can be very complex and classification time is low since performed through a VQ. LVQ has been extensively used to build the so-called *Finnish phonetic typewriter system* (Torkkola et al. 1991). Other studies were also reporting good results for speech recognition systems using LVQ techniques (Mantysalo, Torkkola, and Kohonen 1992),(Kangas, Torkkola, and Kokkonen 1992).

With LVQ, the feature vectors are classified into phone classes, resulting in a (noisy) sequence of phone hypothesis. This sequence is then decoded by a discrete HMMs whose role can be described as to correct the noisy phone sequence. Another point of view is to consider the LVQ as a way to design codebooks with the criterion to minimize the number of frame mis-classification. An interesting question in the framework of the VQ comparison presented in this chapter is how do these LVQ codebooks perform when used in place of a k-means or SOM codebook in discrete HMM speech recognition systems.

We report here several results obtained with LVQ3 on the HER database. The database has been Viterbi aligned on HMMs built from 42 CIP models with 3 states by CIP. Each frame is labelled as belonging to one of the CIP. The procedure to build the codebooks with LVQ3 is as follows :

1. An initial codebook is obtained from the average values of the features in each class. Other initialization procedures can be used but this one leads to good results. We found out that the initial configuration of the codebook is a very sensitive matter for good LVQ-training in the case of strongly overlapping classes.

2. The iterative LVQ3 algorithm is applied and centroid positions are fine-tuned to minimize the classification errors.

Codebooks are then used as usual codebooks except that the label which is output of the VQ is the class label tied to the centroid (and not the index of the centroid as before).

[5] While this statement is not anymore valid if liftering is applied, the computational shortcut still reduce consequently the search time for large codebooks.

Error rate	k-means 128	class mean 126(126)	LVQ3 126(126)	LVQ3 378(126)
Frame Level	-	87.2 %	83.0 %	83.3 %
Word Level	16.7 %	16.0 %	16.1 %	15.8 %

Table 3.5: Error rates on the HER corpus obtained using k-means and LVQ vector quantization methods. Results are obtained using 12 CMS as feature vectors. The notation 126(126) means that 126 centroids are used in the codebook and that (126) labels (42 CIP times 3 states) are used. For the 378(126) configuration, 3 centroids are tied to each state, leading to a codebook size of 378.

Error rate	k-means 32-32-128-128	class mean 32-32-378(126)-378(126)	LVQ3 32-32-378(126)-378(126)
Word Level	7.2 %	7.3 %	7.3 %

Table 3.6: Error rates on the HER corpus obtained using k-means and LVQ vector quantization methods. Results are obtained using 12 CMS as feature vectors. For configurations 32-32-378(126)-378(126), no supervision for ΔE and $\Delta\Delta E$ codebook is used, while CMS and ΔCMS are supervised (3 centroids per class).

Many experiments with LVQ were performed and we refer to the report (Himarnnet 1995) for more details. We will summarize here the main conclusions and illustrate them with some typical results. Table 3.5 and 3.6 reports error rates on the HER database. The same configuration as in section 3.2.2 is used excepted for the feature extraction which is here done with cepstral mean substraction (CMS). The first set of results is obtained using only 12 CMS features and one codebook. The second set of results is obtained with the usual 4 codebook configuration. Conclusions are the following :

- Supervised VQ decreases the number of mis-classifications at the frame level while no or few improvements are seen at the word level. In other words, better frame recognition does not imply better word recognition in the case of discrete HMMs.
- Refinement of the class average values with LVQ3 does not increase performances in terms of word error rates.
- Using three centroids per class improves somehow results when one codebook of 12 CMS is used but no improvements is seen with four codebooks configurations.

MLP used as labellers

MLP can be used to produce labels for discrete HMMs. The approach has been proposed in (Cerf, Ma, and Compernolle 1994) and experiments have been carried out on the HER database in the framework of the Himarnnet project (Himarnnet 1995).

The MLP is supervisely trained to discriminate phones (as explained in chapter 4). A forced Viterbi alignment on the training set is performed to obtain target labels used during training. At recognition time, the MLP outputs are used to label the vector sequence. For each frame, the index of the largest output defines the class label. MLP can be seen here as non-linear vector quantizers.

Although conceptually stronger than unsupervised VQ training, it is not clear why the continuous outputs of the MLP are not used directly by the HMMs to compute word scores[6]. In (Cerf, Ma, and Compernolle 1994), it is argued that for VQ, smaller MLP can be used since the labelling task is somehow easier than when continuous posterior probabilities are required. Nevertheless, the comparison between both continuous and discrete approaches does not show a clear advantage for the discrete one.

Furthermore, MLP VQ does not seem to outperform Euclidean SOM VQ. A MLP trained with 9 frames of context with 230 hidden units and 126 outputs (3 HMM states per phone) as been reported to

[6]This approach is actually studied in chapter 4 of this thesis report.

give 6.5% error rates on the HER database. SOM systems with 4 codebooks (32-32-256-256) performs around 6.2% (see table 3.4). Extensive comparisons can be found in (Himarnnet 1995).

3.3.3 Topological information of SOM

SOM, after training, has the property to excite nearby neurons with nearby inputs. This extra piece of information has not been exploited here but some studies have tried various attempts in this direction.

- In (Zhao and Rowden 1992), the topological information is usefully taken into account by HMM recognisers in the case of small training set. Neurons in the neighbourhood of the winner are used to smooth emission probabilities of the discrete HMM. This technique compares fairly well against other more classical smoothing techniques.
- In (Kangas, Torkkola, and Kokkonen 1992) and (Huang and Kuh 1992), SOM activation is accumulated through the whole feature sequence before being fed into a MLP trained to discriminate isolated words. Feature trajectories are projected on the map and the topological information helps to smooth trajectories. While attractive because globally discriminant, this combination of SOM and MLP can only be used for isolated word recognition. Furthermore, the MLP part needs to be re-trained if a new word is added in the list.
- Torkkola () presented a method to use the quantization error as an extra piece of information for the acoustic modeling in the HMM.

3.4 Conclusion

In this chapter, we have described an original and successful utilisation of Self-Organizing Map (SOM) of Kohonen and discrete HMM for speech recognition systems. Although conceptually weaker than continuous supervised systems like hybrid systems (discussed in chapter 4), the unsupervised approach presented here offers performance and learning speedup advantages when compared to other wel-spread discrete systems.

We compared SOM with two baseline algorithms, namely LBG and k-means. A theoretical analysis of the algorithms gave us a better understanding of their differences. We also performed recognition experiments using the same system on two telephone speech isolated word databases. The following conclusions can be drawn for the two approaches :

1. **Minimum distorsion criterion**. Generalized Lloyd algorithms, better known as k-means, are classically used to design codebooks. The method is iterative, providing an explicit minimization of the average distorsion. A tree-based version of the k-means, referred to as the LBG algorithm, is providing a fast way to perform the VQ during recognition, while degrading the quality of the VQ (larger distorsion). The recognition results obtained using discrete HMMs show that the k-means procedure combined with a fast search algorithm should be preferred to the LBG algorithm.
2. **Self-organization criterion**. As an alternative to k-means and LBG, SOM of Kohonen are proposed to design codebooks. A SOM is a neural network trained by following a non-supervised algorithm in which weight values can be used to form a codebook after training. Weight vectors tend to form a discrete image of the input space, preserving its distribution. The distorsion is decreasing during training, but is not explicitly minimized. Despite its poor distortion performance, the Kohonen algorithm produces codebooks which lead to consistently better recognition rates than when using k-means' family algorithms. Further improvements are also obtained when SOM codebooks are used as a starting point for k-means refinement.

We also went further in the ANN direction, trying to build codebooks in a supervised way using the Learning Vector Quantization (LVQ) algorithm. Training targets are obtained with a forced Viterbi alignment. After training the codebook with LVQ, vector quantization is performed as usual before feeding discrete HMMs. Frame-based classification is better but no improvement is reported at the

word level. This is somehow normal since the discrete HMMs need a good resolution of the input space distribution, which is, in a similar manner as for k-means, not guaranteed by positioning centroids so that frame classification errors are minimized.

Bibliography

Cerf, P. L., W. Ma, and D. V. Compernolle (1994, January). Multilayer perceptrons as labelers for hidden markov models. *IEEE Transactions Acoust., Speech, Signal Processing 2*(1), 185–193. Part II.

Cheng, D.-Y. and A. Gersho (1986). A fast codebook search algorithm for nearest-neighbor pattern matching. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 265–268.

Fontaine, V., J. Hennebert, and H. Leich (1994). Influence of vector quantization on isolated word recognition. In *Proceedings of Eusipco*, Edinburgh, pp. 115–118.

Gersho, A. and R. Gray (1992). *Vector Quantization and Signal Compression*. Kluwer Academic Publishers.

Himarnnet (1995, September). Final report of the himarnnet esprit project.

Huang, Z. and A. Kuh (1992, November). A combined self-organizing feature map and multilayer perceptron for isolated word recognition. *IEEE Transactions on Signal Processing 40*(11), 2651–2657.

Kangas, J., K. Torkkola, and M. Kokkonen (1992). Using soms as feature extractors for speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 341–344.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics 43*, 59–69.

Kohonen, T. (1990, September). The self-organizing map. *Proceedings of the IEEE 78*(9), 1464–1477.

Kohonen, T., G. Brna, and R. Chrisley (1988). Statistical pattern recognition with neural networks: Benchmarking studies. In *Proceedings of the second IEEE International Conference on Neural Networks*, Volume 1, San Diego, pp. 61–68.

Mantysalo, J., K. Torkkola, and T. Kohonen (1992). Lvq-based speech recognition with high-dimensional context vectors. In *Proceedings ICSLP*, Banff, Alberta, Canada, pp. 539–542.

Rabiner, L. and B.-H. Juang (1993). *Fundamentals of Speech Recognition*. Prentice Hall.

Ramasubramanian, V. and K. Paliwal (1988). An optimized k-d tree for fast vector quantization of speech. In *Eusipco*, pp. 875–878.

Torkkola, K. et al. (1991, June). Status report of the finnish phonetic typewriter project. In *Proceedings of the Internaltional conference on Artificial Neural Networks*, Espco, Finland, pp. 771–776.

Y. Linde, A. Buzo, A. G. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communication 28*(1), 84–95.

Zhao, Z. and C. Rowden (1992, December). Use of kohonen self-organiaing feature maps for hmm parameter smoothing in speech recognition. *IEE Proceedings-F 139*(6), 385–390.

Chapter 4

Feedforward ANNs and Continuous HMMs

Although it is a simple matter to write a three- or five-coefficient system of equations, it is often a distinct challenge to write one that is physically relevant.

Hampton N. Shirer

We describe in this chapter speech recognition systems based on continuous HMMs and supervised ANNs. The combination of continuous HMMs and ANNs is called in the literature **hybrid HMM-ANN systems**.

Advantages of hybrid systems have already been underlined in various publications (Bourlard and Morgan 1993; Bourlard and Morgan 1994; Renals, Morgan, Bourlard, M.Cohen, and Franco 1994). In its original form, hybrid systems use multilayer perceptrons (MLPs) to classify input frames into phone categories. At recognition time, MLP outputs are estimates of the frame posterior probabilities which are easily transformed into frame likelihood for processing with classical decoding. At training time, the system uses the Viterbi criterion to associate frames to phone categories. Frame labels are then used to train the ANN in a supervised way. We will briefly recall the functioning and advantages of Viterbi trained hybrid systems in the first section.

In the second section, we present a new theoretical formalism for hybrid systems. It is shown that if the ANN part estimates local posteriors, the hybrid system can be used to model global model posteriors. This formalization provides us with a clear theory unifying the "classical" Viterbi trained hybrid systems and the more recent developments of hybrid systems, namely REMAP (Bourlard, Konig, and Morgan 1995). Starting from this formulation, a new forward-backward training of hybrid HMM/ANN systems is derived. It is also discussed to which extent the formalism unifies a class of recurrent ANNs systems used to estimate local posterior (Robinson and Fallside 1991; Robinson, Hochberg, and Renals 1996). Similarly to what has been done in section 2.4, the set of hypothesis that makes the system tractable are underlined. Theoretical advantages of hybrid systems against Gaussian mixture systems will be put in evidence showing that some hypothesis can be better relaxed.

In the third section, comparisons of performance between Viterbi and forward-backward hybrid sys-

tems are presented and discussed. Graphical representations shows what are the differences of both approaches. Further performance comparisons with classical Gaussian mixtures systems are given.

In the last section, discussions and extensions of the different systems are presented.

4.1 Original Viterbi trained HMM/ANN system

4.1.1 Motivations

HMMs inherently incorporate the sequential and statistical character of the speech signal. Standard HMMs have proved their efficiency in speech recognition, however, they still suffer from several weaknesses, namely:

- a priori choice of a model topology, e.g. a number of states is imposed for each subword model;
- a priori choice of statistical distributions for the emission probabilities $p(x_n|q_k)$ associated with each state: multivariate Gaussian density, mixture of multivariate Gaussian densities,...;
- first order Markov assumption, i.e., the probability of being in a given state at time n only depends on the state at time $n-1$;
- poor discrimination due to the training algorithm which maximises likelihoods instead of a posteriori probabilities.

In section 1.4, we reported about some successful applications of ANNs in pattern recognition such as vision or character recognition. The major strength of ANNs in pattern classification is in the fact that there is no need for any particular assumptions about statistical distributions and independence of input features, and also that ANNs can be trained in such a way that the network exhibits discriminant properties. For speech recognition, the major weakness of ANNs is their inability to deal easily with the time sequential nature of speech.

Recent research tried to take advantage of HMMs, i.e. incorporate the time variability and sequential nature of the speech signal, and of MLPs, i.e. classify without making any particular assumptions about the statistical distribution of speech features. This combination leads to the so-called hybrid system HMM/MLP in which MLPs are used to compute the emission probabilities associated with each HMM state. A large amount of work and studies have already been dedicated to hybrid systems (Bourlard and Morgan 1994; Renals, Morgan, Bourlard, M.Cohen, and Franco 1994; Robinson, Hochberg, and Renals 1996). After several years of optimization, it seems that this mixed approach outperforms both pure HMMs methods and pure ANNs methods in terms of recognition performances. Better results for the hybrid approach are almost systematically reported in the literature when classic and hybrid systems using similar number of parameters (*controlled* experiments) are compared.

4.1.2 Viterbi decoding and training

In hybrid systems, each MLP output is associated with a particular state q_k of the set of states $Q = \{q_0, q_1, \cdots, q_K\}$. The MLP is trained to classify its inputs into one of Q classes. Provided some training conditions (see section 2.5.3), the MLP is able to estimate a posteriori probabilities $p(q_k|x_n)$ when x_n, a particular acoustic vector, is presented as input.

As explained in Eq. 2.27 (or Eq. 2.35 for Viterbi), the computation of HMM likelihoods $P(M|X)$ requires the computation of local likelihoods $p(x_n|q_k)$. The network provides here a posteriori probabilities $p(q_k|x_n)$ and a conversion into local likelihood is therefore necessary. Local posteriors are usually converted into *scaled likelihoods* using Bayes' rule :

$$\frac{p(x_n|q_k)}{p(x_n)} = \frac{p(q_k|x_n)}{p(q_k)} \tag{4.1}$$

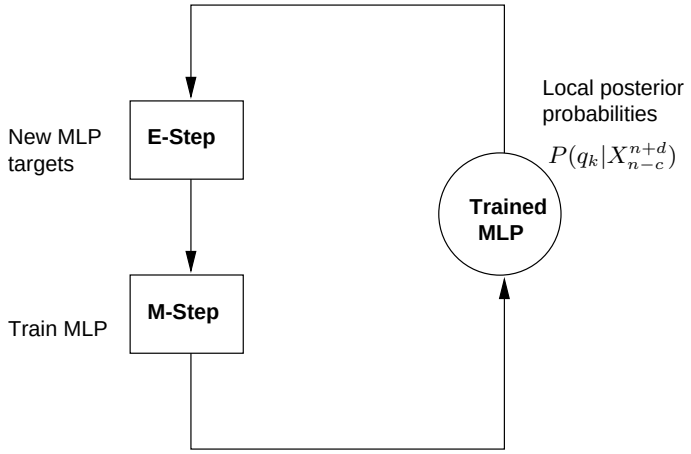


Figure 4.1: Expectation maximisation training of hybrid HMM/ANN system.

Scaled likelihoods $\frac{p(x_n|q_k)}{p(x_n)}$ can be used instead of likelihoods $p(x_n|q_k)$ for the model score computation. Indeed, the introduction of the factor $p(x)$ can be disregarded since it is not dependent to the model (it will influence in the same manner scores of all the competing models). Nevertheless, it is not clear what is actually estimated when scaled likelihoods are used in place of likelihoods. This will be clarified in the next section where the new formulation for hybrid systems will be presented.

The prior probability $p(q_k)$ can easily be computed, counting the number of times each feature vector x_n is associated with state q_k while doing Viterbi forced alignment.

As illustrated in figure 4.1 and 4.2, the training procedure of the hybrid system is quite similar to the one of standard Viterbi trained HMMs. Iterations of expectation and maximisation steps (EM procedure) guarantees the maximisation of the likelihood :

- During the expectation phase, the best state sequence are computed with a Viterbi alignment algorithm (see section 2.4.4) while the parameter set is fixed (HMM transition probabilities and MLP weights). Each frame from the feature sequence is labelled in terms of phone categories. Examples of Viterbi alignment on real data are given further in the text in Fig 2.9.

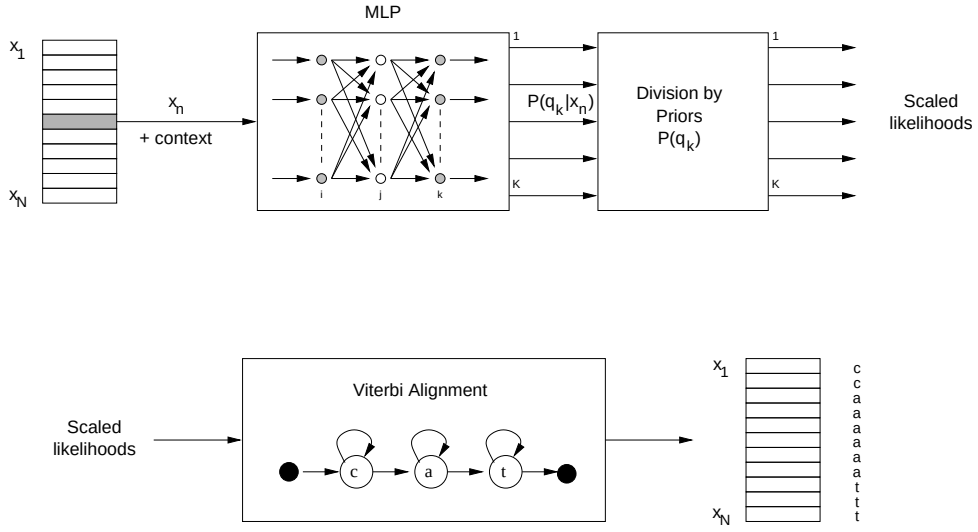
- The frame labelling obtained in the E-phase allows the MLP to be trained with the *1-from-K* training procedure (see section 2.5.3). Standard stochastic backpropagation is generally used to minimize a MSE or entropy error measure. New values for transition probabilities can also be computed from the alignment.

The EM procedure is repeated iteratively until convergence.

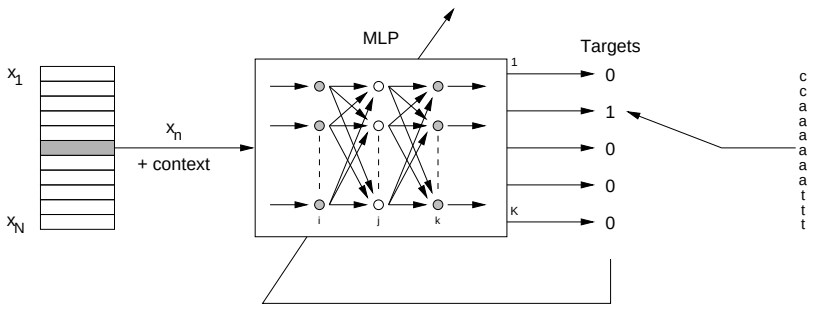
4.1.3 System advantages

Estimating probabilities with MLP enables to make weaker assumptions than standard continuous HMM :

- No assumptions need to be done on the input vector statistics. For example, Gaussian mixture systems often make the assumption of component uncorrelation in order to simplify the local likelihood estimation (diagonal covariance matrix). With MLP, intra-vector correlations will be put in evidence through training. Often, multiple source of evidence as Δ or $\Delta\Delta$ features are combined in order to increase the contextual information. Nevertheless, the dimension of the input vector is limited because the estimation of multivariate likelihood is difficult (large amount of data is required and precision problems are encountered). Because of the discriminant training capabilities, MLP can take as input several adjacent frames without going through these problems.



a) Expectation phase



b) Maximisation phase

Figure 4.2: Training of hybrid HMM/ANN system. a) Expectation phase : MLP parameters are fixed and best state paths are computed with the Viterbi criterion. State labels are associated to acoustic vectors. b) Maximisation phase : MLP parameters are trained with the newly gotten targets.

- No assumption is taken about the functional form of the probability density function. Gaussian HMMs make the assumption that features are distributed following a Gaussian distribution, which is not necessarily true. Increasing the number of mixtures in Gaussian HMM permits to model somehow more complicated distributions, relaxing the Gaussian distribution hypothesis, but at the cost of increasing the computational load.

- MLPs, with their repetitive structures, are well suited to speed up software and hardware implementations (parallelism, VLSI design, ...).

It has also been suggested that MLPs, due to their shared parameters (fully interconnected layers), make a better usage of the available memory resources.

4.1.4 System drawbacks

Using MLPs to estimate posterior probabilities is not so straightforward as it looks. Problems are mainly encountered while training the network :

- Stochastic backpropagation is slow in nature since the training set has to be visited several times (epochs) to reach good performances. Often, dedicated hardware is developed and special care is needed to optimize the input/output data flow to and from the network in order to speed up training.

- Overtraining is often encountered especially if the amount of parameters is large. Special care has to be taken when reducing the learning rate η. Usually, as introduced in (Bourlard and Morgan 1994), the averaged error is measured on an independent *cross-validation* data set and the learning rate is decreased when the error measure on this set is increasing from an epoch to the next. This procedure is generally enough to reduce the risk of overtraining.

- Problems of local minima. Since the number of parameters is usually large and since the classes are strongly overlapping, the network can get stuck in local minima. The only way to reduce risks of local minima is to train more than one network and to discard the one having bad performances. Another point to bare in mind is to carefully sample input patterns randomly in the training set. Indeed, if the sampling is sequential, risks of falling in a local minimum are larger because adjacent frames often belong to the same category and training would be biased.

- The optimal network architecture is found by errors and trials. Nevertheless, using cross-validation and random sampling almost guarantee, in practice, that the network will be correctly trained without overtraining. Therefore and roughly, the more parameters there will be in the network, the better the performances will be.

Another drawback is maybe the "un-scalability" of the system. If, for some reason, the number of phone needs to be reduced or increased, a new MLP has to be completely re-trained. Gaussian mixture can be more easily rescaled, training only the units that are modified. A criticism of MLP lies also in the fact that no interpretation of the weights can be given, making somewhat professional developers uncomfortable with hybrid systems.

All these drawbacks maybe explain the fact that no (or very few) speech companies are actually offering hybrid systems.

4.2 New formulation

In (Bourlard, Konig, and Morgan 1995; Bourlard, Konig, and Morgan 1996) it was shown that it is possible to express the global posterior probability $P(M|X,\Theta)$ of a model M given the acoustic data X and the parameters Θ in terms of the local posteriors $P(q_l^n|q_k^{n-1},x_n,\Theta)$, conditional transition probabilities where q_k^n denotes the specific state q_k of M at time n. In a similar manner as for classic hybrid HMM/MLP systems, these local posteriors can be estimated with an ANN in which the input vector is extended with

the previous state information. A recursive training procedure, known as REMAP[1] was introduced to iteratively estimate the parameter set Θ. This procedure is actually an application of the usual generalized EM algorithm : the global posterior probability of the correct model can be optimized by optimizing the local posterior probabilities through re-estimating targets for the ANN.

The REMAP algorithm and its corresponding theory was used as starting point to re-formulate the theory of the original HMM/ANN system. Looking deeper at REMAP will demonstrate that the original HMM/ANN system (Renals, Morgan, Bourlard, M.Cohen, and Franco 1994) (Robinson, Hochberg, and Renals 1996) trained using local criteria indeed optimizes the global posterior probability, given certain well-defined assumptions. This new formulation of the theory is more general and can explain REMAP and the classical Viterbi training of hybrid systems. Furthermore, in the light of the theory, a forward-backward training algorithm for the original HMM/ANN system has been discovered.

4.2.1 Estimation of Global Posteriors

The objective of the REMAP formulation is to produce an estimate of the global posterior probability of a model M given the acoustic data $X = X_1^N = \{x_1, x_2, \ldots, x_N\}$ and the parameter set Θ :

$$P(M|X) = \sum_{\ell_1=1}^{L} \ldots \sum_{\ell_N=1}^{L} P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N, M|X) \tag{4.2}$$

where $q_{\ell_n}^n$ is HMM state ℓ_n visited at time n, and the summation is over all possible state sequences. The Viterbi approximation maximizes Eq. 4.2 over the best state sequences :

$$\overline{P}(M|X) = \max_{\ell_1,\ldots,\ell_N} P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N, M|X) \tag{4.3}$$

If we consider a particular state sequence, the posterior probability of the state sequence and the model may be decomposed into the product of an acoustic model and a prior over models ("language model" and state sequences) :

$$\begin{aligned} P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N, M|X) &= P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N|X) P(M|X, q_{\ell_1}^1, \ldots, q_{\ell_N}^N) \\ &\simeq \underbrace{P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N|X)}_{\text{ac.model}} \underbrace{P(M|q_{\ell_1}^1, \ldots, q_{\ell_N}^N)}_{\text{prior}} \end{aligned} \tag{4.4}$$

The X dependence in the second factor in Eq. 4.4 is dropped since the hidden part (the state sequence) is hypothesized.

The term $P(M|q_{\ell_1}^1, \ldots, q_{\ell_N}^N)$ is an *indicator* function that is equal to 0 if the state sequence is forbidden by the model topology and to 1 if the state sequence is permitted[2]. Forbidden state sequences are determined by transition between states that have a zero probability. For sake of clarity, we will drop this indicator term for the rest of the development, assuming that the summation (maximisation) in Eq. 4.2 and 4.3 involves only the permitted state sequences.

With the usual assumptions of a first-order Markov process and conditionals on X limited to local context X_{n-c}^{n+c} we can simplify the acoustic factor in Eq. 4.4 with the law of compound probabilities :

[1] For "recursive estimation and maximisation of a posteriori probabilities".

[2] The only case where this function could differ from 0 or 1 is when multiple models have the same state sequence (homophones).

$$\begin{aligned} P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N|X) &= P(q_{\ell_1}^1|X)P(q_{\ell_2}^2|X, q_{\ell_1}^1)\ldots \\ &\quad \ldots P(q_{\ell_N}^N|X, q_{\ell_1}^1, \ldots, q_{\ell_{N-1}}^{N-1}) \\ &= \prod_{n=1}^{N} P(q_{\ell_n}^n|X, Q_1^{n-1}) \\ &\simeq \prod_{n=1}^{N} P(q_{\ell_n}^n|X_{n-c}^{n+d}, q_{\ell_{n-1}}^{n-1}) \end{aligned} \tag{4.5}$$

We can re-write Eq. 4.2:

$$P(M|X) \simeq P(M) \sum_{\ell_1,\ldots,\ell_N} \left[\prod_{n=1}^{N} P(q_{\ell_n}^n|X_{n-c}^{n+d}, q_{\ell_{n-1}}^{n-1}) \right] \tag{4.6}$$

The Viterbi approximation may be obtained by replacing the sum over state sequences $(\ell_1, \ldots, \ell_N)$ with a maximization over state sequences:

$$\overline{P}(M|X) \simeq P(M) \max_{\ell_1,\ldots,\ell_N} \left[\prod_{n=1}^{N} P(q_{\ell_n}^n|X_{n-c}^{n+d}, q_{\ell_{n-1}}^{n-1}) \right] \tag{4.7}$$

Original HMM/ANN system

If we consider a particular term of Eq. 4.6, we can write:

$$\begin{aligned} P(q_{\ell_n}^n|X_{n-c}^{n+d}, q_{\ell_{n-1}}^{n-1}) &= \frac{P(q_{\ell_n}^n, q_{\ell_{n-1}}^{n-1}, X_{n-c}^{n+d})}{P(X_{n-c}^{n+d}, q_{\ell_{n-1}}^{n-1})} \\ &= \frac{P(X_{n-c}^{n+d}|q_{\ell_n}^n, q_{\ell_{n-1}}^{n-1})P(q_{\ell_n}^n, q_{\ell_{n-1}}^{n-1})}{P(X_{n-c}^{n+d}, q_{\ell_{n-1}}^{n-1})} \\ &= \frac{P(X_{n-c}^{n+d}|q_{\ell_n}^n, q_{\ell_{n-1}}^{n-1})P(q_{\ell_n}^n|q_{\ell_{n-1}}^{n-1})P(q_{\ell_{n-1}}^{n-1})}{P(X_{n-c}^{n+d}, q_{\ell_{n-1}}^{n-1})} \end{aligned} \tag{4.8}$$

Assuming the independence of the acoutic frame in time n on the previous state in time $n-1$, we can write:

$$\begin{aligned} P(q_{\ell_n}^n|X_{n-c}^{n+d}, q_{\ell_{n-1}}^{n-1}) &= \frac{P(X_{n-c}^{n+d}|q_{\ell_n}^n)P(q_{\ell_n}^n|q_{\ell_{n-1}}^{n-1})P(q_{\ell_{n-1}}^{n-1})}{P(X_{n-c}^{n+d})P(q_{\ell_{n-1}}^{n-1})} \\ &= \frac{P(X_{n-c}^{n+d}|q_{\ell_n}^n)P(q_{\ell_n}^n|q_{\ell_{n-1}}^{n-1})}{P(X_{n-c}^{n+d})} \end{aligned} \tag{4.9}$$

Recalling Bayes rule:

$$\frac{P(X_{n-c}^{n+d}|q_{\ell_n}^n)}{P(X_{n-c}^{n+d})} = \frac{P(q_{\ell_n}^n|X_{n-c}^{n+d})}{P(q_{\ell_n}^n)} \tag{4.10}$$

we can write:

$$P(q_{\ell_n}^n|X_{n-c}^{n+d}, q_{\ell_{n-1}}^{n-1}) = \frac{P(q_{\ell_n}^n|X_{n-c}^{n+d})P(q_{\ell_n}^n|q_{\ell_{n-1}}^{n-1})}{P(q_{\ell_n}^n)} \tag{4.11}$$

Using the Viterbi criterion, we can rewrite Eq. 4.7 as :

$$\overline{P}(M|X) \simeq \max_{\ell_1,\dots,\ell_p N} \left[\prod_{n=1}^{N} P(q_{\ell_n}^n|X_{n-c}^{n+d}) \frac{P(q_{\ell_n}^n|q_{\ell_{n-1}}^{n-1})}{P(q_{\ell_n}^n)}\right] P(M) \tag{4.12}$$

where local posteriors $P(q_{\ell_n}^n|X_{n-c}^{n+d})$ can be estimated with a MLP. We arrive at a hybrid system similar to those previously developed, (e.g., in (Renals, Morgan, Bourlard, M.Cohen, and Franco 1994)). Eq. 4.12 gives also a clear justification for dividing the local posterior estimate by the training data priors as it was suggested in the original formulation of hybrid systems. $P(q_{\ell_n}^n|X_{n-c}^{n+d})/P(q_{\ell_n}^n)$ are actually the scaled likelihoods of Eq. 4.1 that are used in the decoding.

Eq. 4.12 actually demonstrates that given the previously stated assumptions, the original hybrid HMM/ANN system, as described in section 4.1 do produce an estimate of the global posterior $P(M|X)$.

Equations 4.6 and 4.12 also provide us with a clear way of properly including language model information $[P(M)]$ into the formalism.

Forward-Backward Estimation

We can now derive a forward-backward algorithm for hybrid HMM/ANN training without making use of the Viterbi approximation. Indeed, if we remove the dependency on the previous state in Eq. 4.6, we obtain :

$$P(M|X) \simeq \sum_{\ell_1,\dots,\ell_N} \left[\prod_{n=1}^{N} P(q_{\ell_n}^n|X_{n-c}^{n+d}) \frac{P(q_{\ell_n}^n|q_{\ell_{n-1}}^{n-1})}{P(q_{\ell_n}^n)}\right] P(M) \tag{4.13}$$

This formulation is actually a direct application of the Generalized EM algorithm, where the missing data is the state sequence (as usual in HMM estimation). The E-step is the estimation of ANN targets using a forward-backward recurrence and the M-step is the MLP training. This is a *generalized* EM algorithm since the M-step is not exact.

We can write down forward (α) and backward (β) recurrences:

$$\begin{aligned} \alpha_n(\ell) &= \frac{p(X_1^n, q_\ell^n|M)}{p(X_1^n)} \qquad (4.14)\\ &= \left[\sum_k \alpha_{n-1}(k) p(q_\ell|q_k)\right] \frac{p(q_\ell|x_n)}{p(q_\ell)} \\ \beta_n(\ell) &= \frac{p(X_{n+1}^N|q_\ell^n, X_1^n, M)}{p(X_{n+1}^N)} \qquad (4.15)\\ &= \sum_k \beta_{n+1}(k) P(q_k|q_\ell) \frac{p(q_k|x_{n+1})}{p(q_k)} \end{aligned}$$

which are similar to the standard recurrences used in HMMs, apart from the scaling factor $p(x_n)$. This scaling factor is necessary:

1. To have the α (or β) recursion estimating $\frac{p(X|M)}{p(X)} = \frac{P(M|X)}{P(M)}$.

	$P(M\|X,\Theta)$
REMAP	$\simeq \sum_{\ell_1,\dots,\ell_N} \left[\prod_{n=1}^{N} P(q_{\ell_n}^n \| X_{n-c}^{n+d}, q_{\ell_{n-1}}^{n-1})\right] P(M)$
Forward-Backward	$\simeq \sum_{\ell_1,\dots,\ell_N} \left[\prod_{n=1}^{N} P(q_{\ell_n}^n \| X_{n-c}^{n+d}) \frac{P(q_{\ell_n}^n \| q_{\ell_{n-1}}^{n-1})}{P(q_{\ell_n}^n)}\right] P(M)$
Standard Hybrid	$\simeq \max_{\ell_1,\dots,\ell_N} \left[\prod_{n=1}^{N} P(q_{\ell_n}^n \| X_{n-c}^{n+d}) \frac{P(q_{\ell_n}^n \| q_{\ell_{n-1}}^{n-1})}{P(q_{\ell_n}^n)}\right] P(M)$

Table 4.1: Summary of the different algorithms unified with the formalization.

2. To have the α and β recurrences expressed in terms of $\frac{p(x_n|q_k)}{p(x_n)} = \frac{P(q_k|x_n)}{P(q_k)}$, which is the only value that can be estimated by the ANN (provided that we can get an estimate of $P(q_k)$ – see below).

Assuming that we can estimate state priors in the full forward-backward framework, the ANN targets may then be re-estimated using the following:

$$\begin{aligned} \gamma_n(k) &= P(q_k^n|X,M) \\ &= \frac{p(q_k^n, X|M)}{p(X|M)} \\ &= \frac{\alpha_n(k)\beta_n(k)}{\sum_\ell \alpha_n(\ell)\beta_n(\ell)} \end{aligned} \tag{4.16}$$

As for standard forward-backward recursion, convergence can be proved. Forward-backward MLP training has been previously employed for speech recognition in (Yan, Fanty, and Cole 1997) and for handwriting recognition in (Senior and Robinson 1996). However in this previous work it was assumed an explicit Viterbi segmentation when estimating the priors $P(q_k)$ required by Eq. 4.13. As a generalization of what has been done with Viterbi-based hybrid HMM/ANN systems, priors $P(q_k)$ can be estimated in the framework of the forward-backward procedure as :

$$P(q_k) = \frac{\sum_{n=1}^{N} P(q_k^n|X,M)}{N} = \frac{\sum_{n=1}^{N} \gamma_n(k)}{N} \tag{4.17}$$

which allows us to compute forward and backward recurrences and to iterate the training process.

REMAP

The REMAP (Bourlard, Konig, and Morgan 1995) training algorithm uses the formulation as it is in Eq. 4.6. Local conditional transition probabilities $P(q_k^n|X, q_l^{n-1})$ are estimated by a particular form of MLP in which the input is extended with a binary set of values encoding the state label q_l^{n-1} of permitted previous states. Similarly to what is done with forward-backward, a generalized EM procedure is used to maximize during training the global posterior probability of the word sequences. α and β probabilities can be defined and recursion can be formulated for training and decoding.

Table 4.1 summarizes the different algorithms unified with the formalization.

4.2.2 Discussion and system extension

Soft duration modeling

Similarly to what is done with Viterbi (see section 2.4.5), we can also define the state duration in the case of forward-backward recursion :

$$d_{u_i}(q_k) = \sum_{n=1}^{N} \gamma_n(k) \tag{4.18}$$

At the contrary of Viterbi, forward-backward durations can take non-integer values and duration probabilities are then defined on continuous values.

Implementation of forward-backward recursions

The alpha recursion in Eq. 4.14 at time n requires the computation of the sum over the permitted preceding states of the alphas at time $n-1$. This implies plenty of computational problems because the use of the *log* is now prohibited (summation over k in Eq. 4.14 and 4.15). The problem is coming from the fact that the alphas and betas are going under the precision of the computer (even using doubles) very quickly. Indeed, the fraction $\frac{p(x_n|q_l)}{p(x_n)}$ gives most of the time numbers between 0 and 1 (as observed in practice) and then it is obvious to see that α_n will have an exponent factor proportional to $-n$. A scaling is then necessary to avoid underflow precision. The scaling procedure of the forward-backward Gaussian mixture training can be used, as described for example in (Rabiner and Juang 1993).

Local vs. global discrimination

An interesting question is "*Since we are modeling global posterior probabilities, are we globally discriminant ?*". The answer is no. We are discriminant at the frame level when training the MLP but this does not guarantee discrimination at the word or sentence level. Indeed, phone models are usually shared between words and there are all the chance that training for a words like *dog* will make the posterior probability of similar words like for example *doggy* increase. A global discrimination would require that the score for *doggy* is explicitly decreased.

Nevertheless, in practice, several adjacent frames are concatenated to form the MLP input vector and the MLP estimates a posterior probabilities $P(q_k|X_{n-c}^{n+d})$ which is not so local.

Links with classic max-likelihood approach

The formulation presented above has been written to emphasize local posterior probabilities $P(q_{\ell_n}^n|X_{n-c}^{n+d})$ in the equations. If instead, local likelihoods are used (as in usual in HMMs) the following expression can be written (using exactly the same assumptions) :

$$P(M|X) \simeq P(M) \sum_{\ell_1,\ldots,\ell_N} \left[\prod_{n=1}^{N} P(X_{n-c}^{n+d}|q_{\ell_n}^n) \frac{P(q_{\ell_n}^n|q_{\ell_{n-1}}^{n-1}, M)}{P(X_{n-c}^{n+d})} \right] \tag{4.19}$$

Using Bayes rule, we can show that expression 4.12 and 4.19 are then similar :

$$\frac{P(X_{n-c}^{n+d}|q_{\ell_n}^n)}{P(X_{n-c}^{n+d})} = \frac{P(q_{\ell_n}^n|X_{n-c}^{n+d})}{P(q_{\ell_n}^n)} \tag{4.20}$$

The difference between the hybrid and the likelihood approaches lies at the local level. The hybrid system estimates local posteriors and is then discriminant at the frame level. The likelihood system estimates local probability density functions. Both systems can give us an estimate of the global posterior. Classically, the denominator $P(X)$ in Eq. 4.19 is dropped from the equations because it is constant at recognition time.

Soft segmentation

The Viterbi procedure considers the best state sequence sequence through the HMM. This can be interpreted as taking a hard decision about which state q_k is visited at time n. In other words, we can express $\gamma_n(k)$ in Eq. 4.16 while working with Viterbi, simply setting $\gamma_n(k) = 1$ if state q_k is visited at time n and setting $\gamma_n(k) = 0$ if the state is not visited. The forward-backward procedure can then be seen as a smoother version of the Viterbi procedure, since we have "soft" decision regarding which state q_k is visited at time n. We usually talk about *hard segmentation* when working with Viterbi and *soft segmentation* when working with forward-backward. Similarly, we use the terms *hard targets* and *soft targets* when we refer to MLP training targets.

Taking a smooth decision at the frame level makes more sense, especially at the boundaries between stationary parts of signal, and when the speech signal is degraded for some reasons. For this reason, we expect advantages of using a forward-backward criterion when training in difficult conditions : few training data, noisy data, strong coarticulation effects and bad or flat initialization of the parameter set.

4.3 Experiments

We first performed a set of experiments to demonstrate the differences between discrete, Gaussian mixtures and hybrid HMMs. We also report on comparison between Viterbi and forward-backward trained hybrid systems. Two telephone databases are used : HER and Phonebook. As far as possible, we performed the experiments using the same system configurations in order to ensure reliable comparisons when changing parameters.

4.3.1 HER database

We compare here the performance of classical discrete and Gaussian mixtures HMM systems with hybrid, Viterbi trained systems [3] using the HER database (see appendix A.2 for a description). This german spoken telephone speech database contains 108 phonetically balanced isolated words uttered by 536 talkers. The following settings were used :

- Data sets: 400 randomly choosen talkers were used to train the HMMs, 136 talkers were used as test set.
- Feature extraction: The 26 component feature vectors are composed of 12 LPC cepstrum coefficients, their first derivatives and the first and second derivatives of the log-energy. After preemphasis ($\alpha = 0.95$) and application of a Hamming window, ten LPC coefficients are used to compute the cepstrum. The acoustic vectors are computed every 10 ms over a 30-ms window. Cepstral mean substraction is operated session-wise. The algorithms are described in section 2.3.
- MLP: Training is performed using a stochastic backpropagation with random sampling in the training set. The correction of the matrices values is weighted by a *learning rate* value η which is updated after a presentation of the whole training set (epoch) by use of the following rule:
  - set $\eta_{i+1} = \frac{\eta_i}{2}$ if the error measure E on a independent cross-validation data set is increasing from epoch $i-1$ to epoch i.
  - set $\eta_{i+1} = \eta_i$ if the error measure E on a independent cross-validation data set is decreasing from epoch $i-1$ to epoch i.

  Inputs are scaled as explained in section 2.5.4. Weights and output biases are initialized as explained in section 2.5.5.
- HMM: Context independent phone HMMs are used. Each phone is represented by 3-states and the resulting HMM is strictly left-to-right model. Transition probabilities are fixed to 0.5 and are not reestimated during training.
- Training and decoding: We used the Viterbi criterion to train (10 iterations) and decode the HMMs. No minimum duration modeling has been used.

Comparative results are reported in table 4.2 and 4.3 for, respectively, context independent modeling and context dependent modeling. The following comments can be made :

- CIP discrete systems are the fastest systems but they use a large amount of memory and perform worse than Gaussian mixtures and hybrids.

[3] Hybrid systems results were obtained using the Lernout and Hauspie speech recognition toolkit while discrete and Gaussian mixture results were obtained using the CIRC toolkit.

- Given a similar amount of parameters, hybrid systems perform better than Gaussian mixture systems. Furthermore, the decoding is faster for hybrid systems. This is apparently due to a better exploitation of the microprocessor pipelines with the large matrix multiplications involved in the forward pass of the MLP.

- When comparing two hybrid systems having the same amount of parameters, the one with a larger context in input is better than the one with an increased number of hidden nodes. Further experiments have shown that contexts of about 7-9 frames lead generally to good performances.

- Discrete and Gaussian mixture CDP systems give very good performances but require a high amount of memory to store the parameters. There are no straightforward ways to build CDP hybrid systems without making explode the size of the MLP. Alternative methods to build networks able to model context dependency are presented in (Bourlard and Morgan 1993) and in (Cohen, Franco, Morgan, Rumelhart, and Abrash 1992). They report slight and consistent improvements but at the cost of more complicated implementations and larger CPU for decoding.

4.3.2 Phonebook database

We report here on two sets of experiments carried out on the Nynex task-independent Phonebook database (Pirelli et al. 1995) (see appendix A.1 for a more detailed description). The first experiment aims at comparing Gaussian mixtures HMMs with hybrid, Viterbi trained systems [4]. In the second one, a comparison of Viterbi and forward-backward training criterion is made[5].

The Phonebook task is qualified of *independent* because words in the test set are not represented in the training set. Sub-word modeling (CIP or CDP) is then required for this task. The advantage of task-independent systems lies in the fact that any vocabulary can be generated at recognition time. The database contains isolated word, telephone-speech signals. It consists of more than 92,000 utterances and almost 8,000 different words, with an average of 11 speakers for each word. Each speaker of a demographically-representative set of over 1,300 native speakers of American English made a single telephone call and read 75 words. The database contains 106 **lists** of 75 words. Each list is referred by a 2 letter label (for example `aa, ab`, ...). The speakers and words are different for each list. The CMU dictionary has been used to obtain the phonetic transcription.

Gaussian mixtures vs hybrids

We used the following settings :

- Data sets: The regular training set of 19,000 utterances (21 lists: `*a *h *m *q *t`) and a cross-validation set of 7,000 utterances (8 lists: `*o *y`), as defined in section A.1.

- Feature extraction: The 27 components feature vectors are composed of 12 LPC cepstrum coefficients, their first derivatives, the normalized log-energy, the first and second derivatives of the log-energy. After pre-emphasis ($\alpha = 1.0$) and application of a Hamming window, ten LPC coefficients are used to compute the cepstrum. The acoustic vectors are computed every 10 ms over a 30-ms window. Cepstral mean substraction is operated at the word level[6]. We refer to section 2.3 for a description of the feature extraction algorithms.

- MLP: Training is performed using a stochastic backpropagation with random sampling in the training set. The correction of the matrices values is weighted by a *learning rate* value η which is updated after a presentation of the whole training set (epoch) with the following rule:

  - set $\eta_{i+1} = \frac{\eta_i}{2}$ if the error measure E on a independent cross-validation data set is increasing from epoch $i-1$ to epoch i.

[4] For this set of experiments, the CIRC toolkit was used.

[5] For this set of experiments, STRUT and ICSI tools were used.

[6] In (Dupont, Bourlard, Deroo, Fontaine, and Boite 1997), CMS is performed session-wise on the Phonebook database which is somehow cheating. One should be aware of this when comparing results.

– set $\eta_{i+1} = \eta_i$ if the error measure E on a independent cross-validation data set is decreasing from epoch $i-1$ to epoch i.

A maximum of seven epochs have been used in all cases. Inputs are scaled as explained in section 2.5.4. Weights and output biases are initialized as explained in section 2.5.5.

- Gaussian mixtures: 15 Viterbi iterations were used to train the models.
- HMM: Context independent phone HMMs are used. Each phone is represented by one state and the resulting HMM is strictly left-to-right model. 40 phones are modeled with transition probabilities set to 0.5. Transition probabilities are not re-estimated during training.
- Decoding: The Viterbi criterion is used. The lexicon size is limited to the size of the lists, i.e. 75 words with a global test set size of 6,500 utterances (8 lists: `*d *r`). Two set of results are reported, one without any duration modeling, and the other with a minimum duration modeling as explained in 2.4.5.

Comparative results are reported in table 4.4. Similarly to results for the HER database, hybrid systems perform better than Gaussian mixtures systems when equal amount of parameters are used. Also, increasing the amount of parameters in the MLP results in better recognition performances.

Viterbi vs forward-backward

We demonstrate here the performance of hybrid systems using either the Viterbi or the forward-backward algorithms. We used two different training sets :

1. a small training set of 9,000 utterances and a cross-validation set (used to adapt the MLP training) of 2,000 utterances.
2. the regular training set of 19,000 utterances (21 lists: `*a *h *m *q *t`) and a cross-validation set of 7,000 utterances (8 lists: `*o *y`), as defined in section A.1.

The same settings as in the previous section were used, excepted for the lexicon size which is here enlarged to 600 words to better observe system differences, and also for the acoustic features, which are here 12 log-rasta PLP (+ delta-features + delta-energy) (Hermansky and Morgan 1994).

We used a multilayer perceptron with 234 inputs (9 frames of input context). For training set 1, a MLP of 600 hidden units has been used. For training set 2, a MLP of 1000 hidden units has been used. A minimum duration model is used in both Viterbi and forward-backward (see section 2.4.5 for details on minimum duration models).

In table 4.5, these results show a clear advantage of forward-backward training over Viterbi training for the small training set. No significant difference is observed in the case of the larger training set (Viterbi seems to be slightly better). A graphical analysis of the ANN targets is given in Fig. 4.3 and 4.4. The idea is to get a better understanding of the behaviour of both algorithms and to put forward plausible explanations for their difference of performances. The ANN targets are given with

$$\gamma_n(k) = P(q_k^n | X, M) \tag{4.21}$$

where gammas values define the most likely state at every instant. They are depicted for Viterbi and forward-backward algorithms, respectively in the middle and the bottom parts of the figures. Frames are on the x-axis, HMM states on the y-axis and ANN targets are depicted with gray boxes (the darkest, the closest to 1). Target values are either 0 or 1 for the Viterbi criterion, while it can be any continuous value between 0 and 1 for the forward-backward criterion. There are very few differences between Viterbi and forward-backward targets, except at the boundary between phones where the forward-backward criterion does not take clear-cut decision regarding the state membership.

System	#phones	# states	# parameters	decoding CPU load	Error rates
Discrete	42	126	52 K	1	6.1 %
4 Gaussian	42	126	27 K	15	5.9 %
8 Gaussian	42	126	53 K	30	5.0 %
Hybrid C00 181HU	42	126	28 K	10	4.2 %
Hybrid C33 90HU	42	126	28 K	10	3.5 %
Hybrid C00 345HU	42	126	53 K	19	3.6 %
Hybrid C33 171HU	42	126	53 K	19	3.0 %

Table 4.2: Results are obtained on the HER database with discrete, Gaussian mixtures and continuous Hybrid HMMs using CIP models. Cxy defines the input size of the MLP and means that x frames on left and y frames on right of the central frame are taken. Number of HMM phones and states are given in columns 2 and 3. The number of parameters, conditioning the required memory to run the system, is reported in column 4. The decoding time is reported in column five, taking as reference the discrete system. Error rates are given in the last column.

System	#phones	# states	# parameters	decoding CPU load	Error rates
Discrete	317	951	396 K	1	2.4%
8 Gaussian	317	951	400 K	30	2.8%

Table 4.3: Results are obtained on the HER database with discrete, Gaussian mixtures and continuous Hybrid HMMs using CDP models.

System	#phones	# states	# parameters	ER	ER (dur)
8 Gaussian	40	40	17 K	18.4 %	8.1 %
16 Gaussian	40	40	34 K	17.1 %	7.5 %
MLP C11 128HU	40	40	16 K	11.4 %	5.7 %
MLP C11 275HU	40	40	34 K	8.8 %	4.2 %
MLP C11 400HU	40	40	49 K	8.2 %	3.7 %
MLP C22 190HU	40	40	34 K	10.0 %	4.8 %
MLP C22 275HU	40	40	49 K	8.6 %	4.0 %

Table 4.4: Results on the Phonebook database with Gaussian mixtures and Hybrid HMMs using CIP models. Number of HMM phones and states are given in columns 2 and 3. The number of parameters is reported in column 4. Error rates are given in the last two columns (no minimum duration modeling and minimum duration modeling). Hybrid systems are trained with the Viterbi criterion.

Word error rate	Training set 1	Training set 2
Viterbi	13.7 %	9.8 %
Forward-backward	12.2 %	10.1 %

Table 4.5: The error rates are obtained on the Phonebook database with continuous Hybrid HMMs for two training sets with different sizes. Training set 1 is considerably smaller than training set 2. For training set 1, a C44 MLP with 600 hidden units has been used. For training set 2, a C44 MLP with 1000 hidden units has been used.

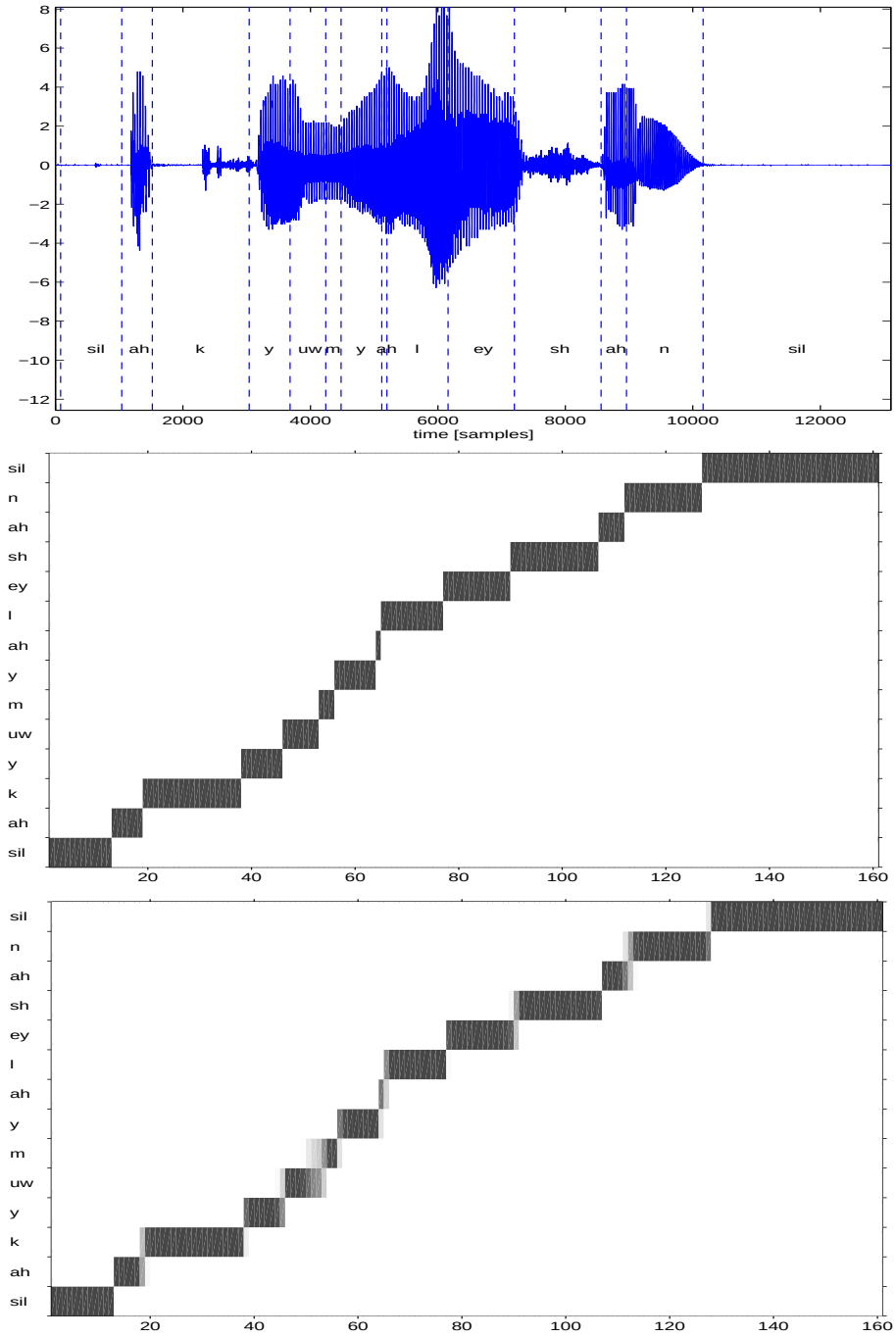


Figure 4.3: Words `accumulation` from the Phonebook database. a) Waveform with Viterbi alignment. b) Gammas with Viterbi criterion. c) Gammas with forward-backward criterion.

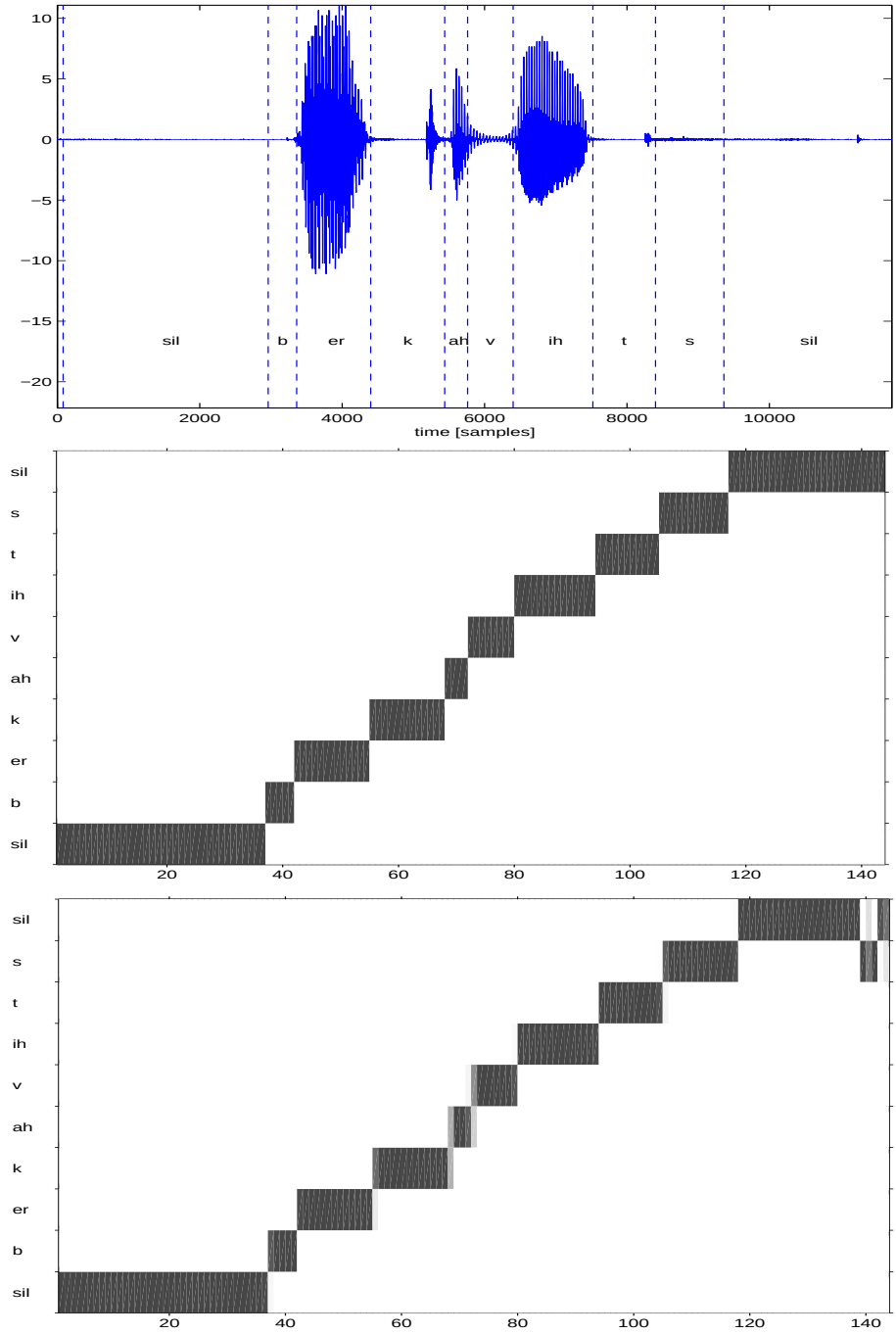


Figure 4.4: Words `berkovitz` from the Phonebook database. a) Waveform with Viterbi alignment. b) Gammas with Viterbi criterion. c) Gammas with forward-backward criterion.

Forward-backward makes much more sense than Viterbi because the vocal tract does not move instantly from one position to the other. Viterbi is therefore not realistic when it takes hard state membership decision for features falling at the phone boundaries. Nevertheless, the amount of vector in this case is quite small and the hard decision effect should be smoothed out when large database are used.

This favorable feature of forward-backward is maybe counterbalanced by another feature: forward-backward does not guarantee that the state path defined with

$$q_n^* = \arg\max_i [\gamma_n(i)] \qquad 1 \leq n \leq N \tag{4.22}$$

is a valid state sequence. This is because the solution of Eq. 4.22 simply determines the most likely state at every instant, without regard to the probability of occurrences of sequences of states. This statement is well illustrated for the last frames of the `Berkowitz` word in Fig. 4.4. The Viterbi criterion does not permit these kind of undesired targets and this is a possible explanation for the better results of Viterbi on the large training set.

4.4 Conclusions

In this chapter, we have analysed hybrid HMM/ANN systems in which the local frame hypothesis are estimated with MLP. Previous studies have already been dedicated to hybrid systems (Bourlard and Morgan 1993; Bourlard and Morgan 1994; Renals, Morgan, Bourlard, M.Cohen, and Franco 1994). The work presented here can be viewed as a extension to these studies.

Starting from a description of the original hybrid HMM/ANN system, we have derived a new theoretical formalism to describe the behaviour of such systems. The theoretical perspective provides us with a better unified view of hybrid systems and their relationships to standard HMMs. In short, the theory shows that such systems can actually be used to compute global model posteriors, provided that the ANN part estimates local posteriors. The classic Viterbi trained hybrid system and the more recent REMAP system are clearly unified with the theory.

This better formalism inspired us to derive a new forward-backward training for hybrid systems. It is shown how to implement the training with the computation of new ANN targets. It is also explained how to compute state duration and priors with the forward-backward scheme.

Several sets of experiments have been carried on isolated word telephone databases. Firstly, we have compared Gaussian mixtures and hybrid systems on the Her database reporting on memory bandwidth, CPU cost and error rates. Results show that, for similar amount of parameters, hybrid systems perform systematically better than Gaussian mixture systems, both in terms of error rates and CPU speed/memory cost. These experiments are confirming the conclusions of previous studies (see e.g. (Bourlard and Morgan 1993)). We cross-validated these experiments on a more difficult task using the Phonebook database. Again, hybrid systems are performing better than Gaussian mixtures and the benefits are even more obvious on this database.

Second, we have performed experiments to compare Viterbi and forward-backward training of hybrid systems. Recognition results obtained on Phonebook show a clear advantage for forward-backward training when a small training set is used. For larger training sets, Viterbi seems to give equivalent or better recognition rates. We have also presented a graphical analysis of the ANN targets in order to better understand the behaviour of both algorithms. Targets obtained with forward-backward are very similar to the one obtained with Viterbi, except at phone boundaries where the forward-backward criterion does not take clear-cut decision regarding the state membership. Forward-backward makes much more sense than Viterbi because the vocal tract does not move instantly from one position to the other. Viterbi is less realistic when it takes hard state membership decision for these features falling at the phone boundaries. Nevertheless, the amount of vectors in this situation is pretty low and the hard decision effect of Viterbi should be smoothed out when large database are used. The explanations are indeed confirmed through the experiments carried on the Phonebook database.

Bibliography

Bourlard, H., Y. Konig, and N. Morgan (1995, September). Remap : Recursive estimation and maximization of a posteriori probabilities in connectionist speech recognition. In *EUROSPEECH'95*, Madrid.

Bourlard, H., Y. Konig, and N. Morgan (1996). A training algorithm for statistical sequence recognition with applications to transition-based speech recognition. *IEEE Signal Processing Letters 3*(7), 203–205.

Bourlard, H. and N. Morgan (1993, November). Continuous speech recognition by connectionist statistical methods. *IEEE Transactions on Neural Networks 4*(6), 893–909.

Bourlard, H. and N. Morgan (1994). *Connectionist Speech Recognition*. Kluwer Academic Publishers.

Cohen, M., H. Franco, N. Morgan, D. Rumelhart, and V. Abrash (1992). Hybrid neural network/hidden markov model continuous speech recognition. In *Proceedings of Intl. Conf. On Speech and Language Processing*, Banff, Canada, pp. 325–328.

Dupont, S., H. Bourlard, O. Deroo, V. Fontaine, and J.-M. Boite (1997). Hybrid hmm/ann systems for training independent tasks: Experiments on 'phonebook' and related improvements. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Munich, Germany, pp. 1767–1770.

Hermansky, H. and N. Morgan (1994, October). Rasta processing of speech. *IEEE Transactions on Speech and Audio Processing, special issue on Robust Speech Recognition 2*(4), 578–589.

Pirelli, J. et al. (1995). Phonebook: A phonetically-rich isolated word telephone speech database. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Detroit.

Rabiner, L. and B.-H. Juang (1993). *Fundamentals of Speech Recognition*. Prentice Hall.

Renals, S., N. Morgan, H. Bourlard, M.Cohen, and H. Franco (1994). Connectionist probability estimators in hmm speech recognition. *IEEE Trans. on Speech and Audio Processing 2*(1), 161–174.

Robinson, T. and F. Fallside (1991). A recurrent error propagation network speech recognition system. *Computer Speech and Language 5*, 259–274.

Robinson, T., M. Hochberg, and S. Renals (1996). *Automatic Speech and Speaker Recognition – Advanced Topics*, Chapter The use of recurrent networks in continuous speech recognition, pp. 233–258. Kluwer Academic Publishers.

Senior, A. and T. Robinson (1996, December). Forward-backward retraining of recurrent neural networks. In *Advances in Neural Information Processing Systems 8*, Denver, Colorado, pp. 743–749. MIT Press.

Yan, Y., M. Fanty, and R. Cole (1997). Speech recognition using neural networks with forward backward probability generated targets. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Munich, Germany, pp. 3241–3244.

Chapter 5

A segmental approach for speaker verification

If we knew what it was we were doing, it would not be called research, would it?

A. Einstein

In recent years, speaker recognition technology has made a lot of progress, but a great deal of issues still remain open. The generic term of speaker recognition comprises all of the many different tasks of distinguishing people on the basis of their voices. There are *speaker identification* tasks - who among many candidate speakers pronounced the available test speech sequence; or *speaker verification* tasks - whether a specific candidate speaker said the available test speech sequence. We are interested in this work on speaker verification. Verifying the identity of a speaker is actually a decision problem between two classes : the *true* speaker (also denominated as *client* or *target* speaker) and the *other* speakers (usually noted as *impostors* or *world* speakers).

As far as the speech mode is concerned, speaker recognition systems can be *text-dependent* or *text-independent*. In text-dependent experiments, the speech sequence used to distinguish the speaker is known. In text-independent tasks, the foreknowledge of what the speaker said is not available.

From the user point of view, text-dependent systems are maybe less convenient than text-independent systems because the user is constrained to remember a password or to repeat a prompted text [1]. Text-dependent systems perform generally better than text-independent systems. There are two reasons for this difference :

- The knowledge of what has been said can be exploited to align the speech signal into more discriminating classes (words or more generally, sub-word speech units). A simple automatic Viterbi alignment can be used during both training and testing time. Speaker and impostor models can then be trained at sub-word levels. Good results are generally reported using this technique (Matsui and Furui 1993).

[1] On the other hand, some users have so few imagination that when asked to speak freely, they are unable to give any speech sample.

- An optimized recombination of these individual sub-word decisions can be done. Several studies on text-dependent systems (Eatock and Mason 1994) (Olsen 1997) (Petrovska and Hennebert 1998) (Hennebert and Petrovska 1998) have demonstrated that some phones show more speaker discriminative power than others [2]. Other studies (Besacier and Bonastre 1998) (Besacier and Bonastre 1997) have also demonstrated that pruning out frames carrying less speaker specific informations may improve system performances. These previous works suggest that a weighting of these individual scores should be performed in order to compute the global decision[3].

We are interested here in building robust text-independent systems. Since the contents of the speech signal is not directly accessible, text-independent speaker verification is usually based on modeling the **global** probability distribution function (pdf) of speakers in acoustic vector space, for example by using Gaussian mixture modeling. We believe that such global approaches are reaching their limits because speaker modeling is somehow too coarse in this case.

Therefore, we propose here to investigate a **segmental** approach in which the speech signal is preclassifyied into keener units. On the one hand, the segmental approach recovers text-dependent advantages since the speech signal is automatically aligned into classes. On the other hand, the implementation is different since we have no clue about what has been said. As for text-dependent systems, we can underline two potential advantages. First, if the speech units are relevant, then speaker modeling is more precise and the system should present better performances than the global approach. Second, if speech units present different discriminative power, then a weighted recombination of individual decisions can be done.

In the first section of this chapter, we describe the different blocks of a baseline global text-independent system. In the second section, we report on a preliminar set of experiments conducted on a text-dependent task for which we know the lexical content of the speech material. This step is taken in order to confirm our assertions and believes in a text-independent segmental approach. In the third section, we present the segmental system that we are going to investigate. The system is divided into three parts: segmentation-labelling, segment modeling and score recombination. In the last section, we report performances comparison of two global systems and the new segmental approach. Implementation issues and future work are also discussed.

5.1 Baseline text-independent speaker verification

A classical text-independent speaker verification system can be presented in three blocks, as illustrated in Fig. 5.1 :

1. **Feature Analysis**: Similarly to what is done for speech recognition, the speech signal is cut into analysis windows undergoing feature extraction. A feature vector is computed for each window so that the speech signal is transformed into a sequence of feature vectors $X = \{x_1, x_2, \cdots, x_N\}$ of length N. Classical feature extraction algorithms are lpc-cepstral analysis and mfcc analysis. Further details can be found in section 2.3 of this text and in (Rabiner and Juang 1993) or (Picone 1993). A comparative review of some feature analysis for speaker verification can be found in (Homayounpour and Chollet 1994).

2. **Pattern Classification**: The sequence of feature vectors is fed into a classifyer that outputs a likelihood score for the client model and the world model, respectively S_c and S_w. We describe in a more detailled way the operations and the methods involved in this block hereafter.

3. **Thresholding**: The verification (reject/accept) of the speaker is performed comparing the ratio of client and world score against a threshold value. There exists different strategies to set the threshold in order to reach a given performance. They are also discussed further in this section.

[2] This fact has been previously validated by phonetic studies (Nolan 1983) in which it was reported that the information about the identity of a speaker is not uniformly distributed among speech segments (phones).

[3] This feature suggests also an interesting way to optimize text-prompted systems(Hennebert and Petrovska 1998). The prompting could also be driven by this information in order to obtain more occurrences of these phones in the speech input.

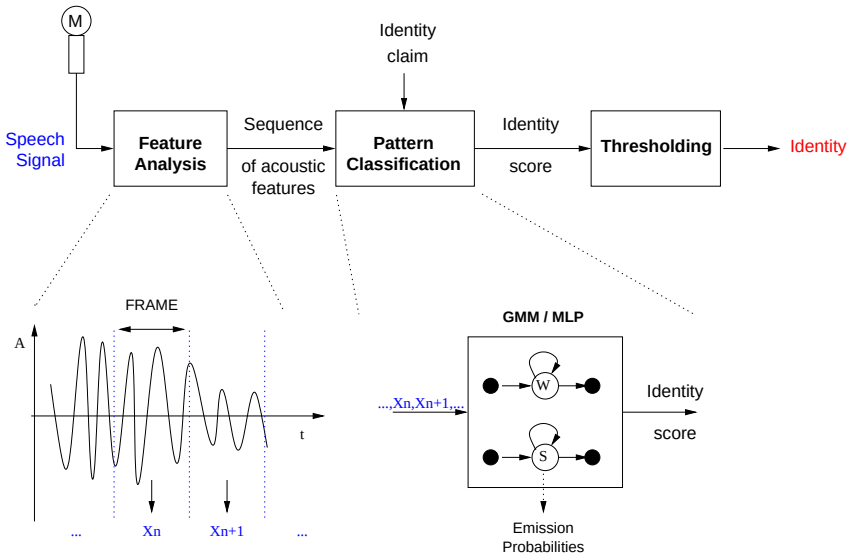


Figure 5.1: Speaker verification system.

5.1.1 Pattern classification

Fig. 5.1 shows a classical way to do pattern classification in text-independent systems, referred here as the **global** method. Assuming independence of successive acoustic vectors, a unique model is tied to the whole vector sequence.

There are several ways to build speaker models : vector quantization, second order statistical methods, gaussian mixtures, ANNs,... We focus here on gaussian mixture and ANN modeling. The first one because it is a classical method employed in many state-of-the-art systems. The second one because we believe it has better modeling capabilities (see discussion in section 4.1).

GMM

Gaussian mixture models (GMM) are used to build estimates of the multivariate distribution. The mixture is a weighted sum of gaussians (see e.g. (Reynolds 1995)) :

$$P(x_n|M) = \sum_{j=1}^{J} w_j \mathcal{N}(x_n, \mu_j, \Sigma_j) \tag{5.1}$$

with the constraint $\sum_{j=1}^{J} w_j = 1$. As illustrated in Fig 5.1, GMM can be viewed as a particular case of Hidden Markov Model (HMM) where there would be a unique state in the automaton. The HMM emission probability is then equal to the output of the unique probability density function tied to the state. Two models are built in, one for the client, one for the world. With the usual HMM assumptions (see section 2.4.1), global likelihood scores are equal to the product of frame likelihoods over the whole sequence :

$$S_c = \quad p(X|M_{client}) \quad = \prod_{n=1}^{N} p(x_n|M_{client}) \tag{5.2}$$

$$S_w = \quad p(X|M_{world}) \quad = \prod_{n=1}^{N} p(x_n|M_{world}) \tag{5.3}$$

MLP

Another possibility to model speakers is to build discriminant models as, for example, multi-layer perceptrons (MLP). We refer to section 2.5 for a more detailled introduction to MLP. A three-layers MLP is depicted in Fig. 2.13 and 2.14.

In previous studies , MLPs have successfully been used for text-independent (Oglesby and Mason 1990) (Farrell, Mammone, and Assaleh 1994) and for fixed-text (Naik and Lubenskt 1994) speaker recognition tasks. We refer to (Bennani and Gallinari 1994) for a review of connectionist approaches for speaker recognition. The main advantages of MLPs against other systems like GMM include, among others, discriminant capabilities, weaker hypotheses on the acoustic vector distributions and possibility to include a larger acoustic frame window as input of the classifier.

In text-independent tasks, MLPs, one per client speaker, are discriminatively trained to distinguish between the client and the world. MLPs with two outputs are generally used, one for the client and the other for the world class. As explained in section 2.5 and in (Bourlard and Wellekens 1990) it has been shown that if each output unit k of the MLP is associated to class categories C_k, it is possible to train the MLP to generate a posteriori probabilities $p(C_k|x_n)$. During training, as explained in (Rumelhart, Hinton, and Williams 1986), the parameters of the MLP (weight matrices) are iteratively updated via a gradient descent procedure in order to minimise the difference between gotten outputs and desired outputs (targets). In our case, target vectors $d(x_n)$ are set to $[1, 0]$ and $[0, 1]$ when the input vector x_n is produced respectively by the client and by the world speaker. The training is said to be *discriminative* because it minimizes the likelihood of incorrect models (through the zeros of the target vector) and maximizes the likelihood of the correct model. The network attempts to model class boundaries, rather than accurate probability density functions for each class.

As shown with Bayes'rule :

$$P(C_k|x_n) = \frac{p(x_n|C_k)P(C_k)}{p(x_n)} \tag{5.4}$$

the MLP training incorporates priors $P(C_k)$ when modeling posterior estimates. These priors are not equal to real-life class prior probabilities but are dependent on the number of samples in the training set :

$$P(C_k) = \frac{N_k^{ts}}{N^{ts}} \tag{5.5}$$

where N^{ts} is the total number of training patterns and N_k^{ts} is the number of training patterns of the class k. Posteriors have then to be scaled to remove the dependency on the number of training patterns[4]. Posterior $P(C_k|x_n)$ are then divided by priors $P(C_k)$ to obtain the so-called *scaled likelihoods* $p(x_n|C_k)/p(x)$. Scaled likelihoods can be used in place of likelihoods at recognition time since the factor $p(x)$ is not dependent on the class.

Client and world scores are then computed according to Eq. 5.2 and 5.3 where likelihoods $p(x_n|client)$ and $p(x_n|world)$ are replaced with scaled likelihoods.

5.1.2 Thresholding

The last block of the speaker recognition system as depicted in Fig. 5.1 is the thresholding block, in which the decision wether to accept or to reject a claimed identity is taken. The decision is performed comparing the ratio of client and world score against a threshold value. The ratio is often computed in the log-domain with :

$$R_c = \log(S_c) - \log(S_w) \tag{5.6}$$

[4]This is even more necessary in the case of speaker verification where the number of patterns in the client class is extremely low in comparison to the number of patterns in the world class.

where R_c is called the log-likelihood ratio. The accept/reject decision is taken as follows :

$$R_c > T \quad \rightarrow accept \tag{5.7}$$
$$R_c \leq T \quad \rightarrow reject \tag{5.8}$$

The threshold T is usually defined to minimize the cost value C_{det}, a weighted measure of false rejection and false acceptation probabilities :

$$C_{det} = C_{fr} P(reject|client) P(client) + C_{fa} P(accept|\overline{client}) P(\overline{client}) \tag{5.9}$$

where C_{fr} is the cost of false rejection, C_{fa} is the cost of false acceptance, $P(reject|client)$ is the probability to reject a client, $P(accept|\overline{client})$ is the probability to accept an impostor, $P(client)$ is the a priori probability of client and $P(\overline{client})$ is the a priori probability of impostor.

Speaker verification is actually a *detection task* involving a tradeoff between the two types of error: missed detections (the system fails to detect a target speaker) and false alarms (the system accept an impostor speaker).

ROC and DET curves

In Eq. 5.9, $P(reject|client)$ and $P(accept|\overline{client})$ are both function of the threshold value T. Given a test set, false acceptance (FA) and false rejection (FR) curves can then be plot giving different values to T. FA and FR curves are often represented as receiver operating curves (ROC)(Egan 1975). Generally, FA rate is plotted on the horizontal axis and and FR rate is plotted on the vertical. ROCs are sometimes not practical when similar systems need to be compared and an alternative representation of the ROC is used: Detection Error Tradeoff (DET)(Martin, Doddington, Kamm, Ordowski, and Przybocki 1997) in which the x and y scales are in the log domain. If the likelihood ratios are normally distributed (this is often observed in practice), the DET curve will be close to a straight line,enabling easy observation of system contrasts. Example of DET curves are given hereafter as, for example, in Fig. 5.6.

Equal error rate

The threshold value where false acceptance and false rejection are equal defines the *equal error rate* (EER). This value is often used to compare the performances of different systems. From a practical point of view, EER is not very realistic because it corresponds to a situation in which the threshold is computed a posteriori with pre-classified speech material. In practice, the threshold needs to be set a priori and this is not an easy matter.

Maybe even more unrealistic, the so-called gender balanced sex-independent (GBSI) EER can be computed, following EAGLES recommendations (Bimbot and Chollet 1995). GBSI-EER are computed setting a posteriori different thresholds for each client while weighting contributions from male and female impostors to have equal importance, i.e. if there are twice as many male impostors than female, let scores from female impostors have double importance. The global EER is then computed from a weighted average of individual EERs, so that male and female population have again equal importance (same principle as above). GBSI is an optimistic error rate value one can measure on a system but permits to normalize system performances regarding sex unbalances.

One could wonder why a thresholding is necessary to compute EER. Indeed, in the case when MLPs are used to classify the speech material, it could be argued that if they are actually estimating the posterior probabilities of the classes, it would not be necessary to use a thresholding procedure to determine the EER. The same discussion can also take place for non-discriminant likelihood approaches in which in theory, a majority vote on the class likelihood should be enough to determine the EER. The problem lies, for likelihood estimators and for a posteriori probability estimators, in the fact that they are biased estimators due to the lack of training datas. This bias needs to be compensated with a thresholding.

Znorm and hnorm

EERs are computed a posteriori, giving an optimistic picture of the performance of systems. In real-life, thresholds need to be set a priori, before running the system in the field. Setting a global threshold would be more practical in this case, but client likelihood scores are not distributed in a similar manner for different clients. There are two main reasons for this : (1) Biased modeling due to the lack of training data and (2) Mismatches between training and testing conditions (different channel, microphones, client health or emotional state, ...).

A global thresholding is therefore not optimal because each client model will generate likelihood ratio with different distributions. An easy practical way to bypass this difficulty is to normalize likelihood ratios in order to smooth out the influences of distribution variabilities. A practical way is to normalize log-likelihood ratio with impostor trials which can easilly be generated. The procedure, referred as **znorm**[5] is as follows :

- Assuming a gaussian distribution, a large amount of impostor attempts is used to estimate the log likelihood ratio average and variance for each client model. Estimates are generally computed on a population of impostor having the same sex as the client.
- During testing, the log-likelihood ratio is normalized using these average and variance values.

A global thresholding on the normalized log-likelihood ratio can then be employed in a more efficient way[6].

Bimodal distribution of log-likelihood scores have been observed in practical telephone-based tasks. The two modes are due to handset microphones which are generally of two kinds : electret or carbon. Since the handset type can be easilly detected, a handset-dependent normalization can be performed (Reynolds 1997). This procedure generally leads to better performances when the client uses mismatched handset for registering and using the system. This kind of normalization is referred as **hnorm** for handset-normalization. A comparison of znorm and hnorm normalization techniques can be found in (Gravier and Chollet 1998).

5.2 Preliminary experiments

Before starting the development of a rather complex text-independent segmental system, we would like, in a first place, to confirm the two assertions :

- modeling speaker on sub-word bases is a realistic approach;
- some sub-word classes should show more speaker discriminative power than others.

We propose to perform several experiments on a text-dependent task for which we can derive easily a fairly good segmentation. Text-dependency guarantees here that we have at our disposal the lexical contents of each utterance.

5.2.1 System description

The text-dependent system (which could actually be a text-prompted system) has two parts.

1. A **speech recognition part** developped with a set of 42 context independent phoneme (CIP) HMMs that are trained in a speaker-independent manner. The HMMs are used to generate automatically segmentation into phonemes with a simple Viterbi forced alignment. We refer to section 2.4.4 for a description of the algorithms. Each feature vector is then labelled with the corresponding phonemes.

[5] In znorm, "**z**" stands for centered.
[6] Actually, the znorm normalization can be seen as a roundabout way to apply a per-client thresholding.

2. The **speaker verification part** formed with a set of MLPs, one for each phoneme/speaker. They are discriminatively trained to distinguish between the target speaker and the world model. MLPs with two outputs are used, one for the client class C_1 and the other for the world class C_2. As explained in section 2.5, a 1-from-K training makes the MLP estimates a posteriori probabilities $p(C_k|x_n)$ when x_n, a particular acoustic vector, is provided to its input.

Similarly to what is done in speech recognition with hybrid HMMs/MLP systems (Bourlard and Morgan 1994), this approach combines the ability of HMMs to handle efficiently the sequential character of speech and the discriminant properties of ANNs. The main drawback using MLPs is that its optimal architecture (essentially the number of hidden nodes) must be selected by trials and errors.

Once the speech is aligned with phonemes, the output of the MLP provides estimates of the client and world a posteriori probabilities at the frame level. The client and world scores of a sequence of N vectors belonging to a phoneme k are obtained as follows :

$$S_{1k} = \sum_n log(\frac{p(C_1|\mathbf{x}_n)}{P(C_1)}) \tag{5.10}$$

$$S_{2k} = \sum_n log(\frac{p(C_2|\mathbf{x}_n)}{P(C_2)}) \tag{5.11}$$

The recombination of scores S_{1k} and S_{2k} in order to take a decision at the word level is not investigated here. Instead, speaker verification EER are computed directly on the S_k measures in order to study the discriminative power of the different phonemes.

5.2.2 Experimental setup

We give hereafter a summary of the experimental. For a more thorough description, we refer to (Petrovska and Hennebert 1998) and (Hennebert and Petrovska 1998).

- **MLP architecture**: MLPs with one input layer, one hidden layer and one output layer of neurons are used. Hidden and output layers are computational layers with a sigmoid as activation function. It has been previously shown (Oglesby and Mason 1990) that using more than one hidden layer did not improve the performance for a speaker identification task and thus this architecture has not been investigated here.

- **MLP training**: During training, target vectors $t(x_n)$ are set to $[1, 0]$ and $[0, 1]$ when the input vector x_n is produced by, respectively, the client and by the world speaker. The acoustic vectors are presented randomly from the available training set. The error criterion used for training is defined as in Eq. 2.45 and the non-linear vector function operated by the MLP on the input vector is described in Eq. 2.44. The parameters of the MLP (weight matrices) are iteratively updated via a stochastic gradient descent procedure in order to minimise the error criterion, i.e. weights are updated after every input presentation during the training process. The correction of the matrices values is weighted by a *learning rate* value η which is updated after a presentation of the whole training set (epoch) with the following rule:

  - set $\eta_{i+1} = \frac{\eta_i}{2}$ if the error measure E on an independent cross-validation data set is increasing from epoch $i-1$ to epoch i.
  - set $\eta_{i+1} = \eta_i$ if the error measure E on an independent cross-validation data set is decreasing from epoch $i-1$ to epoch i.

  Observing an increasing error measure on an independent cross-validation data set from one epoch to another is a sign of over-fitting on the training data set. In order to avoid over-fitting, the update of the weight matrices is discarded before setting the new learning rate value and pursuing with the next epoch. Training is stopped when η falls below a pre-determined value.

- **Sampling procedure**: Two different sampling procedures are used to present the acoustic vectors as input to the MLP's during training. For the first procedure, referred as baseline (BL), the acoustic vectors are picked randomly in the whole training set, built up with the client and world vectors. For the second procedure, referred as Equal File Sampling (EFS), the acoustic vectors are also picked randomly in the training set, but this time taking successively one vector in the client training set and one vector in the world training set. The EFS sampling mode is expected to avoid problems due to large size difference between client and world training sets. Performing EFS does not change the behaviour of the MLP, in the sense that it will still estimate posterior probabilities. The difference lies in the class prior values that are equal in the case of EFS, as explained in the Bayes formula:

$$P(C_k|x_n) = \frac{p(x_n|C_k)P(C_k)}{p(x_n)} \tag{5.12}$$

$$P(C_k) = 0.5 \qquad if EFS \tag{5.13}$$

  When using EFS, a training epoch is terminated when all the feature vectors of the largest training set are visited once. This implies that vectors in the smaller set may be visited more than once during the same epoch.

- **Database description**: The HER Swiss German telephone speech database has been used for the experiments (see appendix A.2 for more details). This database contains 108 phonetically balanced isolated words uttered by 536 speakers. 25 male speakers are selected as the clients of the system. 25 other male speakers are selected to constitute the world model and 25 male speakers are used as impostor speakers. In order to minimize the influence of lack of training data when building the models, a reduced set of 14 phonemes appearing more than 22 times is selected. The training data set for each phoneme model is obtained from a concatenation of 8 client segments and 200 world segments. Independent cross-validation sets are defined in the same way, concatenating 5 segments of each phonemes for the client and 125 segments for the world. 2 true-identity tests are defined for each speaker phoneme model, concatenating 4 and 5 segments. 125 impostor tests are defined for each speaker phoneme model, concatenating 4 segments. In order to have more testing material, 8 distinct train, cross and test data sets are selected from the 22 phoneme occurrences available by concatenating segments in different order. The total of true-identity and impostor tests are then respectively 400 and 25000.

5.2.3 Results

Results by phoneme

Figure 5.2 shows the averaged EER for the 14 selected phonemes. Results are obtained with a 20 hidden nodes MLP trained with 3 consecutive acoustic frames as input using the baseline sampling procedure. The best performance is observed with phoneme *n* which is a nasal. Vowels (*E, e, A, AA, I*) and fricatives (*s, f*) give good and similar performances while plosives (*g, t, k*) and liquids (*l, r, R*) convey less speaker specific informations. Per speaker detailed results show that some phonemes perform better for some speakers while the same phonemes have poor performance for other speakers.

We point out that results are obtained training MLPs with the same occurrence of each individual phonemes and no length normalisation of the segments is performed. Very similar results are reported in (Eatock and Mason 1994) in which a phonetically hand-labelled database is used to train a VQ based speaker verification system.

Influence of the MLP input frame context

The influence of the acoustical window length at the input of the MLP is investigated adding symmetrically left and right frames to the central frame. Experiments with 1, 3, 5, 7 and 11 successive frames

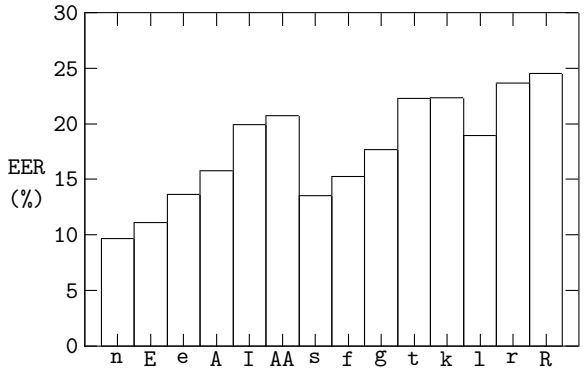


Figure 5.2: EER averaged per phoneme with 20 hidden nodes and 3 input frames for the MLPs.

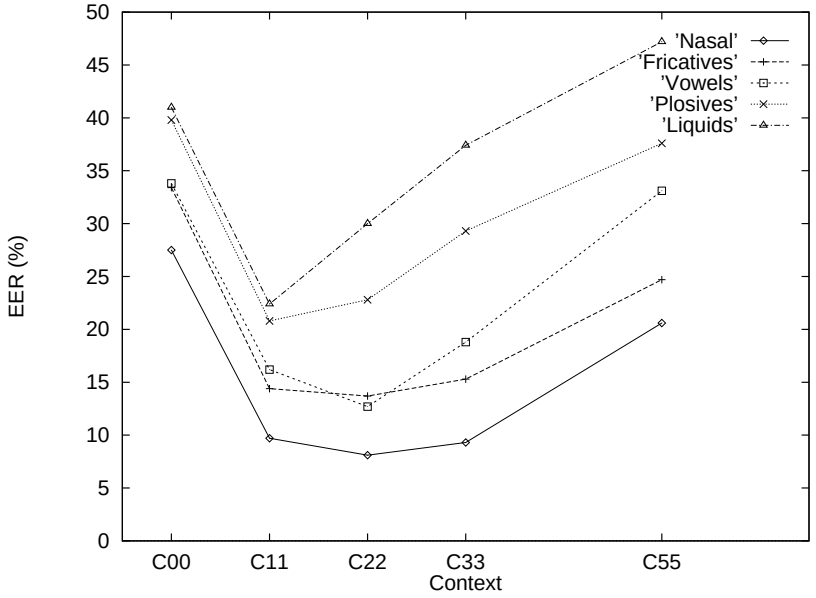


Figure 5.3: EER averaged per phonemic group with different acoustic window length at the input of the MLP. The number of hidden nodes equals 20 and is kept constant for all the experiments. Cxy means x left frames and y right frames of context as MLP input.

	Nasal	Fricatives	Vowels	Plosives	Liquids
C00-20	27.5	33.4	33.8	39.8	41.0
C11-20	9.7	14.4	16.2	20.8	22.4
C22-20	8.1	13.7	12.7	22.8	30.0
C33-20	9.3	15.3	18.8	29.3	37.4
C55-20	20.6	24.7	33.1	37.6	47.2

Table 5.1: EER averaged per phonemic group with different acoustic window length at the input of the MLP. The number of hidden nodes is kept constant.

	Nasal	Fricatives	Vowels	Plosives	Liquids
C00-10	28.1	32.6	36.4	41.4	41.7
C00-20	27.5	33.4	33.8	39.8	41.0
C00-54	31.9	36.2	38.4	43.3	43.8

Table 5.2: EER averaged per phonemic group with different number of hidden nodes in the MLP.

at the input of the MLP are reported on figure 5.3 and are noted as Cxy (meaning x left frames and y right frames of context taken into account). EER are averaged in broad phonetic classes for clarity's sake. Significant improvements are brought when increasing the acoustic window size from $C00$ to $C11$. This result suggests that the frame-to-frame temporal informations convey important speaker specific informations. Increasing furthermore the size of the input to $C22$ improves somehow performances for nasal, fricatives and vowels while plosives and liquids get worse performances. Performing the training with 7 ($C33$) and 11 ($C55$) successive frames degrades performances, except for phonemes e and f in the case of $C33$. Results in figure 5.3 show also that each phoneme class has its optimal MLP input size which gives the best EER. For nasal n, vowels and plosives, the best results are obtained with $C22$, while for plosives and liquids better performances are obtained with $C11$. One should point out that the general degradation introduced for large context ($C33$ and $C55$) can also be a sign of a lack of training data considering the large number of parameters to estimate. Indeed, using larger contexts increases considerably the number of parameters.

Results reported in figure 5.2 and 5.3 are all obtained with MLPs having 20 nodes on the hidden layer and the baseline random sampling procedure is used for training. The inclusion of delta coefficients at the input of the MLP has not been investigated here.

The improvements obtained when larger acoustic frame windows are used are quite important for the configuration investigated here. In the literature, different text-independent predictive systems have been proposed with mitigated results to specifically include the temporal information: predictive MLP's (Montacie, Deleglise, Bimbot, and Caraty 1992) (Hattori 1994) (Artières 1995) giving encouraging results and Auto-Regressive (AR) vector models (Magrin-Chagnolleau, Wilke, and Bimbot 1996) giving contradictory results. Compared with these results, the large amelioration reported here is probably due to the fact that it is easier to exploit the temporal information at the phoneme level than at the word level.

Influence of the number of parameters

Performances obtained with different number of hidden units are reported in table 5.2. The MLP input is fixed to one acoustic frame (no context). Further results and discussions regarding this issue can be found in (Petrovska and Hennebert 1998). A controlled experiment between a $C00$ and a $C11$ configuration in which the number of parameters is kept constant is reported. The idea is to evaluate if the improvement when using a larger MLP input is due to the larger number of parameters or from informations in the frame context. A $C11$ net with 20 hidden nodes (760 weights) is compared to a $C00$ net with 54 nodes (756 weights). The $C00-54$ configuration performed worse than the $C11-20$ but unfortunately, it seems that there is a lack of training data or a problem of local minimum in order to train the $C00-54$ nets since the performance is even worse than the $C00-20$ configuration. Increasing the number of hidden nodes can result in a more powerful and complex classification function but which is more subject to over-fitting or to "getting stuck" in a local minima. This confirms the difficulties to find an optimal configuration for the MLP.

Influence of the sampling mode

The world training set is, on average, 25 times larger than the client training set and training problems due to this large difference are suspected. As explained in section 5.2.2, an alternative sampling procedure referred as EFS is compared to the standard sampling procedure used for training. EER using the baseline (BL) and EFS sampling procedure are given in table 5.3 for the $C00$ and $C11$ configuration. For $C00$, the EFS procedure shows some improvements while for $C11$, there are no significant differences. A clear

	Nasal	Fricatives	Vowels	Plosives	Liquids
C00-BL	27.5	33.4	33.8	39.8	41.0
C00-EFS	23.5	30.9	32.9	38.6	39.2
C11-BL	9.7	14.4	16.2	20.8	22.4
C11-EFS	10.6	13.4	15.4	21.6	22.6

Table 5.3: EER averaged per phonemic group with two different training sampling modes of the MLP.

explanation for this behaviour is not straightforward. One could argue that the $C11$ configuration brings a better class separability which smooth out problems of unbalanced class populations.

5.2.4 Discussion

The speaker verification has been carried on a per-phoneme basis, in order to determine the more discriminative ones. According to the experiments, nasals, fricatives and vowels are found to provide the best performances, followed by plosives and liquids. The influence of the acoustic frame context size at the input of the MLP is studied and significant improvements are reported from the inclusion of several acoustic frames. Such large improvements were not reported for global approaches in other studies. It seems also that each phoneme has its optimal MLP input size minimizing the EER.

5.3 The segmental system

We propose here to use a **segmental** approach in which the speech signal is pre-classifyied into more precise units. On the one hand, the segmental approach recovers text-dependent advantages since the speech signal is automatically aligned into classes. On the other hand, the implementation is different since we have no clue about what has been said. As for text-dependent systems, we can underline two potential advantages. First, if the speech units are relevant, then speaker modeling is more precise and the system should present better performances than the global approach. Second, if speech units present different discriminative power, then intelligent recombination of individual decisions can be done.

The segmental system is depicted in Fig. 5.4. The vector sequence is first segmented and then labelled into a category. In our case, categories are determined with an automatic procedure. Segments are assigned to a model which is selected according to its class label. Finally, a score recombination of the individual models is performed.

The segmental approach requires an accurate recognition of speech segments. Two alternative procedures can be followed :

- Large Vocabulary Continuous Speech Recognition (LVCSR) that provides the hypothesised contents of the speech signal on which classic text-dependent techniques can be applied. LVCSR uses previously trained phone models and a language model, generally a bigram or trigram stochastic grammar.

- ALISP (Automatic Language Independent Speech Processing) tools (Chollet, Černocký, Constantinescu, Deligne, and Bimbot ress) that provide a general framework for creating sets of acoustically coherent units and of corresponding phone-like data transcriptions, with little or no supervision.

LVCSR systems, although very promising for segmental approaches, require large annotated data sets for training and are therefore arduous to work with. Furthermore, LVCSR are often dependent on the speech signal characteristics (language, speech quality, ...), making them less portable to new tasks. On the other hand, ALISP tools are very portable and don't require large training data sets.

That is what led us to consider a text-independent segmental approach based on ALISP tools. Among the available ALISP techniques, we have chosen the temporal decomposition (TD) and vector quantization (VQ) to segment and obtain classes of sounds.

We detail hereafter each component of the segmental system in Fig. 5.4.

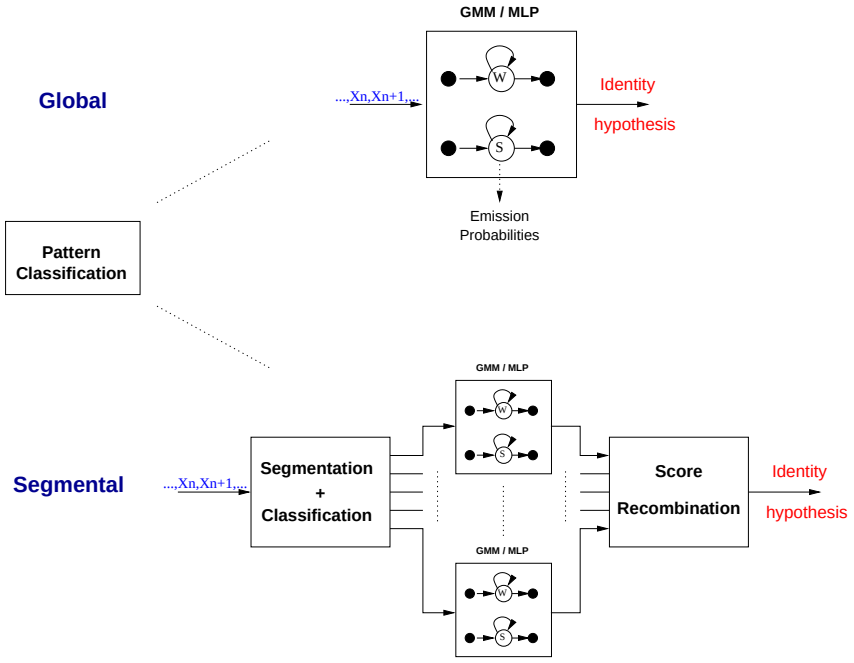


Figure 5.4: Global and segmental speaker verification system.

The temporal decomposition technique is used to compute speech segments, followed with a vector quantization (VQ) that classify them into categories. The modeling part of the system can be similar to the one of global systems. In our case, we used a set of multi-layer perceptrons (MLP) trained to discriminate between client and world speaker. Each MLP is dedicated to work with data segments that are previously selected as belonging to a particular class.

Segmentation

Segmentation and labelling in the segmental system are achieved respectively using temporal decomposition and Vector Quantization (VQ). The purpose is to discover quasi-stationary parts in parametric representations. This method, introduced by Atal (Atal 1983) and refined by Bimbot (Bimbot 1990), approximates the trajectory of i^{th} parameter x_n^i by a sum of m *targets* a_{ik} weighted by *interpolation functions*:

$$\hat{x}_n^i = \sum_{k=1}^{m} a_{ik}\phi_k(n), \qquad i = 1, \ldots, P \tag{5.14}$$

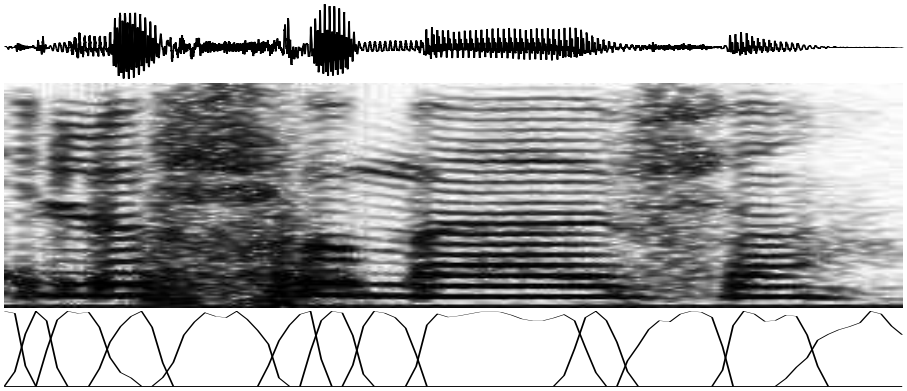
where P is the dimension of the parameter vectors. Equation 5.14 can be written in matrix notation:

$$\underset{(P \times N)}{\hat{\boldsymbol{X}}} = \underset{(P \times m)}{\boldsymbol{A}} \quad \underset{(m \times N)}{\boldsymbol{\Phi}}, \tag{5.15}$$

where the lower line indicates matrix dimensions. The initial interpolation functions are found using local *singular value decomposition* with adaptive windowing, followed by post-processing (smoothing, decorrelation and normalization). Target vectors are then computed by: $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{\Phi}^{\#}$, where $\boldsymbol{\Phi}^{\#}$ denotes the pseudo-inverse of interpolation functions matrix. Interpolation functions and targets are locally refined in iterations minimizing the distance of $\boldsymbol{X}$ and $\hat{\boldsymbol{X}}$. More details on the computation of $\boldsymbol{A}$ and $\boldsymbol{\Phi}$ can be found in (Bimbot 1990).

An example of temporal decomposition is given in Fig. 5.5 [7]. The sample sequence is depicted in

[7] Fig. 5.5 was kindly provided by Jan Černoký (Cernoký, Baudouin, and Chollet 98).



French word "le chômage"

Figure 5.5: Example of temporal decomposition (after Jan Cernocky). Upper part : sample sequence; middle part : spectrogram; bottom part : interpolation functions.

the upper part and the interpolation functions are given in the bottom part. Intersections of successive interpolation functions permit to compute stretches that will determine segments.

Labelling

The next step is *unsupervised clustering*. Among several available algorithms (Ergodic HMM, self-organizing map, etc.), *Vector Quantization* (VQ) was chosen for its simplicity. The VQ codebook $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_L\}$ is trained by a k-means algorithm with binary splitting (Gersho and Gray 1992) as described in section 3.1.2. Codebook training is performed using vectors located in gravity centers of segments computed with temporal decomposition in order to smooth out the effect of noisy vectors at the border of segments.

The quantization if performed on a per-segment basis. The accumulated distance between all vectors in segment and code-vectors is computed. The code-vector leading to the smallest accumulated distance is declared as winner :

$$y_b^s = \min_i \left[\sum_{x \in X_s} d(x, y_i) \right] \tag{5.16}$$

where s denotes a particular segment and b the label of the winner centroid. All vectors in segment s are labelled with b. Temporal decomposition and VQ provide a quasi-phonetic symbolic transcription of data in an unsupervised way. Each vector of the acoustic sequence are declared members of categories C_l determined through the segmentation and the labelling. The number of categories is fixed by the number of centroids in the VQ codebook.

MLP Modeling

As for global systems, the speaker modeling part can be based on any classical tool (GMM, VQ, ...). We have chosen to use MLP because their discriminant property makes them potentially more powerful than non-discriminant likelihood based tools.

The same technique as for global modeling (see section 5.1.1) is applied, but this time, L MLP (L is equal to the number of centroids in the labelling codebook) are used. They are respectively fed with feature vectors having corresponding labels. For example, the MLP associated with category C_l provides a segmental score as follows :

$$S_{cl} = \prod_{x \in C_l} p(M_{cl}|x)/P(M_{cl}) \tag{5.17}$$

$$S_{wl} = \prod_{x \in C_l} p(M_{wl}|x)/P(M_{wl}) \tag{5.18}$$

where products involve vectors previously labelled as member of category C_l with l the index of the code-vector computed as in Eq. 5.16. Subscripts cl and wl denote respectively the client and word model for category C_l. In Eq. 5.17 and 5.18, posterior probabilities are also divided by priors computed in a similar manner according to Eq. 5.5.

Score recombination

The last issue of the segmental approach is the recombination of individual scores. As it will be shown in the experiments, there are units more discriminant that other and therefore, we believe that the recombination should take this information into account. Nevertheless, we left this issue apart in our analysis and used a simple linear recombination of scores to obtain the global decision. The MLP scores are recombined through a simple addition :

$$\log S_c = \sum_{l=1}^{L} \log S_{cl}$$

$$\log S_w = \sum_{l=1}^{L} \log S_{wl} \tag{5.19}$$

Several other techniques for score recombination could be investigated, as for example, weighted recombination with a weighting proportional to the verification performances, or even a supervised non-linear recombination using a further ANN.

5.4 Experiences and results

5.4.1 Task description

Both segmental and global systems were compared in the framework of the NIST-NSA'98 evaluation campaign[8]. The data are selected from the Switchboard database, recorded over telephone lines. The speech is spontaneous and no transcriptions, neither orthographic nor phonetic, are available. The *training set* consists of 250 male and 250 female subjects representing *clients* of the system. The sex mismatch is not studied in these evaluations, so that all experiences are strictly sex-dependent.

A more detailled description of the data sets is given in appendix A.4. For each client, the system is trained under three training conditions depending on the amount of data and denoted 1S for one session, 2S for two sessions and 2F for two sessions-full. The *test set* comprises 2500 test files per sex, and per test condition, each "pretending" to be 10 clients. Three test conditions are defined depending on the available amount of test data: 3, 10 or 30 seconds. The total number of trials to do within NIST evaluations is: 2 sexes $\times$ 9 train–test conditions $\times$ 2500 test files $\times$ 10 trials per test file = 450000. For the evaluation of results, the train–test file couples are separated into following categories: SN (training and test file from the same telephone number), ST (different number but the same handset type), DT (different number and different handset type).

[8]The National Institute of Standards and Technology - National Security Agency organizes every year an evaluation of speaker verification systems. A unique data set and evaluation protocol are provided to each participating laboratory, so that intra- and inter-laboratory algorithms comparisons are significantly easier.

5.4.2 Global system results

GMM

The following experimental setups are used :

- Feature analysis: Vectors of 25 coefficients are computed every 10 ms on an analysis frame of 30 ms. After pre-emphasis and hamming windowing, 12 lpc-cep coefficients are computed from 10th order lpc analysis. Cepstral mean substraction is performed for channel equalization. Vectors are completed with the delta cepstrum and the delta log energy.
- GMM acoustic modeling: Target speakers are modeled with 64 mixtures. A unique world model with 64 mixtures is trained on 100 male and 100 female speakers (mixed handset) taken from the 1997 evaluation data set. Training is performed using max likelihood.
- Znorm normalization: The sex-dependent normalization of the likelihood ratios is based on a large number of impostor access (no handset normalization).
- CPU execution: We used SGI Origin2000, 32 CPU's MIPS R10000 195 MHz, 390 MFLOPS per CPU, 192 Megabytes/cpus.
- Test execution time: One time real-time on a single CPU.

DET curves for the training condition 2F are reported in Fig. 5.6, top part. As expected, better results are obtained when longer test segments are used and when the testing conditions are similar to the training conditions. The best performances are obtained when the client tries to log in using the same phone type and line as for training, i.e. when there is no mismatch between training and testing. DET curves for the 10 seconds test lengths are reported in Fig. 5.6, bottom part. Better performances are obtained when training is performed in two sessions, but the improvement is not so significant in case of mismatched conditions.

A general conclusion we can draw from these results is that the difficulties come, for the most part, from the mismatches that exists between training and testing conditions. The models seem to focus equivalently on the channel characteristics as on the speaker features.

MLP

MLP with 120 neurons in the hidden layer and two outputs are trained to discriminate between the target speaker and 100 speakers of the same sex (mixed handsets, 1997 data). A classic back-propagation algorithm is used to train the MLPs with a learning rate evolution based on a performance measure on a cross-validation data set. The cross-validation set is obtained cutting off about 20 % of the training data.

The experimental setup is very similar to the one of GMM except for the acoustic features that are here limited to the 12 lpc-cepstrum coefficients. The log-energy and the delta are not included. DET curves for training condition 2F and test duration 30 sec are given in Fig. 5.7. The idea was to investigate the influence of the context at the input of the MLP. Increasing the context improved performances in all cases, however a saturation is appearing when 11 frames are used in input (C55 configuration). A good cpu cost-verification performance trade-off is around 7 frames of context (C33 configuration). When compared to the GMM system, the MLP seems to present inferior performances. The explanation may come from the cross-validation technique that is needed to avoid overfitting. Cross-validation requires to put apart training data that are not used to train the parameters. As we have already few data to train the system, this step surely hurts the modeling.

Also, we re-did similar experiments including the delta and energy information as for the GMM system, but with no or few success.

5.4.3 Segmental system results

For the segmental system, the following setups are used :

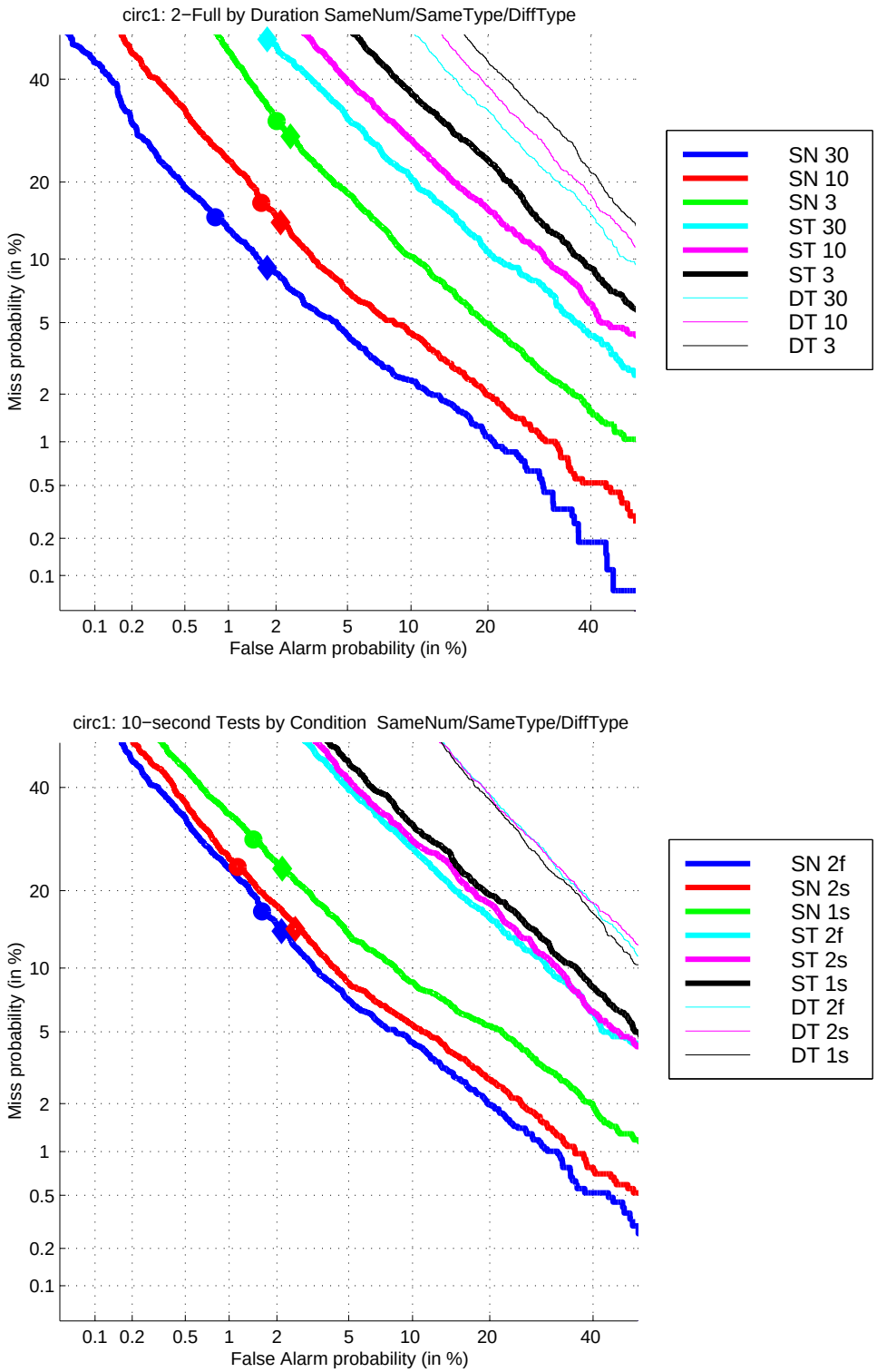


Figure 5.6: GMM system. Upper figure: results for training condition 2F with different testing conditions. Bottom figure: results for 10 sec test length using different training conditions. SN stands for training and test file from the same telephone number, ST stands for different number but the same handset type, DT stands for different number and different handset type.

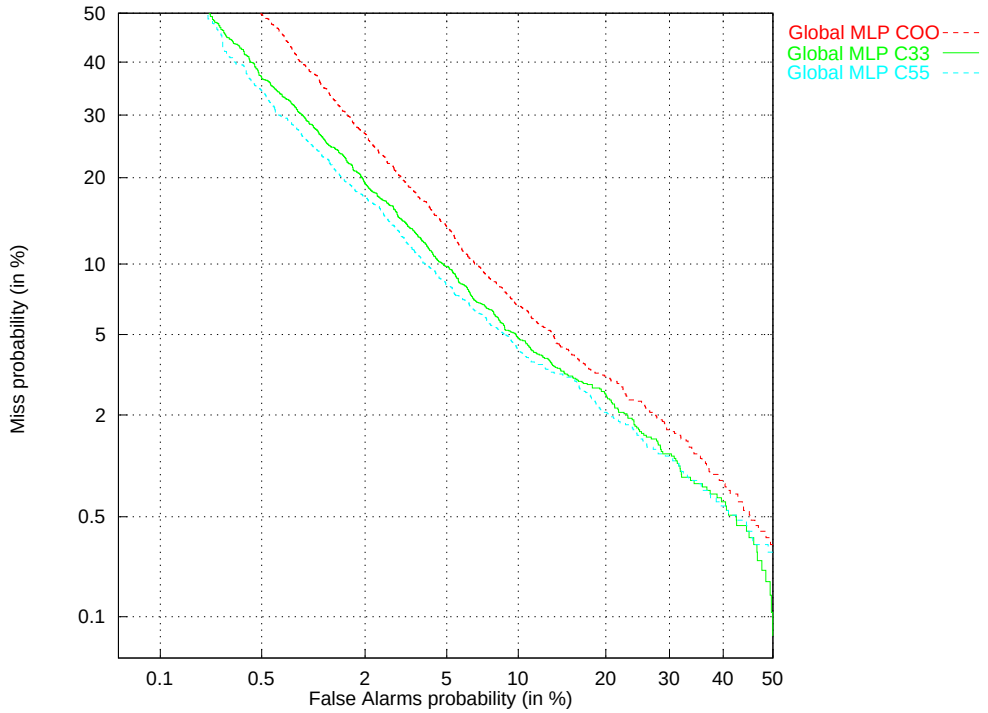


Figure 5.7: Segmental system, results by classes, training condition 2F, test duration 30 sec, SN test condition (training and test file from the same telephone number).

- Feature analysis: Vectors of 12 coefficients are computed every 10 ms on an analysis frame of 30 ms. After pre-emphasis, 12 lpc-cep coefficients are computed from a 10th order lpc analysis. Cepstral mean substraction is performed for channel equalization. Concatenation of 5 adjacent vectors (C22 context) is used as input of the MLP.

- Segmentation: The automatic segmentation of the speech signal is obtained with the temporal decomposition. TD was set up to detect 15 acoustic events per second in average, which is a usual setup for 10 ms cepstral analysis vectors on telephone speech.

- Labelling: Each segment is categorized into 8 classes with VQ (VQ codebooks are trained on 1997 data). The same segmentation procedure is applied to the training and testing material. The number of classes was kept small in order to model somehow broad phonetic classes and to keep the amount of training data large enough to train the MLP's.

- MLP modeling: Eight MLP's are used per target speaker (one MLP per segment label). Each MLP is trained to discriminate between the target speaker and 100 speakers of the same sex (mixed handsets, 1997 data). A classic back-propagation algorithm is used to train the MLPs with a learning rate evolution based on a performance measure on a cross-validation data set. Five consecutive feature vectors are used as input of the MLPs and 20 neurons compose the hidden layer. The LMS error measure is used for the backpropagation.

- Znorm normalization: The znorm is performed in a similar manner as for the GMM system except that it is applied before the recombination of individual class scores.

- CPU execution: We used the SGI Origin2000, 32 CPU's MIPS R10000 195 MHz, 390 MFLOPS per CPU, 192 Megabytes/cpus.

- Test execution time on a single CPU: two times real-time. The temporal decomposition and VQ takes more than one time real-time, while the actual score computation is faster than with GMM.

Parameters of the temporal decomposition and of the VQ labelling were experimentally tuned to reach acceptable performances. Unfortunately (and this is a drawback of the method), there are no ways to avoid this experimental parameter settings.

In a first step, speaker verification performances can be analysed for each classes. The DET curves are depicted in Fig. 5.9 for the training condition 2F and test duration 30 sec. The five more dissimilar classes were selected for sake of clarity. One should point out the large difference of performances between classes, ranging from 9 to 20 % EER. The curve in the upper part of Fig. 5.9 corresponds to the MLP trained on class 0 and fed during testing with with frames sampled randomly in the available test material. The poor performances of this test ensure us that the segmentation and labelling procedures are consistent. Observing a better score would have been a sign that the TD and VQ labelling give unreliable segments and labels. Coherence of acoustic labelling among speakers has also been verified in informal listening tests.

The difference of performances of the individual classes suggests that a score weighting should be performed to compute the global score. A further insight can be obtained in visualizing the normalized client and impostor score distribution, as depicted in Fig. 5.8 for classes 6 and 7. The histogram are obtained in accumulating the log-likelihood scores for the impostor and the client models. The more the distributions are spread apart, the better is the class performance. In Fig. 5.8, impostor scores are distributed around zero since znorm was applied. The intersection of the impostor and client score distribution defines an area proportional to the EER.

In a first step, we did a simple linear recombination of each class score to obtain global decisions. DET curves for matched (SN) and mismatched (DT) conditions are presented in Fig. 5.10 where our best global MLP system is compared to the segmental system. In the case of matched conditions, the segmental approach reaches almost equivalent performances as the global system, while it performs better in the case of mismatched conditions. This result is validating the whole segmental approach. Also, when compared to a state of the art GMM system (see Fig. 5.6), the segmental approach performs slightly better in the case of mismatched conditions.

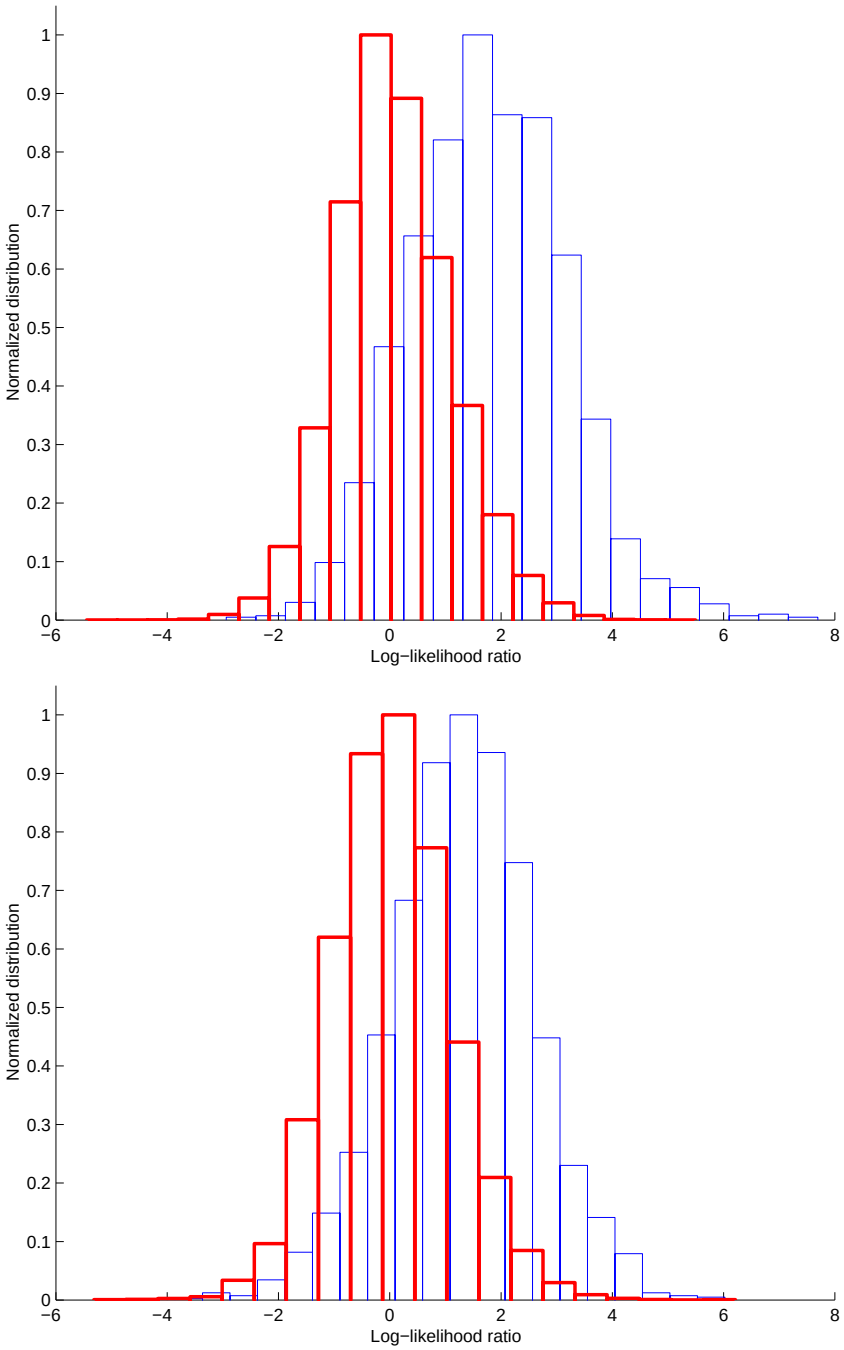


Figure 5.8: Segmental system, log-likelihood score distribution for, respectively, class 6 and 7 for the female speakers. Training condition 2F and testing duration 30 sec.

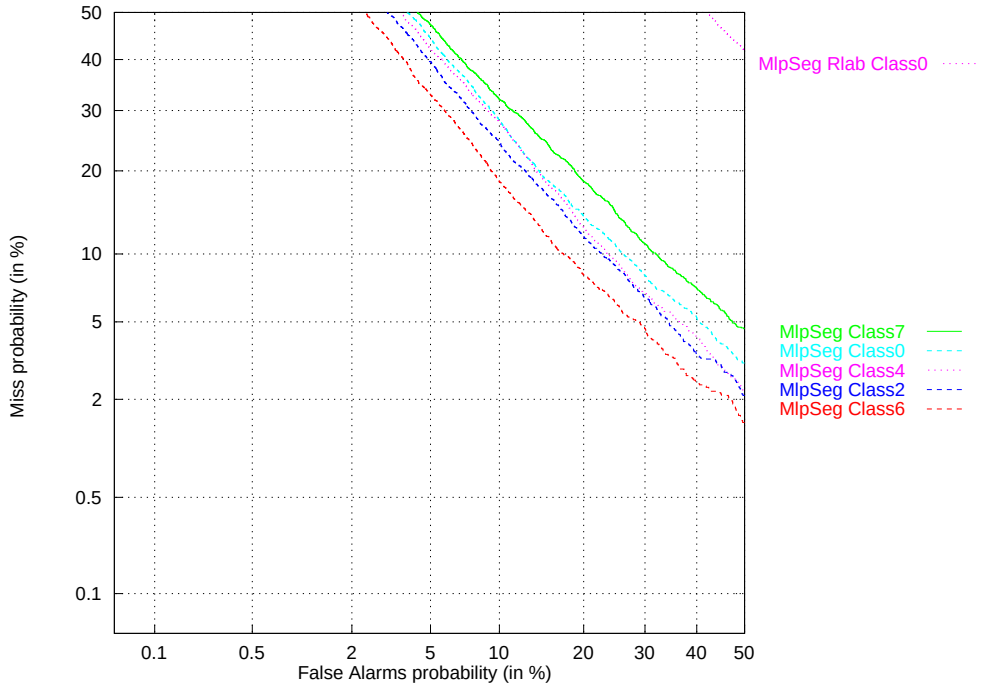


Figure 5.9: Segmental system, results by class, training condition 2F, test duration 30 sec, "same number" (SN) condition for testing.

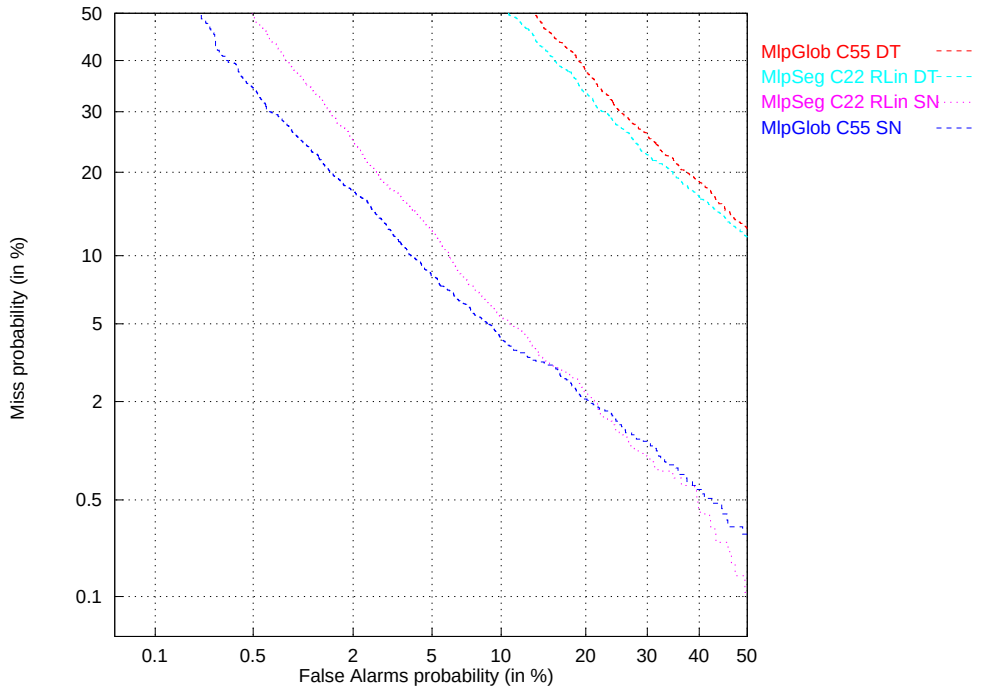


Figure 5.10: Global and segmental system, training condition 2F, test duration 30 sec, same number (SN) and different type.

5.5 Conclusions

Usually, text-independent speaker verification is based on modeling the **global** probability distribution function (pdf) of speakers in acoustic vector space, for example by using Gaussian mixture modeling. We believe that such global approaches are reaching their limits because speaker modeling is somehow too coarse in this case. In this chapter, we have proposed a new **segmental** approach for text-independent speaker verification.

The general idea of the segmental approach is to recover the advantages of text-dependent techniques through an automatic segmentation of the speech material into sub-word units. Speakers are then modeled on these units that are more precise in the acoustic vector space and less sensitive to variabilities. The segmental approach, if an efficient alignment tool is used, should present better robustness in the case of mismatched conditions between training and testing. We also proposed to use discriminant ANN tools to perform more efficiently speaker verification on the individual segments.

5.5.1 Text-dependent preliminary experiments

In a first step, we did preliminary experiments with a text-dependent task in order to validate the potentialities of the approach and to examine the issues. We have built a system based on HMMs and MLPs. HMMs are used to provide the automatic segmentation of the speech signal into phone-like units and the MLPs are used to model speakers on a phone basis. In a similar manner as for hybrid speech recognition systems that were studied in the previous chapter, the system combines here the ability of HMMs to handle efficiently the sequential character of speech and the discriminant properties of MLPs.

An analysis of the speaker verification performances is carried on a telephone database. EER are computed on a per-phone basis in order to determine the more discriminative ones. Two major conclusions can be drawn from the experiments. First, some segments corresponding to nasals, fricatives and vowels are found to provide the best performances, followed by plosives and liquids. Secondly, significant improvements are reported from the inclusion of several acoustic frames at the input of the MLP, with different optimal input size for each phone. This last point suggests that the configuration of the system's parameters can be tuned to optimize individual phone decision.

5.5.2 Text-independent segmental system

We underlined three new issues for text-independent systems based on the segmental approach .

First, the speech signal has to be segmented into sub-word classes. A promising technique would be to do large vocabulary continuous speech recognition, but this requires large annotated data sets for training and, consequently, is arduous to work with. Therefore we opted for an ALISP unsupervised technique based on temporal decomposition and vector quantization. This approach offers the advantage of beeing completely unsupervised and therefore portable to other languages and tasks. On the other hand, a drawback of these ALISP techniques lies in the fact that several parameters of the temporal decomposition and vector quantization needs to be exprimentaly tuned to reach acceptable performances.

Secondly, speaker models have to be trained and tuned in order to minimize the EER on a per-phone basis. We opted here for MLPs that present the advantages to be discriminant and to relax hypothesis on the form of the acoustic feature distribution. They are also able to take into account large window spans in input.

Thirdly, a score recombination of the independent classifiers is necessary to obtain the global speaker verification decision. If, as reported in the preliminary experiments, there are sub-word units more discriminant that other, the recombination should take this information into account. We left this issue apart in our analysis and used a simple linear recombination of scores to obtain the global decision.

A set of experiments have been carried on the text-independent NIST 1998 evaluation data. Three systems are compared. Two **global** systems (GMM and ANN) and the **segmental** system, as described before. Three major conclusions can be drawn from the results:

1. The global GMM and MLP systems perform similarly, with a slight advantage for GMM. The general conclusion we can draw from these systems is that the difficulties come, for the most part,

from the mismatches that exist between training and testing conditions. The models seem to focus equivalently on the channel characteristics as on the speaker specific features.

2. Although a simple linear recombination of individual scores has been used, the segmental approach reaches almost equivalent performances as the global systems in the case of matched conditions.

3. The segmental approach performs better than the two global system in the case of mismatched conditions.

Many issues are still open with the segmental approach as proposed in this chapter. Nevertheless, even using simple strategies for producing the alignment and for recombining individual scores, very good results are already reported on a standard evaluation task.

Bibliography

Artières, T. (1995). *Méthodes prédictives neuronales: application à l' identification du locuteur*. Ph. D. thesis, Université de Paris XI Orsay.

Atal, B. S. (1983). Efficient coding of lpc parameters by temporal decomposition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 81–84.

Bennani, Y. and P. Gallinari (1994, April). Connectionist approaches for automatic speaker recognition. In *ESCA Workshop on Automatic Speaker Recognition, Identification and Verification*, Martigny, Switzerland, pp. 95–102.

Besacier, L. and J.-F. Bonastre (1997, August). Independent processing and recombination of partial frequency bands for automatic speaker recognition. In *Fourteenth Internationnal Conference on Speech Processing*, Seoul, Korea. IEEE Korea Council, IEEE Korea Signal Processing Society.

Besacier, L. and J.-F. Bonastre (1998). Frame pruning for speaker recognition. In *To appear in ICASSP*, Seattle.

Bimbot, F. (1990). An evaluation of temporal decomposition. Technical report, Acoustic research departement AT&T Bell Labs.

Bimbot, F. and G. Chollet (1995). *EAGLES Handbook on Spoken Language Resources and Assessment*, Chapter Assessment of speaker verification systems. Mouton de Gruyter.

Bourlard, H. and N. Morgan (1994). *Connectionist Speech Recognition*. Kluwer Academic Publishers.

Bourlard, H. and C. J. Wellekens (1990, December). Links between markov models and multi-layer perceptrons. *IEEE Trans. Patt. Anal. Machine Intell. 12*(12), 1167–1178.

Cernoký, J., G. Baudouin, and G. Chollet (98, may). Segmental vocoder-going beyond the phonetic approach. In *Submitted to IEEE ICASSP98*, Seattle, WA, US.

Chollet, G., J. Černocký, A. Constantinescu, S. Deligne, and F. Bimbot (in press). *NATO Advanced Study Institute: Computational models of speech pattern processing*, Chapter Towards ALISP: a proposal for Automatic Language Independent Speech Processing. Springer Verlag.

Eatock, J. P. and J. S. Mason (1994). A quantitative assessment of the relative speaker discriminant properties of phonemes. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Volume 1, pp. 133–136.

Egan, J. (1975). *Signal detection theory and ROC analysis*. Academic Press.

Farrell, K. A., R. Mammone, and K. Assaleh (1994). Speaker recognition using neural networks and conventional classifiers. *IEEE Transactions on Speech and Audio Processing 2*(1), 194–205.

Gersho, A. and R. Gray (1992). *Vector Quantization and Signal Compression*. Kluwer Academic Publishers.

Gravier, G. and G. Chollet (1998). Comparison of normalization techniques for speaker verification. In *Speaker Recognition and its Commercial and Forensic Applications, RLA2C*, Avignon, France, pp. 97–100.

Hattori, H. (1994, April). Text-independent speaker verification using neural networks. In *Workshop on Automatic Speaker Recognition and Verification*, Martigny, Switzerland, pp. 103–106.

Hennebert, J. and D. Petrovska (1998). Phoneme based text-prompted speaker verification with multi-layer perceptrons. In *RLA2C 98*, Avignon, France, pp. 55–58.

Homayounpour, M. and G. Chollet (1994, April). A comparison of some relevant parametric representations for speaker verification. In *ESCA workshop on automatic speaker recognition, identification and verification*, Martigny, Switzerland, pp. 185–192.

Magrin-Chagnolleau, I., J. Wilke, and F. Bimbot (1996, May). A further investigation on ar-vector models for text-independent speaker identification. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Atlanta, GA, pp. 401–404.

Martin, A., G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki (1997). The det curve in assesment of detection task performance. In *Eurospeech 1997*, Rhodes, Greece, pp. 1895–1898.

Matsui, R. and S. Furui (1993, April). Concatenated phoneme models for text-variable speaker recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Volume 2, Minneapolis, pp. 391–394.

Montacie, C., P. Deleglise, F. Bimbot, and M. J. Caraty (1992). Cinematic techniques for speech processing : temporal decomposition and multivariate linear prediction. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Volume 1, San-Francisco, pp. 153–156.

Naik, J. M. and D. M. Lubenskt (1994). A hybrid hmm-mlp speaker verification algorithm for telephone speech. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 153–156.

Nolan, F. (1983). *The Phonetic Bases of Speaker Recognition*. Cambridge University Press.

Oglesby, J. and J. S. Mason (1990). Optimization of neural models for speaker identification. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 261–264.

Olsen, J. (1997). A two-stage procedure for phone based speaker verification. In G. B. J. Bigün, G. Chollet (Ed.), *First International Conference on Audio and Video Based Biometric Person Authentication (AVBPA)*, Crans, Switzerland, pp. 219–226. Springer Verlag: Lecture Notes in computer Science 1206.

Petrovska, D. and J. Hennebert (1998). Text-prompted speaker verification experiments with phoneme specific mlp's. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Seattle, pp. 777–780.

Picone, J. (1993, September). Signal modeling techniques in speech recognition. *Proceedings of the IEEE 81*(9), 1214–1247.

Rabiner, L. and B.-H. Juang (1993). *Fundamentals of Speech Recognition*. Prentice Hall.

Reynolds, D. (1995). Automatic speaker recognition using gaussian mixture speaker models. *The Lincoln Laboratory Journal 8*(2), 173–191.

Reynolds, D. (1997). Comparison of background normalization methods for text-independent speaker verification. In *Eurospeech*, pp. 1895–1898.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland (Eds.), *Parallel Distributed Processing. Exploration in the Microstructure of Cognition*, Volume 1. Massachusetts Institute of Technology Press.

Chapter 6

Conclusions

An expert is a man who has made all the mistakes, which can be made, in a very narrow field.

Niels Bohr

In this thesis, we were concerned with the two fields of automatic speech recognition and automatic speaker recognition in telephony. More precisely, we have analysed HMM systems in which ANNs are used in place of more classical tools.

We have investigated three systems. The first one, mostly original, concerned the use of Self-Organizing Maps in discrete HMMs for isolated word speech recognition. The second system concerned continuous hybrid HMM/ANN systems, extensively studied in previous research work. Its analysis permitted to bring a new theoretical framework and to introduce some extensions regarding the way the system is trained. The last system concerned the implementation of an ANN segmental approach for text-independent speaker verification. This last study is original and has been developed through intensive collaborations with two colleagues.

More detailed conclusions on each systems are given hereafter.

6.1 Self-Organizing Maps and Discrete HMMs

We have proposed an original and successful utilisation of Self-Organizing Map (SOM) of Kohonen in the framework of discrete HMM used for speech recognition. The vector quantizer, instead of being designed with a classical minimum distortion algorithm (k-means), is designed with an unsupervised self-organizing neural network. Although conceptually weaker than continuous supervised systems like hybrid systems, the unsupervised approach presented here offers performance and learning speedup advantages when compared to other discrete systems.

We compared SOM with two baseline algorithms, namely LBG and k-means. A theoretical analysis of the algorithms gave us a better understanding of their differences. SOM tend to place centroids to form a discrete image of the input space, preserving its distribution. This behaviour is in favor of the training of discrete HMM systems that need good resolution of the feature distribution in highly

populated regions. K-means and LBG, on the other hand, provides an explicit minimization of the local average distorsion but does not guarantee that centroids are globally located to mimic the input space distribution. We compared performances of these algorithms with recognition experiments using the same system on two different telephone speech databases. Analysing the experimental results, SOM produced codebooks which lead to consistently better recognition rates than when using k-means or LBG. Further improvements were also obtained when SOM codebooks are used as a starting point for k-means refinement.

We also went further in the ANN direction, trying to build codebooks in a supervised way using the Learning Vector Quantization (LVQ) algorithm. Training targets are obtained with a forced Viterbi alignment. After training the codebook with LVQ, vector quantization is performed as usual before feeding discrete HMMs. Frame-based classification is better but no improvement is reported at the word level. This is somehow normal since the discrete HMMs need a good resolution of the input space distribution, which is, in a similar manner as for k-means, not guaranteed by positioning centroids so that frame classification errors are minimized.

6.2 Feedforward ANNs and Continuous HMMs

We have analysed hybrid HMM/ANN systems in which the local frame hypothesis are estimated with MLP. Previous studies have already been dedicated to hybrid systems (Bourlard and Morgan 1993; Bourlard and Morgan 1994; Renals, Morgan, Bourlard, M.Cohen, and Franco 1994). The work presented here can be viewed as a extension or a continuation of these studies.

Starting from a description of the original hybrid HMM/ANN system, we have proposed a new theoretical formalism to describe the behaviour of such systems. The theoretical perspective provides us with a better unified view of hybrid systems and their relationships to standard HMMs. In short, the theory shows that such systems can actually be used to compute global model posteriors, provided that the ANN part estimates local posteriors. The classic Viterbi trained hybrid system and the more recent REMAP system are clearly unified with the theory.

This better formalism inspired us to derive a new forward-backward training for hybrid systems. It is shown how to implement the training with the computation of new ANN targets. It is also explained how to compute state duration and priors with the forward-backward scheme.

Several sets of experiments have been carried on isolated word telephone databases. Firstly, we have compared Gaussian mixtures and hybrid systems on the Her database reporting on memory bandwidth, CPU cost and error rates. Results show that, for similar amount of parameters, hybrid systems perform systematically better than Gaussian mixture systems, both in terms of error rates and CPU speed/memory cost. These experiments are confirming the conclusions of previous studies (see e.g. (Bourlard and Morgan 1993)). We cross-validated these experiments on a more difficult task using the Phonebook database. Again, hybrid systems are performing better than Gaussian mixtures and the benefits are even more obvious on this database.

Second, we have performed experiments to compare Viterbi and forward-backward training of hybrid systems. Recognition results obtained on Phonebook show a clear advantage for forward-backward training when a small training set is used. For larger training sets, Viterbi seems to give equivalent or better recognition rates. We have also presented a graphical analysis of the ANN targets in order to better understand the behaviour of both algorithms. Targets obtained with forward-backward are very similar to the one obtained with Viterbi, except at phone boundaries where the forward-backward criterion does not take clear-cut decision regarding the state membership. Forward-backward makes much more sense than Viterbi because the vocal tract does not move instantly from one position to the other. Viterbi is less realistic when it takes hard state membership decision for these features falling at the phone boundaries. Nevertheless, the amount of vectors in this case is quite small and the hard decision effect of Viterbi should be smoothed out when large database are used. The explanations are indeed confirmed through the experiments carried on the Phonebook database.

6.3 Segmental approach for speaker verification

Usually, text-independent speaker verification is based on modeling the **global** probability distribution function (pdf) of speakers in acoustic vector space, for example by using Gaussian mixture modeling. We believe that such global approaches are reaching their limits because speaker modeling is somehow too coarse in this case. Therefore, we have proposed a new **segmental** approach for text-independent speaker verification.

The general idea of the segmental approach is to recover the advantages of text-dependent techniques through an automatic segmentation of the speech material into sub-word units. Speakers are then modeled on these units that are more precise in the acoustic vector space and less sensitive to variabilities. The segmental approach, if an efficient alignment tool is used, should present more robustness in the case of mismatched conditions between training and testing. We also proposed to use discriminant ANN tools to perform more efficiently speaker verification on the individual segments. We underlined three issues for text-independent systems based on the segmental approach.

First, the speech signal has to be segmented into sub-word classes. A promising technique would be to do large vocabulary continuous speech recognition but this requires large annotated data sets for training and, consequently, is harder to work with. Therefore we opted for an ALISP unsupervised technique based on temporal decomposition and vector quantization. This approach gives the advantage of being completely unsupervised and therefore portable to other languages and tasks. On the other hand, a drawback of these ALISP techniques lies in the fact that several parameters of the temporal decomposition and vector quantization needs to be exprimentaly tuned to reach acceptable performances.

Secondly, speaker models have to be trained and tuned in order to minimize the EER on a per-phone basis. We opted here for MLPs that present the advantages to be discriminant and to relax hypothesis on the form of the acoustic feature distribution. They are also able to take into account large window spans in input.

Thirdly, a score recombination of the independent classifiers is necessary to obtain the global speaker verification decision. If, as reported in the preliminary experiments, there are sub-word units more discriminant that other, the recombination should take this information into account. We left this issue apart in our analysis and used a simple linear recombination of scores to obtain the global decision.

A set of experiments have been carried on the text-independent NIST 1998 evaluation data. Three systems are compared. Two **global** systems (GMM and ANN) and the **segmental** system, as described before. Three major conclusions can be drawn from the results:

1. The global GMM and MLP systems perform similarly, with a slight advantage for GMM. The general conclusion we can draw from these systems is that the difficulties come, for the most part, from the mismatches that exist between training and testing conditions. The models seem to focus equivalently on the channel characteristics as on the speaker specific features.

2. Although a simple linear recombination of individual scores has been used, the segmental approach reaches almost equivalent performances as the global systems in the case of matched conditions.

3. The segmental approach performs better than the two global system in the case of mismatched conditions.

Many issues are still open with the segmental approach as proposed in this work. Nevertheless, even using simple strategies for performing the alignment and for recombining individual scores, very good results are already reported on a standard evaluation task.

Appendices

Appendix A

Databases

The different databases used in this work are detailled in this appendix.

A.1 Phonebook

PhoneBook is a recent database collected by NYNEX (Pirelli et al. 1995). It has been conceived for the development of large vocabulary, isolated words applications[1]. The core section of PhoneBook consists of a total of 93,667 isolated-word utterances, totalling 23 hours of speech with nearly 8000 distinct words. Each word was read by 7 to 13 speakers, with a total of 1358 speakers each saying up to 75 words. All data were collected in 8-bit mu-law digital form directly from a T1 telephone line. Talkers were adult native speakers of American English chosen to be demographically representative of the U.S.

The goal of PhoneBook is to serve as a large database of American English word utterances incorporating all phonemes in as many segmental/stress contexts as are likely to produce coarticulatory variations, while also spanning a variety of talkers and telephone transmission characteristics.

There are 106 word lists used in the corpus, and these are identified by two-character sequential alphabetic codes: "aa", "ab", "ac", ... "eb".

A.1.1 Common features to all the experiments

We tried to keep the following features constant throughout all the experiments. Basically, all these features have been taken from previous experiments carried out by the team in Mons.

Set	Lists	nLists	nSpeak	nWords	nUtt	nFrames
train	*a *h *m *q *t	21	283	1,581	19,421	2,761,063
cross	*o *y	8	106	603	7,291	1,054,954
test	*d *r	8	96	602	6,598	955,554
total	*a *h *m *q *t *o *y *d *r	37	485	2786	33310	

Table A.1: Some general statistics about the word lists in the training, cross and test set. The number of frames has been computed with a frame length of 200 samples and a frame shift of 80 samples (frame overlap of 120 samples)

Training Set

A 'small' training set (regarding the size of the database) totalling approximately 5 hours of speech is used: *a, *h, *m, *q and *t files (21 files). See table A.1 for general statistics about wordlists. Each file

[1] spontaneous numerical utterances are also available in the corpus

is associated with a word list of 75 words uttered by up to 8 talkers. We have to underline the fact that the speakers are not shared between different word list, i.e. a speaker has been asked to utter the words and only the words belonging to a particular wordlist. This fact should give strong advantage to blind deconvolution techniques like cepstral mean substraction. There are 283 speakers in the training and set. See table A.2 for some statistics about speakers/files and genders.

File label	Female	Male	total
aa	8	6	14
ah	8	5	13
am	8	6	14
aq	8	6	14
at	7	7	14
ba	9	5	14
bh	5	7	12
bm	7	9	16
bq	6	5	11
bt	6	5	11
ca	8	8	16
ch	9	6	15
cm	8	5	13
cq	7	7	14
ct	6	6	12
da	9	7	16
dh	9	6	15
dm	9	5	14
dq	8	4	12
dt	5	6	11
ea	7	5	12
total	157	126	283
total(%)	55 %	45 %	

Table A.2: Some statistics about genders and number of speakers in each word list of the training set.

Cross-Validation Set

All *o and *y files (8 files). Each file is associated with a word list of 75 words uttered by up to 8 talkers. See table A.1 for general statistics about wordlists.See table A.3 for some statistics about speakers/files and genders.

Test Set

All *d and *r files (8 files). Each file is associated with a word list of 75 words uttered by up to 8 talkers. See table A.1 for general statistics about wordlists.See table A.4 for some statistics about speakers/files and genders.

Since the lexicon is different in each of these 8 files, we then have the choice of recognizing the 8 files as a whole (yelding a lexicon of 600 words) or recognizing each file independently (with a lexicon of about 75 words). In the second case, the recognition rate will be the (unweighted) average of the 8 recognition rates.

Lexicon

The lexicon is derived almost totally from the CMU(0.4) lexicon. Some of the words were not present in the CMU dictionary and have been transcribed manually.

File label	female	male	total
ao	9	5	14
ay	7	7	14
bo	6	5	11
by	7	6	13
co	8	5	13
cy	8	8	16
do	8	4	12
dy	7	6	13
total	60	46	106
total(%)	57 %	43 %	

Table A.3: Some statistics about genders and number of speakers in each word list of the cross validation set.

File label	female	male	total
ad	6	5	11
ar	7	3	10
bd	6	5	11
br	8	7	15
cd	9	6	15
cr	7	4	11
dd	6	4	10
dr	8	5	13
total	57	39	96
total(%)	59 %	41 %	

Table A.4: Some statistics about genders and number of speakers in each word list of the test set

A.2 HER

The HER database has been recorded in the framework of the *Himarnnet Esprit Project*. This german spoken speech database contains 108 phonetically balanced isolated words uttered by 536 talkers. It has been recorded over telephone lines. A training set of 400 randomly choosen talkers is defined while 136 talkers are kept apart to test the systems. The list of context independent phonemes is given in table A.5.

A.3 HIM

The HIM database is a small isolated word speech database, recorded over telephone lines. The language is American English. It contains 53 words pronounced by a total of 226 speakers. A training set of 190 speakers is defined to train the HMMs and 36 speakers are used as test set.

The database has been transcribed in terms of context independent phoneme (CIP) HMMs. 41 CIP plus a model for the silence is used for the transcription.

A.4 NIST'1998 : Switchboard

The speaker verification NIST-NSA'98 evaluation campaign used data selected from the Switchboard database. The speech signal is recorded over telephone lines. The speech is spontaneous (natural conversation between two peers) and no transcriptions, neither orthographic nor phonetic, are available.

Vowels/diphthongs	Example	Consonants	Example
a	Hahn	p	Punkt
A	Tanne	b	Berg
e	Kehle	t	Tinte
E	Fell	d	Dich
i	Riese	k	Kiel
I	Kind	g	Gans
o	Rose	f	Feld
O	Koch	v	Wach
u	Buch	s	Fels
U	Kunst	S	Schnee
ű	Bűhne	z	Saal
Ű	Kűste	Z	Journal
ő	Rőmer	ç	Milch
ˆ	Kőpfe	x	Bach
$	Belag	l	Licht
@	Baum	r	Reise
Y	Heute	R	Ferne
J	Reiter	j	Jade
		h	Hand
		m	Man
		n	Norden
		N	Ring
		?	The?ater
sil			

Table A.5: Context independent phoneme list for the HER database.

A.4.1 Training set

The *training set* consists of 250 male and 250 female subjects representing *clients* of the system (the sex mismatch is not studied in these evaluations, so that all experiences are strictly sex-dependent).

Three training conditions are defined. Two of these conditions use 2 minutes of training speech data from the target speaker, while the other training condition use more than 2 minutes of training speech data. The conditions are:

1. "**One-session**" training. The training data is 2 minutes of speech data taken from only one conversation.
2. "**Two-session**" training. Equal amounts of training data are taken from two different conversations collected from the same phone number. The same phone number is used for the two sessions.
3. "**Two-session-full**" training. All available speech data taken from two different conversations collected from the same phone number.

The actual duration of the training files used for the 1-session and 2-session training conditions varies from the nominal value of 1 minute, so that whole turns may be included whenever possible. Actual durations for these training conditions are constrained to be within the range of 55-65 seconds. The duration of the training files for the two-session-full condition are constrained and are variable over speakers.

A.4.2 Test set

Performance are computed and evaluated separately for female and male target speakers and for the 3 training conditions. For each of these training conditions, there are 3 different test conditions of interest. These are:

1. Test segment duration. Performance will be computed separately for 3 different test durations. These durations are nominally 3 seconds, 10 seconds and 30 seconds. Actual duration varies from nominal so that whole turns may be included whenever possible. Actual durations are constrained to be within the ranges of 2-4 seconds, 7-13 seconds, and 25-35 seconds, respectively. A single turn is used for the test segments whenever possible.
2. Same/different phone number. Performance are computed separately for test segments from the training phone number versus those segments from different phone numbers. For this test, the handset type label (electret or carbon-button) are the same as that used in training, and it is provided as side information to the system under test.
3. Same/different handset type. Performance are computed separately for test segments with the same handset type label as training, versus segments with a different handset label. For this test, all test segments are from phone numbers different from the training number.

The *test set* comprises 2500 test files per sex, and per test condition, each "pretending" to be 10 clients. The total number of trials to do within NIST evaluations is: 2 sexes $\times$ 9 train–test conditions $\times$ 2500 test files $\times$ 10 trials per test file = 450000. For the evaluation of results, the train–test file couples are separated into following categories: SN (training and test file from the same telephone number), ST (different number but the same handset type), DT (different number and different handset type).

Appendix B

Backpropagation and Kolmogorov's theorem

B.1 Stochastic backpropagation algorithm

The behaviour of artificial neural networks is determinded by their weight values. A common algorithm in learning of weights in multi-layer networks is the so-called *stochastic backpropagation* [1]. We shall explain the algorithm in the case of a three layers networks as depicted in figure B.1. Generalisation to networks with larger number of layers is straightforward. The approach proceeds in two steps. First an input pattern x_n is presented to the network and the output is determined (n is the index of an input pattern choosen randomly in the training set). Secondly, based on an error measure between the actual output and the desired *target* output, the weights are adjusted to reduce error.

The notations are the following. Subscripts i, j and k denote respectively neurons from the input, hidden and output layers. Similarly, connection weights will be noted w_{ij} for input-to-hidden nodes and w_{jk} for hidden-to-output nodes. Activations are noted a_j for neurons on the hidden layer and a_k for neurons on the output layer. Neuron's outputs are noted g_j and g_k for hidden and output layers. The *propagation* function $F_k(x, w)$ of the network is given by :

$$F_k(x,w) = g_k = f(a_k) = f\left(\sum_j g_j w_{jk}\right) = f\left(\sum_j f(a_j) w_{jk}\right) = f\left(\sum_j f\left(\sum_i x_i w_{ij}\right) w_{jk}\right) \quad \text{(B.1)}$$

A common error measure for a pattern n taken randomly[2] in the training set is the sum over outputs of the squared difference between the target output vector t and the actual gotten output vector g_k :

$$E_n = \frac{1}{2}\sum_k (t_{nk} - g_{nk})2 = \frac{1}{2}(t_n - g_n)2 \quad \text{(B.2)}$$

Other error measures can be defined but we shall use this one to explain the backpropagation algorithm. Equations may differ with another error measure but the principle is the same. The learning rule is based on a simple gradient descent whose principle is to change the weight values in a direction that will reduce the error measure :

[1] Other versions of backpropagation (batch and online) will be explained later on.

[2] The way sampling in the training set is performed is of crucial importance, especially for speech processing. This will be discussed later in the text.

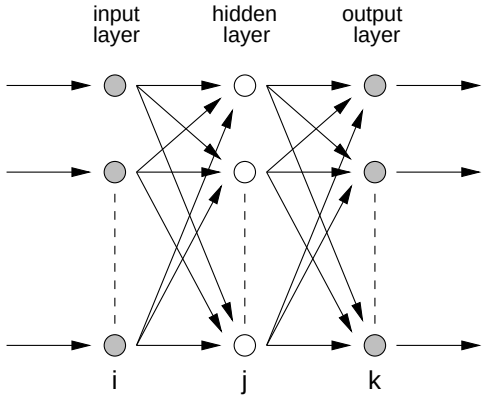


Figure B.1: Muti-layer perceptron with 3 layers. The role of the input layer is to distribute the input features. No computation is actually performed at this stage. Hidden and output layers are computational. Subscripts i, j and k denote respectively neurons from the input, hidden and output layers. Similarly, connection weights will be noted w_{ij} for input-to-hidden nodes and w_{jk} for hidden-to-output nodes.

$$\Delta w_{ij} = -\eta \frac{\partial E_n}{\partial w_{ij}} \tag{B.3}$$

$$\Delta w_{jk} = -\eta \frac{\partial E_n}{\partial w_{jk}} \tag{B.4}$$

where subscripts i, j and j, k in equations B.3 and B.4 denote respectively connection weigths for the input-to-hidden layer and connection weights for the hidden-to-output layer. η is the *learning rate* which indicates the relative size of the change in weights. Weights are updated for each presentation of an input pattern as :

$$w(n+1) = w(n) + \Delta w(n) \tag{B.5}$$

We first consider the hidden-to-output weights w_{jk} of equation B.4. Using the chain rule, we have :

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial w_{jk}} \tag{B.6}$$

where a_k is the activation of neuron k. We have dropped the pattern index n for sake of clarity. The *sensitivity* of neuron k is defined as

$$\delta_k \equiv -\frac{\partial E}{\partial a_k} \tag{B.7}$$

The sensitivity describes how the error changes with the neuron's activation. Using the chain rule, we can write :

$$\delta_k \equiv -\frac{\partial E}{\partial a_k} = -\frac{\partial E}{\partial g_k} \frac{\partial g_k}{\partial a_k} = (t_k - g_k) f'(a_k) \tag{B.8}$$

Coming back to equation B.6, we have :

$$\Delta w_{jk} = \eta \delta_k g_j = \eta (t_k - g_k) f'(a_k) g_j \tag{B.9}$$

From Eq. B.9, we see that the weight update is proportional to $(t_k - g_k)$, which makes intuitive sense. Indeed, if we consider, for example, $(t_k - g_k) > 0$, the gotten output is too small and the weight must be increased, as explained by the learning rule ($f'(a)$ is always positive for a sigmoid or a *tanh* AV). For the ideal case where $t_k = g_k$, the learning rule tells us that the weights should not be changed.

Similarly and pursuing further the application of the chain rule, we can write write the equation of the gradient descent for the input-to-hidden weights of Eq. B.3 :

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial g_j} \frac{\partial g_j}{\partial a_j} \frac{\partial a_j}{\partial w_{ij}} \tag{B.10}$$

The first term of the right hand side can be developped as follows :

$$\begin{aligned} \frac{\partial E}{\partial g_j} &= -\sum_k (t_k - g_k) \frac{\partial g_k}{\partial g_j} \\ &= -\sum_k (t_k - g_k) \frac{\partial g_k}{\partial a_k} \frac{\partial a_k}{\partial g_j} \\ &= -\sum_k \underbrace{(t_k - g_k) f'(a_k)}_{\delta_k} w_{jk} \end{aligned} \tag{B.11}$$

Similarly, we can define the *sensitivity* of a hidden neuron j as

$$\delta_j \equiv -\frac{\partial E}{\partial a_j} = -\frac{\partial E}{\partial g_j} \frac{\partial g_j}{\partial a_j} = f'(a_j) \sum_k \delta_k w_{jk} \tag{B.12}$$

From equation B.12, we see that the sensitivity of a hidden neuron is related to the sum of sensitivities of the output neurons multiplied by the hidden-to-output weights. Sensitivities are said to be *back-propagated* from the output layers to the hidden layer, which explain why the method is called backpropagation. From Eq. B.11, we can write Eq. B.3 as :

$$\Delta w_{ij} = \eta \delta_j x_i = \eta x_i f'(a_j) \sum_k \delta_k w_{jk} \tag{B.13}$$

B.2 MLP inversion

We can pursue further the backpropagation rule to discover in which direction the input values x_i should be moved to decrease the error measure. This problem is called *inversion* of systems in the litterature and it solution will be used in the framework of data-driven feature extraction. The inversion rule, also based on a gradient descent, changes the input values in a direction that will reduce the error measure :

$$\Delta x_i = -\eta \frac{\partial E}{\partial x_i} \tag{B.14}$$

Taking the definitions of sensitivities from Eq. B.7 and B.12 and pursuing further the application of the chain rule, we can write from equation B.3 :

$$\begin{aligned}\frac{\partial E}{\partial x_i} &= \sum_j \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial x_i} \\ &= \sum_j \delta_j w_{ij} \\ &= \sum_j \sum_k \delta_k w_{jk} f'(a_j) w_{ij} \end{aligned} \tag{B.15}$$

A new set of input feature vectors can be obtained with :

$$x(n+1) = x(n) + \Delta x(n) \tag{B.16}$$

B.3 MLP and Bayes theory

Let's suppose we train the network for doing classification with K output neurons, one for each class ω_k. The target values [3] are set to :

$$t_k = \begin{cases} 1 & \text{if } x \in \omega_k \\ 0 & \text{otherwise} \end{cases} \tag{B.17}$$

We show here that, with this type of training, the backpropagation rule will make asymptotically fit the network's output to the Bayes discriminant functions when minimizing the squared error. We use the network and notations of figure B.1.

The training error for output k is :

$$\begin{aligned} E_k(w) &= \sum_x (F_k(x,w) - t_k)^2 \\ &= \sum_{x\in\omega_k} (F_k(x,w)-1)^2 + \sum_{x\notin\omega_k} (F_k(x,w)-0)^2 \\ &= n\left\{ \frac{n_k}{n}\frac{1}{n_k}\sum_{x\in\omega_k}(F_k(x,w)-1)^2 + \frac{n-n_k}{n}\frac{1}{n-n_k}\sum_{x\notin\omega_k} F_k^2(x,w)\right\} \end{aligned} \tag{B.18}$$

where n_k is the number of patterns in class ω_k and n is the total number of patterns. In the limit of infinite data, we can write :

$$\begin{aligned} \lim_{n\to\infty}\frac{1}{n}E_k(w) &= P(\omega_k)\int (F_k(x,w)-1)^2 p(x|\omega_k)dx + P(\omega_{i\neq k})\int F_k^2(x,w)p(x|\omega_{i\neq k})dx \\ &= \int F_k^2(x,w)(P(\omega_k)p(x|\omega_k) + P(\omega_{i\neq k})p(x|\omega_{i\neq k}))dx \\ &\quad -2\int F_k(x,w)P(\omega_k)p(x|\omega_k)dx + \int P(\omega_k)p(x|\omega_k)dx \end{aligned} \tag{B.19}$$

Recalling Bayes'rule :

$$P(\omega_k|x) = \frac{P(x|\omega_k)P(\omega_k)}{P(x)} = \frac{P(x,\omega_k)}{P(x)} \tag{B.20}$$

[3] If target values are equal to the actual posterior probabilities, the property that is going to be developped is still valid

and a property of mutually exclusive events ω_k and $\omega_{i\neq k}$

$$P(x) = P(x, \omega_k) + P(x, \omega_{i\neq k}) \tag{B.21}$$

we can write Eq. B.19 as

$$\begin{aligned} \lim_{n\to\infty} \frac{1}{n} E_k(w) &= \int F_k^2(x,w)p(x)dx - 2\int F_k(x,w)p(x,\omega_k)dx + \int p(x,\omega_k)dx \\ &= \int (F_k(x,w) - P(\omega_k|x))^2 p(x)dx + \int P(\omega_k|x)(1-P(\omega_k|x))p(x)dx \end{aligned} \tag{B.22}$$

The second term of the right hand side of Eq. B.22 is independent of the weights values w. Since the backpropagation rule minimizes $E_k(w)$, it minimizes the first term of Eq. B.22 :

$$\int (F_k(x,w) - P(\omega_k|x))^2 p(x)dx \tag{B.23}$$

which is true for each class w_k $(k = 1 \dots K)$. Backpropagation thus minimizes the sum :

$$\sum_{k=1}^{K} \int (F_k(x,w) - P(\omega_k|x))^2 p(x)dx \tag{B.24}$$

and the outputs of the trained network will approximate the a posteriori probabilities, provided that the network can indeed represent the functions $P(\omega_k|x)$, provided an infinite amount of training data, provided an optimal gradient descent (that does not fall into local minima), etc. In real-life application, these conditions are not guaranteed and the outputs will not represent probabilities.

B.4 Kolmogorov's theorem

We are interested here in the question : Can every decision boundary be implemented by three layers networks of neurons ? Equivalently, can any continuous function from input to output be implemented in a three-layer net, given sufficient number of hidden units, proper nonlinearities and weights ? The anser is *yes* and comes from Kolmogorov (and others later). Kolmogorov prooved that any continuous function $F(\mathbf{x})$ defined on the unit hypercube I^n ($I = [0,1]$ and $n \geq 2$) can be represented in the form

$$F(\mathbf{x}) = \sum_{j=1}^{2n+1} \Xi_j \left(\sum_{i=1}^{n} \psi_{ij}(x_i) \right) \tag{B.25}$$

for properly choosen functions Ξ_j and ψ_{ij}. Equation B.25 states that any function $F(\mathbf{x})$ can be expressed as the sum of $2n+1$ functions taking as input a sum of n functions, one for each components of the input feature vector $\mathbf{x}$. The link with neural networks is quite straightforward. In neural network terminology, the Ξ_j would be hidden neuron's activation functions, each connected to a layer of n neurons with nonlinear activation functions ψ_{ij} taking as input the features x_i. Nevertheless, the relationship of Kolmogorov's theorem to practical neural networks is quite tenuous, mainly because the functions Ξ_j and ψ_{ij} are not the simple weighted sum passed through nonlinearities used as, for example, in MLPs. Furthermore, the theorem does not tell us how to find the form of the functions Ξ_j and ψ_{ij} based on couples of known input-output data, as it is done in MLP training.

A more intuitive answer to the above questions is inspired by Fourier's theorem that states that any continuous function $F(\mathbf{x})$ can be approximated arbitrarily closely by a sum of harmonic functions. Such harmonic functions (sine wave for example) could possibly be implemented by neural networks whose

hidden units have smooth sigmoidal activation functions. One can imagine that neurons are paired in opposition, so each pair implements a bump that are summed to form a sine wave. Another set of neurons could implement a sine wave of a different frequency, and so forth. Ad hoc hidden-to-output weights computed as in a Fourier synthesis would then enable the full network to implement the desired function $F(\mathbf{x})$.

Kolmogorov or Fourier theorem are conceptual tools that do not explain how neural networks in fact function. In practice, we never find that through training, the network is pairing sigmoids to get bumps so that a Fourier-like representation is performed. It just gives us confidence in the power of neural networks in that any desired function can be implemented by a three-layer network. On the other hand it does not tell us how to do it. Even if there were constructive proof, it would be of little interest in practical pattern recognition since we do not know the desired function (we are not in a Fourier synthesis problem) and that desired function is related to the training data set in very complicated ways.

Appendix C

Review of other ANN approaches for ASR and ASkR

This appendix is dedicated to the analysis of some Artificial Neural Network (ANN) approaches that were not detailed in previous chapters. Their basic principles are briefly described as well as their current applications and performances in speech and speaker recognition.

C.1 Dumb multilayer perceptron for isolated word recognition

The most obvious way to use multilayer perceptrons for speech recognition is to present all acoustic vectors of a speech unit (phoneme of word) at once at the input and to detect the most probable speech unit at the output by determining the output neuron with the highest activation (Fig. C.1, for word recognition). The learning algorithm can be the conventional backpropagation, or a more sophisticated variation of it. In the learning phase, the desired output is 1 for the correct speech unit, and 0 for all other speech units. In this way, not only the correct output is reinforced for the corresponding sequence of acoustic vectors, but simultaneously the wrong outputs are weakened. For this reason, the multilayer perceptron has a better discriminating ability than the hidden Markov model.

The problems associated with this approach are multiple. Clearly, for word recognition, a huge number of input units has to be used. This implies an even larger number of parameters to be determined by learning and consequently the necessity to dispose of a large database. The approach is only feasible for a small vocabulary of isolated words. Its use for continuous speech is out of question.

The method seems to be more appropriate for phoneme recognition. However, in this case, a phoneme segmented database has to be available for learning, which often is not the case. Furthermore, for recognition, in principle the speech signal has to be phoneme segmented which is a nontrivial task. In the corresponding approach with hidden Markov models, the time alignment is performed automatically in the recognition phase by the Viterbi algorithm.

Actually, the time variability of speech in general presents a problem for the multilayer perceptron. The same word pronounced by different speakers or even by the same speaker has quite different durations. Therefore, with a fixed number of inputs to the network, either some acoustic vectors have to be cut if the word pronunciation is too long, or some inputs have to be set to some arbitrary values if the word pronunciation is too short. This implies that a given neuron has not always an input that corresponds to the same part of the word pronunciation. This certainly influences negatively the recognition rate.

In order to improve the situation, various methods to align the speech signal to a fixed length, before feeding it to the multilayer perceptron, have been proposed as in (Huang and Kuh 1992).

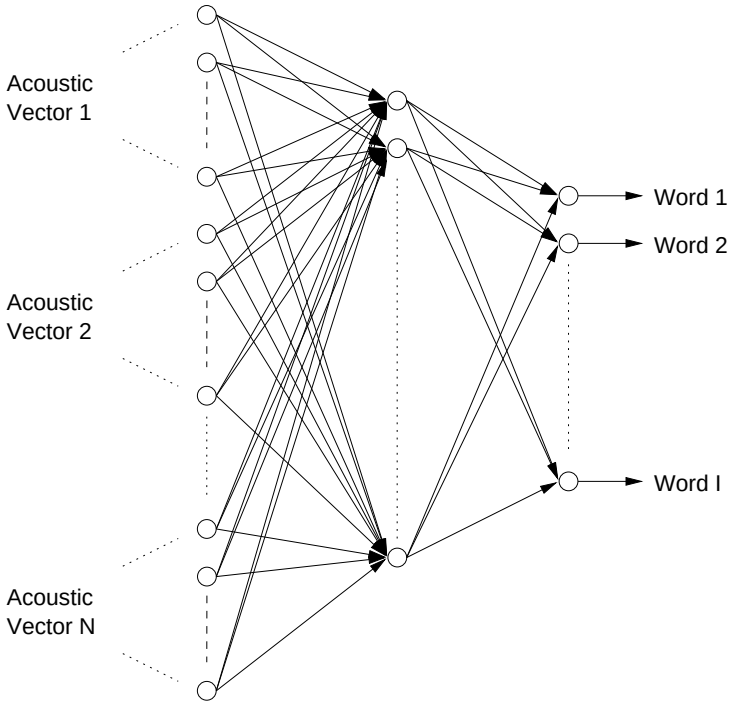


Figure C.1: Simple word recognition by a multilayer perceptron.

C.2 Time-delay neural network for recognition

Speech recognition experiments using MLPs have been successfully carried out mostly on isolated word recognition for a small vocabulary (e.g. digit recognition task). This obvious limitation in performance of the pure MLP approach is a consequence of the inability of the MLP to deal properly with the dynamic nature of the speech as well as its intrinsic variability.

In order to take into account temporal relationships between acoustic events it has been proposed by Waibel (Waibel, Hanazawa, Hinton, Shikano, and Lang 1989) to modify the architecture of the MLP in such a way that in each layer, delayed inputs are weighted and summed. This modification gives the ability to relate and compare the current inputs to their past history.

The basic unit of a MLP weights its inputs and sums them through a nonlinear function. In the TDNN version, this basic unit is modified by introducing delays. Figure C.2 shows the basic unit of a TDNN in which n delayed versions of each input is taken into account.

Since hidden layers are also delayed, the TDNN is not really equivalent to an MLP which takes into account more context, but certainly, as claimed by its authors, allows the discovery of the relationships between the acoustic-phonetic features.

The training of the TDNN network can be carried out using the backpropagation method traditionnally used for MLP.

Only a few experiments have been reported with TDNN. In (Waibel, Hanazawa, Hinton, Shikano, and Lang 1989) and (Lang, Waibel, and Hinton 1992) the TDNN has been used for a speaker-dependent recognition of the English phonemes "B", "D" and "G" in varying contexts, furthermore comparisons were made with several HMMs trained to perform the same task. Performance evaluation showed that the TDNN achieved a 98.5% recognition rate while the rate obtained with HMM was 93.7%.

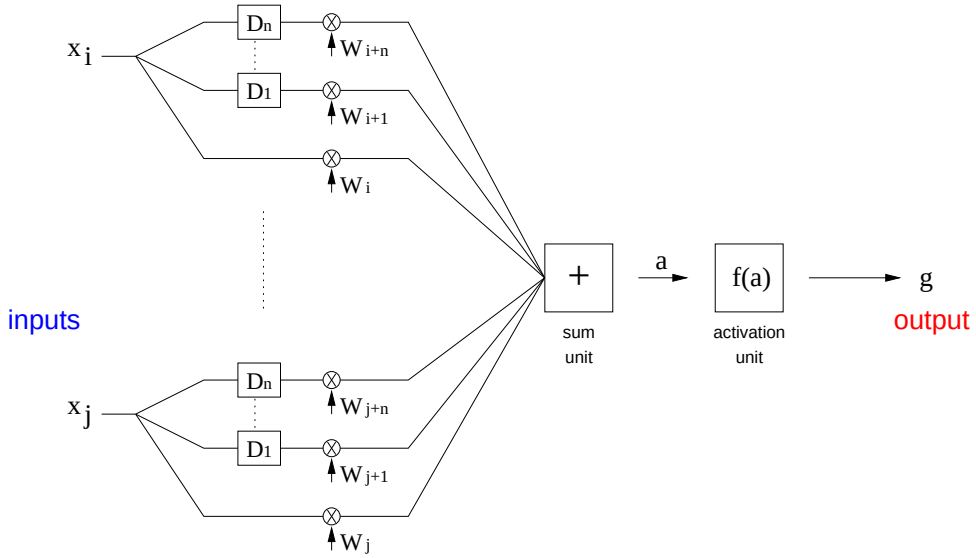


Figure C.2: Basic Time Delay Unit

C.3 Hidden control neural network for recognition

Multilayered neural nets have been mainly proposed as universal approximators for system modeling and nonlinear prediction. However if they are very well suited in the case of time-invariant nonlinear systems, it has been extremely difficult even impossible to apply them directly in the case of complicated nonstationary signals, such as speech signals. The reason for this failing is obvious, it is quite impossible that a network with fixed parameters can take into account and characterise the temporal and spectral variabilities of speech signals. In most of the reported experiments with nonlinear prediction using MLP, additional mechanisms have been implemented in order to enable the network to cope with the time-varying dynamics of the speech signals. In short at some stage a switching control mechanism on the weights of the network has to be implemented or at least as described above a modification (i.e. time alignment) of the input signal has to be performed.

In order to cope with the nonstationary nature and variability of speech signals a control neural architecture has been recently proposed, called the "Hidden Control Neural Network" (HCNN). The nonlinear network (MLP) is used as a predictor model of the speech signal. "A hidden control input signal, allowing the network's mapping to change over time, provides the ability to capture the nonstationary properties and to learn the underlying temporal structure of the modelled signal" (Levin 1990) (Levin 1993) (Iso 1993).

The initial idea has been proposed in (Levin 1990), however HCNN have been also investigated in (Iso 1993) in the context of speech recognition. In this context the HCNN can be viewed as a source of statistical models.

Suppose that our task is to recognize M words of a vocabulary and that for instance each word (as in the case of the HMM) can be represented by a sequence of left-to-right hidden states the number of which are supposed known for each word. Basically the idea behind HCNN is to train a MLP which is optimized as a predictor of speech features. The originality is that for each word an optimal sequence of control signals has to be discovered. This control signal is merely a constant real input vector which feeds the MLP during the active time of its associated state. It is worth noting that there is only one MLP common for all the M words and not M concurrent MLPs.

The approach is comparable to the HMM approach, indeed, the sequence of the hidden control inputs can be related to the hidden sequence of the HMM states.

This control signal sequence is said to be optimal in the sense that it maximizes the likelihood (average of the logarithmic probabilities to observe each of the M words given their associated control sequence).

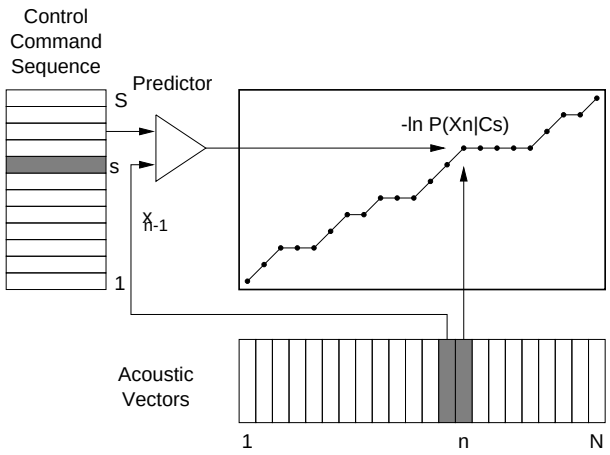


Figure C.3: HCNN model architecture (after Iso).

The main difficulty in the training phase is that various parameters have to be determined (MLP weights, values of each component of the control sequence in a given state) and that a time alignment algorithm has to be used in order to find the optimal firing instants of the M consecutive control signals.

After the training phase each word is associated with a sequence of control signals and for each word a mapping of the probability of the observation given any of the M control signal sequences is available.

In the recognition phase one has to find the most probable sequence of control signals mainly by a maximisation of the likelihood using a time-alignment algorithm.

We give below a flavour of the method presented in (Iso 1993). Consider figure C.3 where the so-called input speech is a sequence of feature vectors $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ (D components). It is assumed that each basic speech unit (phoneme) is associated with a certain number of control signal vectors. For instance each phoneme will be associated with three or four control signals (three or four states in the HMM terminology). A word control sequence is obtained as a concatenation of the phonemes control signals. In short if there are S states for a word, there are S control signals $\{c_1, c_2, \cdots, c_S\}$ for a word which have to be determined in the training phase. In (Iso 1993), an MLP which acts as a nonlinear predictor is driven by this control sequence. The MLP is fed with P preceding feature vectors and with the control command c_s ($P = 1$ in fig. C.3). The output is the predicted value

$$\hat{\mathbf{x}}_{n,s} = f(\mathbf{x}_{n-1}, \cdots, \mathbf{x}_{n-P}, c_s) \tag{C.1}$$

Figure C.3 shows the behaviour of the HCNN in the recognition phase, for each competing word to be recognized a time alignment algorithm computes the best instant firing of the N consecutive control signals associated with each of the words. A score is then computed, in short this score is linked to the prediction power error of the predictive MLP given the control sequence. The "best" score, i.e. the best control sequence traces back to the associate word which is the recognized word.

(Iso 1993) makes the assumption that the prediction error is distributed following a gaussian, the probability of producing the speech feature vector given the control command is

$$-lnP(\mathbf{x}_n|c_s) = \frac{1}{2}(\mathbf{x}_n - \hat{\mathbf{x}}_{n,s})^T \Sigma_s^{-1} (\mathbf{x}_n - \hat{\mathbf{x}}_{n,s}) + \frac{1}{2} ln(2\pi)^D |\Sigma_s| \tag{C.2}$$

where Σ_s is the $D \times D$ covariance matrix for each control command c_s. With this formulation and making the usual assumption as in the HMMs framework (independence of acoustic events and independence of control commands), the probability of an acoustic observation given the control command sequence associated with the m^{th} word of the vocabulary is

$$P(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N | c_{m,1}, c_{m,2}, \cdots, c_{m,S_m}) = \prod_{n=1}^{N} P(\mathbf{x}_n | c_{m,s(n)}) \tag{C.3}$$

The control command sequence which maximises the probability of the acoustic observation defines the most probable word (whose index is m^*) given the model parameter:

$$P(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N | c_{m^*,1}, c_{m^*,2}, \cdots, c_{m^*,S_m^*}) = \max_m \{ \max_{c_{m,1}, c_{m,2}, \cdots, c_{m,S_m}} \prod_{n=1}^{N} P(\mathbf{x}_n | c_{m^*,s(n)}) \} \tag{C.4}$$

In the recognition phase, the search of $\max_{c_{m,1}, c_{m,2}, \cdots, c_{m,S_m}}$ determines the time-alignement between the input speech feature vector sequence and control command sequences. The optimal time-alignment hich gives the maximum probability is determined by dynamic programming.

The MLP weights, the control command and the covariance matrix have to be found during the training phase. Two criteria can be used for this optimization problem. The first one tempts to maximise the average of the logarithmic probabilities in C.4 for all the training utterances (maximum likelihood criterion). The second one is a discriminative criterion. For more details, the reader is referred to (Iso 1993), (Levin 1993) and (Martinelli 1994).

(Iso 1993) reports speaker-dependent English spoken letter recognition experiments in which the HCNN configuration exhibited a recognition accuracy rate of 98.8 %. The vocabulary consisted of the 26 letters of the English alphabet. The database contained about 5000 connected letters.

Bibliography

Huang, Z. and A. Kuh (1992, November). A combined self-organizing feature map and multilayer perceptron for isolated word recognition. *IEEE Transactions on Signal Processing 40*(11), 2651–2657.

Iso, K. I. (1993). Speech recognition using dynamical model of speech production. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Minneapolis, pp. 283–286.

Lang, J., A. Waibel, and G. E. Hinton (1992). *A Time-Delay Neural Network Architecture for Isolated Word Recognition*, Chapter 0, pp. 388–408. Artificial Neural Networks, Paradigms, Applications and Hardware Implementations. IEEE Press.

Levin, E. (1990). Word recognition using hidden control neural architecture. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Albuquerque, pp. 433–436.

Levin, E. (1993). Hidden control neural architecture modeling of nonlinear varying systems and its applications. *IEEE Transactions on Neural Networks 4*(1), 109–116.

Martinelli (1994, March). Hidden control neural network. *IEEE Transactions on Circuits and Systems-II : Analog and Signal Processing 41*(3), 245–247.

Pirelli, J. et al. (1995). Phonebook: A phonetically-rich isolated word telephone speech database. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Detroit.

Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang (1989, March). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Siganl Processing 37*(3), 329–339.

List of Figures

2.1 Automatic Speech Recognition system (a) and Automatic Speaker Recognition system (b) 12
2.2 Speech signal of the word *accumulation* taken from the Phonebook database. (a) : waveform, (b) partial waveform, (c) narrowband spectrogram of (a), (d) power spectrum magnitude of (b). . . . 13
2.3 Breakdown of the ASR recognition system. **Pre-processing** : a feature analysis transforms the speech waveform in a better representation for classification. **Pattern matching** : a pattern classification block matches acoustic features on acoustic models and outputs a set of winner model hypothesis. **Post-processing** : languages models are applied to discard wrong hypothesis. . . . 16
2.4 Breakdown of the ASkR system. **Pre-processing** : a feature analysis transforms the speech waveform in a better representation for classification. **Pattern matching** : pattern classification matches acoustic features with speaker models and outputs identity scores. **Post-processing** : a thresholding is applied on the identity scores and a identity decision is taken. . . . 16
2.5 LPC-cepstrum extraction, after Rabiner. . . . 17
2.6 Hamming window, $L = 240$ (left) and liftering window, $D = 12$ (right) . . . 19
2.7 Left-to-right HMM with self loop . . . 21
2.8 Forward and backward recursion. The HMM topology is left-right but permits in this case to skip the next state and to go to the second next state. The forward recursion propagates from left to right to compute the α probabilities. For example, the lattice point corresponding to $\alpha_n(3)$ can only be reached from states q_1, q_2 and q_3, and the recursion in this point is $\alpha_n(3) = [\alpha_{n-1}(1)p(q_3|q_1) + \alpha_{n-1}(2)p(q_3|q_2) + \alpha_{n-1}(3)p(q_3|q_3)]\, p(x_n|q_3)$. The backward probabilities are computed in a similar manner but propagating the recursion from right to left. Any HMM topology can be taken into account, the beam of forward/backward connections getting larger with more intricated topologies. . . . 25
2.9 Example of Viterbi alignment from the Phonebook database. a) `prolongs` b) `videotapes`. 28
2.10 Duration modeling with HMMs. a) Topology with no duration modeling. b) Histogram type duration modeling in which transition probabilities between a state and a final phone is set equal to a particular duration probability. c) Minimum duration modeling in which a state is forced to be visited a given number of time. . . . 30
2.11 Artificial neuron or perceptron : basic processing unit which performs a non-linear operation on the weighted sum of its inputs. . . . 30
2.12 Activation functions. a) Sigmoid function $f(a) = \frac{1}{1+\exp(-x)}$. b) Hyperbolic tangent $f(a) = \tanh(a)$. . . . 31
2.13 Muti-layer perceptron with L layers of neurons. . . . 32
2.14 Muti-layer perceptron with 3 layers. The role of the input layer is to distribute the input features. No computation is actually performed at this stage. Hidden and output layers are computational. Subscripts i, j and k denote respectively neurons from the input, hidden and output layers. Similarly, connection weights will be noted w_{ij} for input-to-hidden nodes and w_{jk} for hidden-to-output nodes. . . . 32
2.15 Bias can be treated as connection weights from an additional neuron whose output is clamped to 1. . . . 33

3.1 Discrete ASR system. A vector quantization is performed at the output of the feature extraction front-end, mapping the acoustic vector sequence into a centroid label vector sequence. . . . 40
3.2 Kohonen self-organizing map. The network is made up of two layers of neurons : the **input layer** whose function is to distribute the inputs and the **output layer** with its lateral interaction which is active during training time. . . . 43
3.3 Different types of neighbourhood functions. a) Rectangular neighbourhood, b) exponential neighbourhood, c) Mexican hat neighbourhood. For the Mexican hat function, neurons close to the winner neuron on the map are driven towards the input vector while neurons far from the winner are repelled. . . . 44
3.4 Discrete probability density functions for HMMs. A histogram of probabilities is tied to each state of the HMM. . . . 45

4.1 Expectation maximisation training of hybrid HMM/ANN system. . . . 57
4.2 Training of hybrid HMM/ANN system. a) Expectation phase : MLP parameters are fixed and best state paths are computed with the Viterbi criterion. State labels are associated to acoustic vectors. b) Maximisation phase : MLP parameters are trained with the newly gotten targets. . . . 58
4.3 Words `accumulation` from the Phonebook database. a) Waveform with Viterbi alignment. b) Gammas with Viterbi criterion. c) Gammas with forward-backward criterion. . . . 69
4.4 Words `berkovitz` from the Phonebook database. a) Waveform with Viterbi alignment. b) Gammas with Viterbi criterion. c) Gammas with forward-backward criterion. . . . 70

5.1 Speaker verification system. . . . 77
5.2 EER averaged per phoneme with 20 hidden nodes and 3 input frames for the MLPs. . . . 83
5.3 EER averaged per phonemic group with different acoustic window length at the input of the MLP. The number of hidden nodes equals 20 and is kept constant for all the experiments. Cxy means x left frames and y right frames of context as MLP input. . . . 83
5.4 Global and segmental speaker verification system. . . . 86
5.5 Example of temporal decomposition (after Jan Cernocky). Upper part : sample sequence; middle part : spectrogram; bottom part : interpolation functions. . . . 87
5.6 GMM system. Upper figure: results for training condition 2F with different testing conditions. Bottom figure: results for 10 sec test length using different training conditions. SN stands for training and test file from the same telephone number, ST stands for different number but the same handset type, DT stands for different number and different handset type. . . . 90
5.7 Segmental system, results by classes, training condition 2F, test duration 30 sec, SN test condition (training and test file from the same telephone number). . . . 91
5.8 Segmental system, log-likelihood score distribution for, respectively, class 6 and 7 for the female speakers. Training condition 2F and testing duration 30 sec. . . . 93
5.9 Segmental system, results by class, training condition 2F, test duration 30 sec, "same number" (SN) condition for testing. . . . 94
5.10 Global and segmental system, training condition 2F, test duration 30 sec, same number (SN) and different type. . . . 94

B.1 Muti-layer perceptron with 3 layers. The role of the input layer is to distribute the input features. No computation is actually performed at this stage. Hidden and output layers are computational. Subscripts i, j and k denote respectively neurons from the input, hidden and output layers. Similarly, connection weights will be noted w_{ij} for input-to-hidden nodes and w_{jk} for hidden-to-output nodes. . . . 112

C.1 Simple word recognition by a multilayer perceptron. . . . 118
C.2 Basic Time Delay Unit . . . 119
C.3 HCNN model architecture (after Iso). . . . 120

List of Tables

3.1 Error rates on the HIM corpus obtained using LBG and k-means vector quantization methods. Codebooks (*a*-*b*-*c*-*d*) means that we used 4 codebooks : *a* centroids for delta energy; *b* centroids for delta-delta energy; *c* centroids for cepstrum; *d* centroids for delta-cepstrum. 46

3.2 Error rates on the HIM corpus obtained using Kohonen and Kohonen+k-means vector quantization methods. Codebooks (*a*-*b*-*c*-*d*) means that we used 4 codebooks : *a* centroids for delta energy; *b* centroids for delta-delta energy; *c* centroids for cepstrum; *d* centroids for delta-cepstrum. 46

3.3 Error rates on the HER corpus obtained using LBG and k-means vector quantization methods. Codebooks (*a*-*b*-*c*-*d*) means that we used 4 codebooks : *a* centroids for delta energy; *b* centroids for delta-delta energy; *c* centroids for cepstrum; *d* centroids for delta-cepstrum. 47

3.4 Error rates on the HER corpus obtained using Kohonen and Kohonen+k-means vector quantization methods. Codebooks (*a*-*b*-*c*-*d*) means that we used 4 codebooks : *a* centroids for delta energy; *b* centroids for delta-delta energy; *c* centroids for cepstrum; *d* centroids for delta-cepstrum. 47

3.5 Error rates on the HER corpus obtained using k-means and LVQ vector quantization methods. Results are obtained using 12 CMS as feature vectors. The notation 126(126) means that 126 centroids are used in the codebook and that (126) labels (42 CIP times 3 states) are used. For the 378(126) configuration, 3 centroids are tied to each state, leading to a codebook size of 378. 49

3.6 Error rates on the HER corpus obtained using k-means and LVQ vector quantization methods. Results are obtained using 12 CMS as feature vectors. For configurations 32-32-378(126)-378(126), no supervision for ΔE and $\Delta\Delta E$ codebook is used, while CMS and ΔCMS are supervised (3 centroids per class). 49

4.1 Summary of the different algorithms unified with the formalization. 63

4.2 Results are obtained on the HER database with discrete, Gaussian mixtures and continuous Hybrid HMMs using CIP models. Cxy defines the input size of the MLP and means that x frames on left and y frames on right of the central frame are taken. Number of HMM phones and states are given in columns 2 and 3. The number of parameters, conditioning the required memory to run the system, is reported in column 4. The decoding time is reported in column five, taking as reference the discrete system. Error rates are given in the last column. 68

4.3 Results are obtained on the HER database with discrete, Gaussian mixtures and continuous Hybrid HMMs using CDP models. 68

4.4 Results on the Phonebook database with Gaussian mixtures and Hybrid HMMs using CIP models. Number of HMM phones and states are given in columns 2 and 3. The number of parameters is reported in column 4. Error rates are given in the last two columns (no minimum duration modeling and minimum duration modeling). Hybrid systems are trained with the Viterbi criterion. 68

4.5 The error rates are obtained on the Phonebook database with continuous Hybrid HMMs for two training sets with different sizes. Training set 1 is considerably smaller than training set 2. For training set 1, a C44 MLP with 600 hidden units has been used. For training set 2, a C44 MLP with 1000 hidden units has been used. 68

5.1 EER averaged per phonemic group with different acoustic window length at the input of the MLP. The number of hidden nodes is kept constant. 83
5.2 EER averaged per phonemic group with different number of hidden nodes in the MLP. . . 84
5.3 EER averaged per phonemic group with two different training sampling modes of the MLP. 85

A.1 Some general statistics about the word lists in the training, cross and test set. The number of frames has been computed with a frame length of 200 samples and a frame shift of 80 samples (frame overlap of 120 samples) 105
A.2 Some statistics about genders and number of speakers in each word list of the training set. 106
A.3 Some statistics about genders and number of speakers in each word list of the cross validation set. 107
A.4 Some statistics about genders and number of speakers in each word list of the test set . . 107
A.5 Context independent phoneme list for the HER database. 108

Glossary - Acronyms

ALISP Automatic Language Independent Speech Processing.

ANN Artificial Neural Network.

ASR Automatic Speech Recognition.

ASkR Automatic Speaker Recognition.

ASSP IEEE Transactions on Acoustics, Speech and Signal Processing.

CIRC Chaire des Circuits et Systèmes (Circuits and Systems Chair).

CMS Cepstral Mean Substraction. Usual operation also referred as blind deconvolution consisting in removing the average vector from the sequence of acoustic vectors in order to equalize the effects of the communication channel.

Continuous speech recognition Recognition of continuously spoken input, with no pauses between words. Contrasts with isolated word recognition

coarticulation Modification of the pronunciation of a phoneme because of the surrounding phonetic context. This effect is related to the inertia of the vocal tract that can not move instantaneously from one state to another state.

DET Detection Error Tradeoff.

DTMF The technical term for TouchTone (stands for Dual Tone Multi-frequency).

EER Equal error rate.

ETSI European Telecommunications Standards Institute.

GBSI EER Gender balanced sex-independent equal error rate.

GMM Gaussian mixtures modelling.

HCNN Hidden-control neural network.

HER Swiss-german telephone isolated word database. The name was given next to the recording of the HIM database.

HIM Isolated word american english telephone database. Named after the Himmarnnet European Esprit Project.

HMM(s) Hidden Markov Model(s).

Hnorm Handset normalization.

ICASSP IEEE International Conference on Acoustics, Speech and Signal Processing.

ICSLP International conference on speech and language processing.

IEEE The Institute of Electrical and Eelectronics Engineers.

ISO International Standards Organisation.

Isolated word recognition recognition of speech, spoken one word at a time. Contrasts with continuous speech recognition.

Large vocabulary systems Recognition systems which have vocabularies of several thousand words.

LBG Linde-Buzo-Gray.

Lombart effect Changes in articulation due to the environment influence. For example, when a talker speaks in a environment with a masking noise, it has been reported that the first formant of a vowel increases while the second formant decreases. The difficulty with Lombart effect is a lack of understanding as to how to quantify it.

LPC Linear Predictive coefficients. Coefficients of all-pole filters used to modelize frames of speech signal. An efficient recursion can be used to compute cepstrum features from LPC features.

LVCSR Large Vocabulary Continuous Speech Recognition.

LVQ Learning Vector Quantization.

MLP Multi-layer perceptrons.

MSE Mean square error.

NIST National Institute of Standards and Technology.

NLP Natural Language Processing.

NSA National Security Agency.

pdf Probability density function.

perplexity or word average branching factor, measure on average of the number of words following a given word. Several large vocabulary recognition tasks have shown a perplexity to be on the order of 100.

phone Actual sounds produced by the vocal tract while speaking and that correspond to an ideal phoneme definition.

phoneme A phoneme is often defined as an ideal sound unit associated with a set of articulatory gestures. The phonemes of a language, therefore comprise a minimal theoretical set of units that are sufficient to convey all meaning in the language. Due to many different factors (accent, gender, coerticulatory effects, ...), a phoneme will have a variety of acoutic manifestation. The related and actual sounds that are produced in speaking are called the phones.

RLA2C La Reconnaissance du Locuteur et ses Applications Commerciales et Criminalistiques.

ROC Receiver operating curve.

SI Speaker identification

SOM Self-Organizing Maps of Kohonen. Familly of neural networks trained by following a non-supervised algorithm.

STRUT Speech Recognition Software Toolkit.

SV Speaker Verification.

TD Temporal decomposition.

TDNN Time delay neural network.

touchtone The system of tones generated by most modern, push-button telephones.

VQ Vector quantization.

Znorm Centered normalization.

General bibliography

Almeida, A. P. and J. E. Franca (1996, March). Digitally programmable analog building blocks for the implementation of artificial neural networks. *IEEE Transactions on Neural Networks 7*(2), 506–514.

Artières, T. (1995). *Méthodes prédictives neuronales: application à l' identification du locuteur*. Ph. D. thesis, Université de Paris XI Orsay.

Atal, B. S. (1983). Efficient coding of lpc parameters by temporal decomposition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 81–84.

Bahl, L., F. Jelinek, and R. Mercer (1983, March). A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 5*(2), 179–190.

Bahl, R. and al. (1989, May). Large vocabulary natural language continuous speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 465–468.

Bennani, Y. and P. Gallinari (1994, April). Connectionist approaches for automatic speaker recognition. In *ESCA Workshop on Automatic Speaker Recognition, Identification and Verification*, Martigny, Switzerland, pp. 95–102.

Besacier, L. and J.-F. Bonastre (1997, August). Independent processing and recombination of partial frequency bands for automatic speaker recognition. In *Fourteenth Internationnal Conference on Speech Processing*, Seoul, Korea. IEEE Korea Council, IEEE Korea Signal Processing Society.

Bimbot, F. (1990). An evaluation of temporal decomposition. Technical report, Acoustic research departement AT&T Bell Labs.

Bimbot, F. and G. Chollet (1995). *EAGLES Handbook on Spoken Language Resources and Assessment*, Chapter Assessment of speaker verification systems. Mouton de Gruyter.

Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Clarendon Press Oxford.

Bourlard, H., Y. Konig, and N. Morgan (1995, September). Remap : Recursive estimation and maximization of a posteriori probabilities in connectionist speech recognition. In *EUROSPEECH'95*, Madrid.

Bourlard, H., Y. Konig, and N. Morgan (1996). A training algorithm for statistical sequence recognition with applications to transition-based speech recognition. *IEEE Signal Processing Letters 3*(7), 203–205.

Bourlard, H. and N. Morgan (1993, November). Continuous speech recognition by connectionist statistical methods. *IEEE Transactions on Neural Networks 4*(6), 893–909.

Bourlard, H. and N. Morgan (1994). *Connectionist Speech Recognition*. Kluwer Academic Publishers.

Bourlard, H. and C. Wellekens (1988). Links between markov models and multilayer perceptrons. In D. Touretzky (Ed.), *Advances in Neural Information Processing*, Volume 1, San Mateo CA, pp. 502–510. Moran Kaufmann.

Bourlard, H. and C. J. Wellekens (1990, December). Links between markov models and multi-layer perceptrons. *IEEE Trans. Patt. Anal. Machine Intell. 12*(12), 1167–1178.

Boves, L. (1998). Commercial applications of speaker verification: Overview and critical success factors. In *Speaker Recognition and Its Commercial and Forensic Applications (RLA2C)*, Avignon, France, pp. 150–159.

Cerf, P. L., W. Ma, and D. V. Compernolle (1994, January). Multilayer perceptrons as labelers for hidden markov models. *IEEE Transactions Acoust., Speech, Signal Processing 2*(1), 185–193. Part II.

Cernoký, J., G. Baudouin, and G. Chollet (98, may). Segmental vocoder-going beyond the phonetic approach. In *Submitted to IEEE ICASSP98*, Seattle, WA, US.

Cheng, D.-Y. and A. Gersho (1986). A fast codebook search algorithm for nearest-neighbor pattern matching. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 265–268.

Chollet, G., J. Černocký, A. Constantinescu, S. Deligne, and F. Bimbot (in press). *NATO Advanced Study Institute: Computational models of speech pattern processing*, Chapter Towards ALISP: a proposal for Automatic Language Independent Speech Processing. Springer Verlag.

Cohen, M., H. Franco, N. Morgan, D. Rumelhart, and V. Abrash (1992). Hybrid neural network/hidden markov model continuous speech recognition. In *Proceedings of Intl. Conf. On Speech and Language Processing*, Banff, Canada, pp. 325–328.

Comerford, R., J. Makhoul, and R. Schwartz (1997, December). The voice of the computer is heard in the land (and it listens too. *IEEE Spectrum 34*(12).

Deller, J., J. Proakis, and J. Hansen (1993). *Discrete-Time Processing Of Speech Signals*. Macmillan Publishing Company.

Doddington, G. (1985, November). Speaker recognition - identifying people by their voices. *Proceedings of the IEEE 73*(11), 1651–1664.

Duda and Hart (1973). *Pattern Classification and Scene Analysis*. Wiley and Sons.

Dupont, S., H. Bourlard, O. Deroo, V. Fontaine, and J.-M. Boite (1997). Hybrid hmm/ann systems for training independent tasks: Experiments on 'phonebook' and related improvements. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Munich, Germany, pp. 1767–1770.

Eatock, J. P. and J. S. Mason (1994). A quantitative assessment of the relative speaker discriminant properties of phonemes. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Volume 1, pp. 133–136.

Egan, J. (1975). *Signal detection theory and ROC analysis*. Academic Press.

Farrell, K. A., R. Mammone, and K. Assaleh (1994). Speaker recognition using neural networks and conventional classifiers. *IEEE Transactions on Speech and Audio Processing 2*(1), 194–205.

Floch, J.-L. L., C. Montacié, and M.-J. Caraty (1995, September). Speaker recognition experiments on ntimit database. In *Eurospeech*, Volume 1, Madrid, Spain, pp. 379–382.

Fontaine, V., J. Hennebert, and H. Leich (1994). Influence of vector quantization on isolated word recognition. In *Proceedings of Eusipco*, Edinburgh, pp. 115–118.

Gersho, A. and R. Gray (1992). *Vector Quantization and Signal Compression*. Kluwer Academic Publishers.

Gravier, G. and G. Chollet (1998). Comparison of normalization techniques for speaker verification. In *Speaker Recognition and its Commercial and Forensic Applications, RLA2C*, Avignon, France, pp. 97–100.

Gross, N., P. Judge, O. Port, and S. Wildstrom (1998, February 23). Let's talk. *Business Week*, 45–56.

Gupta, S. K., F. Soong, and R. Haimi-Cohen (1996). High accuracy connected digit recognition for mobile applications. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Atlanta, Georgia, pp. 57–60.

Hanson, B. A. and T. H. Applebaum (1990, April). Robust speaker-independent word recognition using static, dynamic and acceleration features: experiments with lombard and noisy speech. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Albuquerque, New Mexico, pp. 857–860.

Hattori, H. (1994, April). Text-independent speaker verification using neural networks. In *Workshop on Automatic Speaker Recognition and Verification*, Martigny, Switzerland, pp. 103–106.

Hennebert, J. (1998). Towards data-driven non-linear feature extraction. In *ICSLP*, Sidney.

Hennebert, J., M. Hasler, and H. Dedieu (1994). Neural networks in speech recognition. *Invited lecture, Micro Computer School*, 23–40.

Hennebert, J. and D. Petrovska (1996). Post: Parallel object-oriented speech toolkit. In *ICSLP 96*, Philadelphia, pp. 1966–1969.

Hennebert, J. and D. Petrovska (1998). Phoneme based text-prompted speaker verification with multilayer perceptrons. In *RLA2C 98*, Avignon, France, pp. 55–58.

Hennebert, J., C. Ris, H. Bourlard, S. Renals, and N. Morgan (1997). Estimation of global posteriors and forward-backward training of hybrid hmm/ann systems. In *Eurospeech*, Rhodes, Greece, pp. 1951–1954.

Hermansky, H. (1991). Compensation for the effect of the communication channel in auditory-like analysis of speech. In *Eurospeech*.

Hermansky, H. and N. Malayath (1998). Speaker verification using speaker-specific mappings. In *RLA2C*, Avignon, France, pp. 111–114.

Hermansky, H. and N. Morgan (1994, October). Rasta processing of speech. *IEEE Transactions on Speech and Audio Processing, special issue on Robust Speech Recognition 2*(4), 578–589.

Hertz, J., A. Krogh, and R. G. Palmer (1991). *Introduction to the theory of Neural Computation*. Santa Fe Institute Studies in the Sciences of Complexity. Addison Wesley.

Himarnnet (1995, September). Final report of the himarnnet esprit project.

Homayounpour, M. and G. Chollet (1994, April). A comparison of some relevant parametric representations for speaker verification. In *ESCA workshop on automatic speaker recognition, identification and verification*, Martigny, Switzerland, pp. 185–192.

Huang, Z. and A. Kuh (1992, November). A combined self-organizing feature map and multilayer perceptron for isolated word recognition. *IEEE Transactions on Signal Processing 40*(11), 2651–2657.

IEEE-TSAP (1994, January). Special issue on neural networks for speech. *IEEE Transactions on Speech and Audio Processing 2*(1).

Iso, K. I. (1993). Speech recognition using dynamical model of speech production. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Minneapolis, pp. 283–286.

Jelinek, F. (1969). Fast sequential decoding algorithm using a stack. *IBM Journal of Research and Development 13*, 675–685.

Juang, B.-H., L. R. Rabiner, and J. G. Wilpon (1987). On the use of bandpass filtering in speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 947–954.

Kangas, J., K. Torkkola, and M. Kokkonen (1992). Using soms as feature extractors for speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 341–344.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics 43*, 59–69.

Kohonen, T. (1990, September). The self-organizing map. *Proceedings of the IEEE 78*(9), 1464–1477.

Kohonen, T., G. Brna, and R. Chrisley (1988). Statistical pattern recognition with neural networks: Benchmarking studies. In *Proceedings of the second IEEE International Conference on Neural Networks*, Volume 1, San Diego, pp. 61–68.

Konig, Y., L. Heck, M. Weintraub, and K. Sonmez (1998). Nonlinear discriminant feature extraction for robust text-independent speaker recognition. In *RLA2C*, Avignon, France, pp. 72–75.

Lang, J., A. Waibel, and G. E. Hinton (1992). *A Time-Delay Neural Network Architecture for Isolated Word Recognition*, Chapter 0, pp. 388–408. Artificial Neural Networks, Paradigms, Applications and Hardware Implementations. IEEE Press.

Lee, C.-H. (1997a). A unified statistical hypothesis testing approach to speaker verification and verbal information verification. In *Speech Technology in the Public Telephone Network, Where Are We Today ?*, Rhodes, pp. 63–72.

Lee, C.-H., E. Giachin, L. Rabiner, R. Pieraccini, and A. E. Rosenberg (1992, April). Improved acoustic modeling for large vocabulary continuous speech recognition. *Computer Speech and Language 6*(2), 103–127.

Lee, L.-S. (1997b, July). Voice dictation of mandarin chinese. *IEEE Signal Processing Magazine 14*(4), 63–101.

Lennig, e. a. (1992, October). Automated bilingual directory assistance trial in bell canada. In *Proceedings of the first IEEE workshop on Interactive Voice Technology for Telecommunications Applications*, Piscataway, N. J.

Levin, E. (1990). Word recognition using hidden control neural architecture. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Albuquerque, pp. 433–436.

Levin, E. (1993). Hidden control neural architecture modeling of nonlinear varying systems and its applications. *IEEE Transactions on Neural Networks 4*(1), 109–116.

Lippmann (1989). Review of neural networks for speech recognition. *Neural Computation 1*(1), 1–38.

Magrin-Chagnolleau, I., J. Wilke, and F. Bimbot (1996, May). A further investigation on ar-vector models for text-independent speaker identification. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Atlanta, GA, pp. 401–404.

Mantysalo, J., K. Torkkola, and T. Kohonen (1992). Lvq-based speech recognition with high-dimensional context vectors. In *Proceedings ICSLP*, Banff, Alberta, Canada, pp. 539–542.

Martin, A., G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki (1997). The det curve in assesment of detection task performance. In *Eurospeech 1997*, Rhodes, Greece, pp. 1895–1898.

Martinelli (1994, March). Hidden control neural network. *IEEE Transactions on Circuits and Systems-II : Analog and Signal Processing 41*(3), 245–247.

Matsui, R. and S. Furui (1993, April). Concatenated phoneme models for text-variable speaker recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Volume 2, Minneapolis, pp. 391–394.

Montacie, C., P. Deleglise, F. Bimbot, and M. J. Caraty (1992). Cinematic techniques for speech processing : temporal decomposition and multivariate linear prediction. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Volume 1, San-Francisco, pp. 153–156.

Morgan, N. and H. Bourlard (1995, May). Neural networks for statistical recognition of continuous speech. *Proceeding of the IEEE 83*(5), 742–770.

Mozer, M. (1996, August). Neural-network speech processing for toys and consumer electronics. *IEEE expert 11*(1), 4–5.

Naik, J. M. (1990, January). Speaker verification: A tutorial. *IEEE Communications Magazine 28*(1), 42–48.

Naik, J. M. and D. M. Lubenskt (1994). A hybrid hmm-mlp speaker verification algorithm for telephone speech. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 153–156.

Ney, H. (1990, April). Acoustic-phonetic modeling using continuous mixture densities for the 991-word darpa speech recognition task. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Albuquerque, New Mexico, pp. 713–716.

Ney, H., D. Mergel, A. Noll, and A. Paeseler (1987, April). A data-driven organization of the dynamic programming beam search for continuous speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 833–836.

Nilsson, N. (1980). *Principles of Artificial Intelligence*. Palo Alto, CA: Tioga Publishing Co.

Nolan, F. (1983). *The Phonetic Bases of Speaker Recognition*. Cambridge University Press.

Oglesby, J. and J. S. Mason (1990). Optimization of neural models for speaker identification. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 261–264.

Olsen, J. (1997). A two-stage procedure for phone based speaker verification. In G. B. J. Bigün, G. Chollet (Ed.), *First International Conference on Audio and Video Based Biometric Person Authentication (AVBPA)*, Crans, Switzerland, pp. 219–226. Springer Verlag: Lecture Notes in computer Science 1206.

O'Shaugnessy, D. (1986, October). Speaker recognition. *IEEE Trans. Acoust., Speech, Signal Processing*, 4–17.

Petrovska, D., J. Cernocky, J. Hennebert, and G. Chollet (1998). Text-independent speaker verification using automatically labelled acoustic segments. *Accepted for publication in Journal for dialectology and linguistics (IPS supplement)*.

Petrovska, D. and J. Hennebert (1998). Text-prompted speaker verification experiments with phoneme specific mlp's. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Seattle, pp. 777–780.

Petrovska, D., J. Hennebert, J. Cernocky, and G. Chollet (1998). Text-independent speaker verification using automatically labelled acoustic segments. In *ICSLP*, Sidney.

Petrovska, D., J. Hennebert, D. Genoud, and G. Chollet (1996, November). Semi-automatic hmm-based annotation of the polycost database. In *Proceedings of a COST 250 workshop on Application of Speaker Recognition Techniques in Telephony*, Vigo, Spain, pp. 23–26.

Petrovska, D., J. Hennebert, H. Melin, and D. Genoud (1998). Polycost : A telephone-speech database for speaker recognition. In *RLA2C*, Avignon, France, pp. 211–214.

Pican, N., F. Alexandre, and P. Bresson (1996, February). Artificial neural networks for the presetting of a steel temper mill. *IEEE Expert Intelligent Systems and Their Applications 11*(1), 22–27.

Picone, J. (1993, September). Signal modeling techniques in speech recognition. *Proceedings of the IEEE 81*(9), 1214–1247.

Pirelli, J. et al. (1995). Phonebook: A phonetically-rich isolated word telephone speech database. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Detroit.

Rabiner, L. (1989, February). Tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*(2), 257–285.

Rabiner, L. and B.-H. Juang (1993). *Fundamentals of Speech Recognition*. Prentice Hall.

Rabiner, L., J. Wilpon, and F. Soong (1989). High performance connected digit recognition using hidden markov models. *IEEE Transactions Acoustics Speech Signal Processing 37*, 1214–1225.

Ramasubramanian, V. and K. Paliwal (1988). An optimized k-d tree for fast vector quantization of speech. In *Eusipco*, pp. 875–878.

Renals, S., N. Morgan, H. Bourlard, M.Cohen, and H. Franco (1994). Connectionist probability estimators in hmm speech recognition. *IEEE Trans. on Speech and Audio Processing 2*(1), 161–174.

Reynolds, D. (1995). Automatic speaker recognition using gaussian mixture speaker models. *The Lincoln Laboratory Journal 8*(2), 173–191.

Reynolds, D. (1997). Comparison of background normalization methods for text-independent speaker verification. In *Eurospeech*, pp. 1895–1898.

Rezgui, A. and N. Tepedelenlioglu (1990). The effect of the slope of the activation function on the back-propagation algorithm. In *Proceedings International Joint Conference Neural Networks*, Volume 1, Washington, pp. 707–710.

Robinson, T. and F. Fallside (1991). A recurrent error propagation network speech recognition system. *Computer Speech and Language 5*, 259–274.

Robinson, T., M. Hochberg, and S. Renals (1996). *Automatic Speech and Speaker Recognition – Advanced Topics*, Chapter The use of recurrent networks in continuous speech recognition, pp. 233–258. Kluwer Academic Publishers.

Roe, B. and J. G. Wilpon (1993, November). Whither speech recognition: The next 25 years. *IEEE Communications Magazine 31*(11), 54–62.

Rosenberg, A. E. and F. K. Soong (1992). Recent research in automatic speaker recognition. In S. Furui and M. M. Sondhi (Eds.), *Advances in Speech Signal Processing*, pp. 701–738. New York: Marcel Dekker.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland (Eds.), *Parallel Distributed Processing. Exploration in the Microstructure of Cognition*, Volume 1. Massachusetts Institute of Technology Press.

Schlang, M., T. Poppe, and O. Gramchow (1996, August). Neural network for steel manufacturing. *IEEE Expert Intelligent Systems and Their Applications 11*(4), 8–9.

Senior, A. and T. Robinson (1996, December). Forward-backward retraining of recurrent neural networks. In *Advances in Neural Information Processing Systems 8*, Denver, Colorado, pp. 743–749. MIT Press.

Tesauro, G., J. Kephart, and G. Sorkin (1996, August). Neural networks for computer virus recognition. *IEEE Expert Intelligent Systems and Their Applications 11*(4), 5–6.

Torkkola, K. et al. (1991, June). Status report of the finnish phonetic typewriter project. In *Proceedings of the Internaltional conference on Artificial Neural Networks*, Espco, Finland, pp. 771–776.

Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang (1989, March). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Siganl Processing 37*(3), 329–339.

Waibel and Lee (Eds.) (1990). *Readings in Speech Recognition*, Volume 1. San Mateo, CA: Morgan Kaufman Publisher, Inc.

Y. Linde, A. Buzo, A. G. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communication 28*(1), 84–95.

Yaeger, L. (1996, August). Neural networks provide robust character recognition for newton pda. *IEEE Expert Intelligent Systems and Their Applications 11*(4), 10–12.

Yan, Y., M. Fanty, and R. Cole (1997). Speech recognition using neural networks with forward backward probability generated targets. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Munich, Germany, pp. 3241–3244.

Zhao, Z. and C. Rowden (1992, December). Use of kohonen self-organiaing feature maps for hmm parameter smoothing in speech recognition. *IEE Proceedings-F 139*(6), 385–390.

Index

Activation function, 30
ALISP, 85, 129
ANN, 129
Artificial Neural Networks, 6
ASkR, 5, 129
ASkR Applications, 6
ASR, 2, 129
ASR applications, 4
ASSP, 129
Autocorrelation analysis, 18
Automatic Speaker Recognition, 5
Automatic Speech Recognition, 2

Backpropagation, 111
Bayes theory, 114
Beam search, 27

Centroid, 40
CIRC, 129
Client, 75
CMS, 129
Coarticulation, 129
Codebook, 40
Codeword, 40
Continuous speech recognition, 129
Cross-validation, 15

Database
    HER, 107
    HIM, 107
    Phonebook, 105
    Switchboard, 107
Delta coefficients, 19
Delta-delta coefficients, 19
DET, 129
DET curve, 79
Difficulties of the speech signal, 12
Discrete HMM, 40
Discrete HMMs, 29
Discriminative training, 34
Distorsion measure, 40
DTMF, 129
Duration modeling, 29

EER, 79, 129
Equal error rate, 79
ETSI, 129

False acceptance, 79
False rejection, 79
Feature extraction, 15
Finnish phonetic typewriter, 48
Forward probability, 24
Forward-Backward recursion, 24
Frame blocking, 17
Frame windowing, 17
Front-end, 15

Gaussian mixtures modeling, 77
GBSI EER, 79, 129
Gender balanced sex-independent EER, 79
General Lloyd algorithm, 41
Global posterior estimation, 60
GMM, 77, 129

Hamming window, 18
HCNN, 119, 129
HER, 129
HER Database, 107
Hidden Control Neural Networks, 119
HIM, 129
HIM Database, 107
HMM, 129
    Discrete, 29
    discrete pdf, 44
    Gaussian mixtures, 27
    Probability density function, 27
    Reestimation, 26
        Discrete, 44
Hnorm, 80, 129
Hybrid systems, 56
    Forward-backward training, 62
    new formulation, 59
    REMAP training, 63
    Viterbi trained, 56

ICASSP, 129
ICSLP, 129
IEEE, 129
Impostors, 75
Inversion, 113
ISO, 129

Isolated word recognition, 129

K-means, 41
Kohonen maps, 42
Kolmogorov's theorem, 115

Large vocabulary systems, 130
LBG, 130
Liftering, 18
Linear predictive coding, 18
Lloyd algorithm, 41
Log-energy, 19
Log-likelihood ratios, 78
Lombart effect, 19, 130
LPC, 18, 130
LPC-cepstral analysis, 17
LVCSR, 130
LVQ, 48, 130

Maximum A Posteriori criterion, 21
Minimum duration modeling, 29
MLP, 29, 130
  1-fromK training, 34
  Activation function, 30
  Backpropagation, 31, 111
  Bayes theory, 114
  Bias, 31
  Inversion, 113
  Isolated word recognition, 117
  Kolmogorov's theorem, 115
  MSE, 33
  Posterior probability, 34
  Propagation, 31
  Sensitivity, 112
  Sigmoid, 30
  Sum-of-squares error, 33
  Tanh, 30
  Time-delay, 118
MLP:Discriminative training, 34
MSE, 33, 130
Multilayer perceptrons, 29

NIST, 130
NIST-NSA Evaluation, 88, 107
NLP, 130
NSA, 130

Over-training, 15

Pattern recognition, 14
Pdf, 130
Perplexity, 130
Phone, 130
Phonebook Database, 105
Phoneme, 130
Post-processing, 15
Pre-emphasis, 17
Pre-processing, 15
Probability density function, 27

Reestimation, 26
Reinforcement learning, 15
ROC, 130
ROC curve, 79

Segmentation, 27
Self-organizing maps of Kohonen, 42
Sensitivity, 112
SI, 130
Sigmoid, 30
SOM, 42, 130
Speaker identification, 75
Speaker verification, 75
  ALISP, 85
  Baseline system, 76
  DET, 79
  EER, 79
  False acceptance, 79
  False rejection, 79
  GBSI EER, 79
  GMM, 77
  Hnorm, 80
  LVCSR, 85
  MLP modeling, 78
  ROC, 79
  Score recombination, 88
  Segmental, 75
  Temporal decomposition, 86
  Text-dependent, 75
  Text-independent, 75
  Thresholding, 78
  Znorm, 80
Speaker-dependent, 3
Speaker-independent, 3
Speaking rate, 3
Stack decoding, 24
STRUT, 130
Sum-of-squares error, 33
Supervised learning, 7, 15
SV, 130
Switchboard, 88
Switchboard database, 107

Tanh, 30
TD, 130
TDNN, 118, 130
Temporal decomposition, 86
Test set, 15
Thresholding, 78

Time-delay ANN, 118
Touchtone, 130
Training set, 15

Unsupervised learning, 7

Vector quantization, 40
VQ, 130
    distorsion, 40
    Fast search, 47
    General Lloyd algorithm, 41
    K-means, 41
    Learning Vector Quantization, 48
    Lloyd algorithm, 41
    nearest neighbour quantizers, 41
    Voronoi regions, 41

Znorm, 80, 130

Curriculum Vitæ

Jean Hennebert was born on December 29, 1970, in Mons, Belgium. He received the electrical engineering degree ("Ingénieur Civil Electricien - Specialization in Telecommunications, mention: grande distinction") from the Faculté Polytechnique de Mons (FPMS), Belgium, in June 1993.

He joined the Circuits and Systems group of the Ecole Polytechnique Fédérale de Lausanne (EPFL) in 1993, first as a student in the framework of an Erasmus student exchange, then as a research assistant. At EPFL, he took part in the Himarnnet Esprit project dedicated to the comparison of state-of-the art speech recognition systems using neural networks. He then took part in the COST 250 project dedicated to speaker verification in telephony and turned his research in neural networks based speaker verification systems. In 1998, he took part in the speaker verification contest organized by the National Institute for Standards and Technolgy (NIST) as part of the european ELISA consortium. He started working towards the PhD degree since 1995 in the field of speech and speaker recognition with Hidden Markov Models and Artificial Neural Networks. While working on his thesis, he was also a visiting researcher for 6 months at the International Computer Science Institute, Berkeley.

His research interests are in the fields of neural networks, speech signal processing, pattern recognition, Hidden Markov Models, speech and speaker recognition. He is member of the European Speech Communication Association (ESCA), of the Société Française d'Acoustique (SFA) and is a reviewer for the IEEE Transactions on Circuits and Systems and various conferences.

List of publications

Fontaine, V., J. Hennebert, and H. Leich (1994). Influence of vector quantization on isolated word recognition. In *Proceedings of Eusipco*, Edinburgh, pp. 115–118.

Hennebert, J. (1998). Towards data-driven non-linear feature extraction. In *ICSLP*, Sidney.

Hennebert, J., M. Hasler, and H. Dedieu (1994). Neural networks in speech recognition. *Invited lecture, Micro Computer School*, 23–40.

Hennebert, J. and D. Petrovska (1996). Post: Parallel object-oriented speech toolkit. In *ICSLP 96*, Philadelphia, pp. 1966–1969.

Hennebert, J. and D. Petrovska (1998). Phoneme based text-prompted speaker verification with multi-layer perceptrons. In *RLA2C 98*, Avignon, France, pp. 55–58.

Hennebert, J., C. Ris, H. Bourlard, S. Renals, and N. Morgan (1997). Estimation of global posteriors and forward-backward training of hybrid hmm/ann systems. In *Eurospeech*, Rhodes, Greece, pp. 1951–1954.

Petrovska, D., J. Cernocky, J. Hennebert, and G. Chollet (1998). Text-independent speaker verification using automatically labelled acoustic segments. *Accepted for publication in Journal for dialectology and linguistics (IPS supplement)*.

Petrovska, D. and J. Hennebert (1998). Text-prompted speaker verification experiments with phoneme specific mlp's. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Seattle, pp. 777–780.

Petrovska, D., J. Hennebert, J. Cernocky, and G. Chollet (1998). Text-independent speaker verification using automatically labelled acoustic segments. In *ICSLP*, Sidney.

Petrovska, D., J. Hennebert, D. Genoud, and G. Chollet (1996, November). Semi-automatic hmm-based annotation of the polycost database. In *Proceedings of a COST 250 workshop on Application of Speaker Recognition Techniques in Telephony*, Vigo, Spain, pp. 23–26.

Petrovska, D., J. Hennebert, H. Melin, and D. Genoud (1998). Polycost : A telephone-speech database for speaker recognition. In *RLA2C*, Avignon, France, pp. 211–214.