Acoustic Level
Speech Recognition

Adrian Edward Lucas

Thesis submitted for the degree of Doctor of Philosophy in the Department of Electronic & Electrical Engineering, University of Surrey, Guildford, Surrey, U.K.

October 1991
Abstract

A number of techniques have been developed over the last forty years which attempt to solve the problem of recognizing human speech by machine. Although the general problem of unconstrained, speaker independent connected speech recognition is still not solved, some of the methods have demonstrated varying degrees of success on a number of constrained speech recognition tasks.

Human speech communication is considered to take place on a number of levels from the acoustic signal through to higher linguistic and semantic levels. At the acoustic level, the recognition process can be divided into time-alignment (the removal of global and local timing differences between the unknown input speech and the stored reference templates) and reference template matching. Little attention seems to have been given to the effective use of acoustic level contextual information to improve the performance of these tasks.

In this thesis, a new template matching scheme is developed which addresses this issue and successfully allows the utilization of acoustic level context. The method, based on Bayesian decision theory, is a dynamic time warping approach which incorporates statistical dependencies in matching errors between frames along the entire length of the reference template. In addition, the method includes a speaker compensation technique operating simultaneously.

Implementation is carried out using the highly efficient branch and bound algorithm. Speech model storage requirements are quite small as a result of an elegant feature of the recursive matching criterion. Furthermore, a novel method for inferencing the special speech models is introduced.

The new method is tested on data drawn from nearly 8000 utterances of the 26 letters of the British English Alphabet spoken by 104 speakers, split almost equally between male and female speakers. Experiments show that the new approach is a powerful acoustic level speech recognizer achieving up to 34% better recognition performance when compared with a conventional method based on the dynamic programming algorithm.
Acknowledgments

I am indebted to many friends and colleagues who have provided me with moral and technical support which has enabled me complete the doctorate.

Firstly, I offer my sincere thanks to my supervisor, Professor Josef Kittler, who has made the whole exercise possible by providing theoretical initiatives and technical guidance of the highest quality. For enabling me to make sense of my results, and for providing the recording of his voice in Chapter 3, I thank Dr. Ted Chilton. Thank you to Chris James for spending many frustrating hours recording my voice for the analysis in Chapter 2. I am truly grateful to Dr. Geoff Nicholls and Dr. John Princen for providing inspiration when the going was tough, and to Isaac Ng, who not only applied his extensive software skills to make this document possible, but gave continuous help, support and friendship. Thanks also to Chris Smith for reading the first draft (sorry about the maths, Chris!), to Tony Carraro for sharing his knowledge on psycholinguistics with me and to Tania Lomas for her enthusiasm and support. For additional financial assistance, I am grateful to the Institution of Electrical Engineers for honouring me with The Leslie H. Paddle Scholarship Award.

The biggest thank you goes to my mother and to my aunt who, throughout my life and against all odds, have given me love, care and support without which I would have achieved nothing. It is to them that I dedicate this work.
## Contents

Abstract

Acknowledgments

1 Introduction
   1.1 Speech: a Natural Man-Machine Interface .......................... 1
   1.2 Speech Recognition in Operation .................................. 2
   1.3 Scope of the Thesis ......................................... 4
   1.4 Aims ........................................ 5
   1.5 Achievements ........................................ 6
   1.6 Layout of the Thesis ........................................ 7
   References ........................................ 9

2 Issues in Speech Recognition
   2.1 Introduction ................................... 10
   2.2 Overview ..................................... 11
   2.3 Inherent Factors in Speech Communication ................. 11
      2.3.1 Coarticulation ........................... 11
      2.3.2 Accent and Dialect .......................... 12
      2.3.3 Prosody .................................. 13
      2.3.4 Occasion & Time of Day ..................... 14
      2.3.5 Context .................................. 14
      2.3.6 Speaker Sex and Age .......................... 15
      2.3.7 Speech Rate ................................ 15
      2.3.8 Noise & Environment .......................... 15
      2.3.9 Word Boundaries ........................... 16
      2.3.10 Homophones, Syntactic & Semantic Ambiguity .... 18
      2.3.11 Common Solutions to Speech Variabilities .... 18
   2.4 Practical Problems ..................................... 19
      2.4.1 Size of the Training Database ....................... 19
      2.4.2 Choice of Basic Recognition Unit .................... 20
      2.4.3 End-Pointing .................................. 23
   2.5 Constraining the Problem ................................... 24
      2.5.1 Isolated and Continuous Speech Recognition .......... 24
      2.5.2 Speaker Independent & Speaker Dependent Systems .... 25
      2.5.3 Vocabulary Size ................................ 25
      2.5.4 Word Spotting ................................ 26
   2.6 Summary ..................................... 26
   References ........................................ 27
3 Digital Speech Analysis

3.1 Introduction .............................................. 29
3.2 Source-Filter Model .................................. 29
3.3 Digital Speech Analysis and Windowing ............... 31
3.4 Parametric Representation of the Speech Signal ............... 36
  3.4.1 Mel-Frequency Cepstral Coefficients ................. 36
3.5 Pattern Analysis and Feature Extraction ................. 37
3.6 Summary ..................................... 39
References ...................................... 40

4 A Review of Approaches to Automatic Speech Recognition 41

4.1 Introduction ............................... 41
4.2 Early Attempts .................. 41
4.3 Template Matching and Pattern Recognition ................. 42
4.4 Incorporating Linguistic Knowledge ............... 43
4.5 Dynamic Time Alignment ............................ 44
4.6 Hidden Markov Models ...................... 45
4.7 Artificial Neural Networks ...................... 46
4.8 Automatic Speech Recognition Systems .............. 50
4.9 Non-Linear Time Alignment Using Dynamic Programming 52
  4.9.1 Non-Linear Time Alignment ............... 52
  4.9.2 Introduction to Dynamic Programming .............. 53
  4.9.3 Dynamic Programming Applied to Isolated-Word Recognition 55
  4.9.4 Constraints ...................................... 59
  4.9.5 Dynamic Programming Applied to Connected Speech Recognition 63
4.10 Summary ..................................... 64
References ....................................... 64

5 The Matching Criterion 71

5.1 Introduction ................................... 71
5.2 A Framework for the Solution ................ 72
5.3 Problem Formulation .............................. 73
  5.3.1 The Pattern Matching Problem .......... 74
  5.3.2 Time Alignment Transformation .......... 76
5.4 Mean Level Compensation ...................... 79
  5.4.1 The Need For Mean Level Compensation .......... 79
  5.4.2 Compensation Scheme ......................... 81
5.5 Recursive Form of the Criterion Function .......... 85
5.6 Speech Model Storage ...................... 88
5.7 Summary ..................................... 89
References ....................................... 90

6 Implementation 91

6.1 Introduction ................................... 91
6.2 Why Not Use DP? .......................... 92
6.3 The Search Problem ....................... 93
  6.3.1 Computational Complexity of Exhaustive Search .............. 93
6.4 An Introduction to the Branch and Bound Algorithm .............. 95
6.5 Control of the Search ....................... 97
  6.5.1 Global Constraints ......................... 97
  6.5.2 Local Constraints and Thresholds .......... 99
6.6 Details of the Branch and Bound Implementation .......... 102
  6.6.1 Formal Algorithm Statement .......... 105
6.7 Development In Hardware .................. 107

References ....................................... 107
Chapter 1

Introduction

1.1 Speech: a Natural Man-Machine Interface

Machines play an important role in our society. Whether they are used to enhance our quality of life, or enable a task to be performed more quickly and reliably, it is hard to ignore their presence. Throughout an average day, we use many machines. Some we find easy to use, either by intuition or by learning the skills needed to operate them. Sometimes, however, it is difficult to use a machine as effectively as we would like. It may possess significant capabilities, but we are unable to utilize them easily or fully.

Problems with the use of machines can often be traced to the interaction between the human and the machine - commonly referred to as man-machine interaction, the man-machine interface or MMI. Much effort has been directed towards the improvement of the man-machine interface and the results are most noticeable in the areas of computing (high resolution graphical window environments, icons, mouse control, etc.) and consumer goods (bar-code scanners for video recorders, etc.), for example. To enable us to interact with machines more naturally and effectively, the apparatus should ideally be able to respond to our natural forms of communication.

Human communication is complex and varied. However, speech is arguably one of the most common methods for communicating our ideas. It follows, therefore, that automatic speech recognition (ASR) would be a very useful facility for some types of machine to possess.

More importantly, perhaps, there are a number of trivial and repetitive speech-based tasks currently performed by humans which could be adequately automated using ASR (telephone directory enquiries, for example). In such cases, the human is simply taking on the role of a speech-to-text translator. Ironically, stating the problem in this manner encapsulates the difficulty of the task perfectly; for machines, the role of speech-to-text translator is not such a simple one to play.
1.2 Speech Recognition in Operation

There are probably as many applications for automatic speech recognition as there are types of machine. Furthermore, speech recognition is merely a subset of the much wider discipline - human natural language understanding and synthesis. Within this category, we can place such topics as text-to-speech conversion/speech synthesis, automatic language translation and speaker recognition. In reality, approaches to these problems often differ quite considerably. Nevertheless, the applications to which such techniques can be put are largely interlinked. This idea is illustrated in Figure 1.1. A brief thought about the tasks to which Figure 1.1 refers will reassure us that there exists a motivation to achieve the goal of unconstrained automatic speech recognition by machine. In particular, there are a number of specific motives for pursuing this goal.
Motivation for Speech Recognition

- it is a natural form of human communication
- high speed communication - 120-250 words per minute
- leaves hands and eyes available for other tasks
- needs no new skill training for most people
- it is omnidirectional and therefore permits unconstrained movement
- permits operation in darkness or around obstacles
- facilitates remote access by telephone (c.f. dual-tone multi-frequency (DTMF) dialing (tone-dialing), facsimile, modems)
- requires minimal keyboard, key-pad or control panel space
- allows use by blind persons
- enables multi-modal communication (e.g. combine speech with a tactile input device such as a mouse).

The list, although not exhaustive, does highlight some desirable advantages of a man-machine interface possessing speech recognition capabilities.

Perhaps we should not get carried away with these ideas. We should not expect to be able to apply the hypothetical faultless speech recognizer at all types of machine application and expect to achieve success every time. Some striking examples of tasks where a speech interface would not be of benefit are:

- positional control - almost always better to use tactile devices for tasks such as drawing and general artwork
- some types of text manipulation such as “cutting-and-pasting” and copying
- tasks requiring privacy or quiet.

Attempts to apply current speech recognition capabilities to real-life tasks (such as computer windowing management [2]) have raised these and other problems.
1.3 Scope of the Thesis

Typical of virtually all man-machine studies, speech recognition is vastly inter-disciplinary. It draws on knowledge from linguistics, phonetics, phonology, computer science, mathematics, electrical engineering, artificial intelligence, psycholinguistics and others. Any attempt to quantify and categorize the process of recognizing speech risks making unqualified assumptions about the task. While such assumptions are acceptable for solving a given problem, we ought to be aware that better solutions to the general task may exist by analyzing the basic processes in a different way. Given current theories, however, the general synopsis of the basic elements that comprise a typical speech recognition system is presented in Figure 1.2.

![Figure 1.2: A typical recognition system will comprise a number of specific functions leading from the time-domain speech signal to the output — a “word-level” decision. The system may contain top-down constraints utilizing linguistic knowledge.](image)

It has been widely accepted that the automatic speech recognition procedure can be categorized into recognizable levels, sometimes overlapping and sometimes with interconnections. Putting aside the question of what these levels really are, almost all practical speech recognition systems take some time-domain speech signal (the output from a microphone) as their input and attempt to produce an intelligible transcription
of this signal at the output. At an intermediate stage, a pattern matching procedure takes
place which compares the incoming utterance with a set of standard reference template
patterns (Figure 1.2) to produce "goodness-of-fit" scores. The reference templates have
been designed and chosen to be good approximations to the average pronunciations
of each word in the recognizers's vocabulary. For a more thorough discussion of the
issues in speech recognition system design, the reader is referred to the collection
of established and authoritative texts and reviews which have emerged over the last

The scope of this thesis is confined to the pattern matching stage of the recogni-
tion process. It does not aim to explicitly solve the problems associated with reference
template design, top-down constraint strategies nor to perform experiments to deter-
mine the best basic recognition unit to use. However, some of these issues will be
discussed in relation to the work.

1.4 Aims

Since research into speech recognition began in the early 1950's, a number of approaches
to solving the acoustic-level recognition task have been developed. Of these, the most
famous and popular methods are/have been

- dynamic time warping (DTW),
- hidden Markov models (HMMs),
- artificial neural networks (ANNs)/connectionist methods.

Today, we see the acoustic level recognition task as consisting primarily of two
processes: time-alignment and reference template matching. Time-alignment aims to take
out local and global timing differences, between the input and the template, and tem-
plate matching compares the input speech with each of the stored reference speech
models. DTW and HMMs address both processes simultaneously while ANNs are
often applied to pre-warped speech (see Appendix A) and thus address the template
matching task solely.\(^1\)

Traditionally, the DTW approach rests the time-alignment process upon criteria
acquired in a single frame of speech. Determining the optimal match between a refer-
ence template and an utterance in this way presupposes that the objective function,
or matching criterion function, is separable. That is, the matching criterion can be
expressed in terms of additive components, each of which is a function of a single

\(^1\)There are exceptions to this, notably time-delay neural networks (TDNNs) — page 49.
speech frame. This assumes that errors between the two waveforms being matched are statistically independent with respect to time. Furthermore, conventional objective functions for DTW do not incorporate a speaker-to-speaker compensation mechanism.

Thus, the hypothesis put forward in this thesis is that

"better acoustic-level speech recognition performance can be achieved by employing a non-linear time-alignment template matching scheme which utilizes statistical dependencies of the matching errors between speech frames in the time domain and incorporates a speaker compensation technique."

Using this hypothesis as a basis, the aims of the thesis are to:

1. Develop a template matching criterion which exploits statistical dependence and incorporates a speaker compensation technique;

2. Develop a recursive form of the above matching criterion which facilitates sequential computation;

3. Devise a search strategy which performs a minimization of this criterion;

4. Formulate a strategy for inferencing the reference templates (speech models) from training data;

5. Create a set of experiments which test the validity of the stated hypothesis and yield a sound conclusion;

6. Suggest areas of the work which need further investigation and/or development.

1.5 Achievements

The ideas under investigation in this thesis are important both in terms of a better understanding of acoustic-level speech signal properties and for the improvement in speech recognition performance which has been obtained by utilizing these properties. Until now, there seems to have been little reported success of incorporating long-range acoustic-level context into the DTW time-alignment process at the template matching stage. The main achievements of the work can be summarized as follows:

1. A recursive matching criterion has been developed which incorporates frame-to-frame statistical dependencies and simultaneously performs speaker compensation by removing the mean frequency level of the utterance as recognition progresses. The resulting Mahalanobis distance-based criterion function effectively
exploits the correlations between frames along the entire length of the reference template. Template storage is kept to a minimum through an elegant feature of the matching criterion;

2. A search strategy, based upon the branch and bound algorithm, has been developed which optimizes the criterion function and guarantees to find the optimal time-alignment transformation between the input and the reference model;

3. A number of strategies have been suggested which enable flexible control of the search to be achieved including the control of the speed/recognition performance characteristics;

4. A iterative method for creating templates was introduced which includes a novel technique for reconstructing rank-deficient covariance matrices;

5. Correlations between frames in the speech waveform were shown to exist through observations of the class covariance matrices. Both vowel (“aeiou”) and consonant (“bdgptv”) sounds exhibited strong, and broadly similar, correlations.

6. The new matching criterion was shown to achieve significantly better classification performance when compared with a conventional dynamic programming approach tested on spoken letter (“bdg” and “ptv”) speaker independent recognition tasks. Performance for vowels (“aeiou”) was only marginally improved which is believed to be mainly due to the steady state nature of these sounds.

1.6 Layout of the Thesis

The thesis contains an overview of the basic elements and issues in speech recognition in Chapters 2 – 4, and the original research work — a detailed description, implementation and validation of a new template matching scheme for speech recognition — in Chapters 5 – 8. The overview sections, taken together, form a self-contained study and can be read by anyone interested in gaining an introduction to speech recognition. The new work can also be read in isolation but the reader new to the field may need to pursue some of the references made to earlier sections of the thesis. There is a Glossary of terms on page 149 which the reader may find useful when a specific term needs clarifying.

The Overview: The overview begins by introducing the main Issues in Speech Recognition. We divide this subject into inherent factors in speech communication and practical
problems related to the implementation of speech recognition systems. The chapter concludes by outlining the methods of constraining the problem to real-life applications.

Then, a summary of the techniques employed in the pre-processing stage of the speech recognition system is given in Chapter 3, *Digital Speech Analysis*. This covers all the essential preliminary procedures for converting raw speech into a format commonly utilized in the speech recognition systems of today.

*A Review of Approaches to Speech Recognition* (Chapter 4) attempts to cover the major work undertaken in the field since the early 1950s and includes a catalogue of ASR systems developed and/or manufactured to date. To highlight some of the techniques which have been developed to cope with the problems of time-alignment and template matching, a more detailed study of Dynamic Time Warping/Alignment is provided. This section addresses the basic DTW concept using *dynamic programming* (DP) and its application to speech recognition for both discrete utterance and continuous speech recognition.

The New Work: The second half of the thesis opens by introducing *The Matching Criterion* used in the new template matching scheme (Chapter 5). Firstly, the limitations of conventional DP-matching schemes are exposed and a new speaker-adaptive template matching criterion is developed which incorporates acoustic-level context. A recursive form of the criterion is derived which allows sequential computation of the matching process.

In Chapter 6 — *Implementation* — we state the reasons why DP is not suitable for minimizing the new matching criterion and go on to develop the highly efficient and optimal branch and bound search algorithm for this task. Full details of applying this algorithm are described along with the local and global search constraints.

A strategy for inferencing, or building, reference speech models, or templates, from training data is described in Chapter 7. The scheme is not straightforward requiring the development of a technique for overcoming the numerical problems encountered during the inferencing process.

Experimental work and results are presented in Chapter 8, using a database of spoken letters, which validate the new speech template matching scheme. This is followed in Chapter 9 by a discussion of the material covered in the thesis, a summary of the achievements and a statement of some areas for further research.
References


Chapter 2

Issues in Speech Recognition

2.1 Introduction

Automatic speech recognition (ASR) by machine is a complex task requiring knowledge from many disciplines. Over the last forty years or so, linguists, research scientists and engineers have been developing and applying numerous techniques to the problem with varying degrees of success. Today, therefore, we have a massive store of knowledge, theories, methods and real-life products from which to form an overview of the task and its many problems.

In this chapter, a synopsis of our basic knowledge and experience relating to human and machine speech communication is presented.

Inherent and Practical Factors: Attention is focused firstly on the inherent factors in speech communication and then upon some of the practical problems in the implementation of speech recognition systems. The section covers issues such as: coarticulation, accent, dialect, prosody, occasion, context, sex, speech rate and environmental noise. Further ambiguities are mentioned including word-boundaries, homophones, and syntactic/semantic ambiguity. The practical problems addressed include training set size, choice of recognition unit and end-pointing. These problems are illustrated with excerpts of real speech in spectrogram format.

Constraining the Problem: In Section 2.5, we discuss methods of constraining the problem such that real-life solutions can be attained. In order to design a reasonably error-free speech recognition system with today's knowledge and technology, the speech technologist answers questions regarding: isolated/continuous speech recognition, speaker dependence/independence, vocabulary size and environmental restrictions.
2.2 Overview

Unconstrained automatic speech recognition (ASR) by machine is proving to be a difficult task to accomplish. Whatever the reasons are for this, at least a sizable list of identifiable issues and problems has been accumulated which are commonly believed to lay at the heart of the task. The list can be divided broadly into those factors which are inherent to speech communication and those which are practical, often allied to a particular implementation or system model.

### Issues in Speech Recognition

**Inherent:**
- Coarticulation
- Accent and Dialect
- Prosody (pitch, intensity, timing)
- Occasion & time of day
- Context
- Speaker sex and age
- Speech rate
- Noise & environment
- Word boundaries
- Homophones
- Syntactic ambiguity
- Semantic ambiguity

**Practical:**
- Size of training database
- Choice of basic recognition unit:
  - word
  - syllable
  - demi-syllable
  - allophone
  - phoneme
  - diphone
  - tri-phone/tri-gram
  - acoustic sub-word unit
- End-pointing

2.3 Inherent Factors in Speech Communication

2.3.1 Coarticulation

Coarticulation can be defined as the effect of neighbouring sounds on the position of the tongue and other articulators in the mouth and vocal tract (lips, jaw, etc). This means that a sound spoken in isolation is changed when spoken in conjunction with other sounds. Coarticulation is, therefore, a major problem for many conventional approaches, largely because they attempt to restrict the complexity of the recognition task by basing the whole process on the recognition of discrete and isolated units of speech.
Coarticulation can be defined as the effect of neighbouring sounds on the position of the tongue and other articulators in the mouth and vocal tract. In this spectrogram of the words, "Tom Burton", the phoneme /t/ appears in both words with significantly different spectra.

To see how the effect manifests itself, consider the spectrogram of the words "Tom Burton" in Figure 2.1. Although the phoneme /t/ appears in both words, its acoustic realization is different in each case depending upon its immediate phonetic environment and position within the word. In the second word, "Burton", for example, the /t/ seems to have disappeared entirely! Clearly, any scheme which attempts to match this example sequence to a single, pre-stored phoneme template for /t/ will be fraught with difficulties. Inclusion of several models for /t/ will alleviate the problem to a large extent.

2.3.2 Accent and Dialect

For a speech recognition system to be truly speaker independent (see Section 2.5.2), it should be able to cope with speakers from different geographical regions who speak the common language for which the system was designed (e.g. English, Japanese, etc.). The two major obstacles to achieving this requirement are accent and dialect, defined as follows.

ACCENT: a difference in pronunciation (or phonetic quality) between speakers; often regional. There is usually little problem understanding the meaning.

DIALECT: the use of alternative words and changes in grammar and pronunciation often used only by members of a certain region. The meaning is sometimes
difficult to understand.

If the phonetic qualities of each pronunciation of a given word can be modeled reliably and are included in the template store, then accent can be dealt with. To overcome the problem of dialect, an extended grammar and/or lexicon is needed to successfully recognize the alternative words and phrases.

2.3.3 Prosody

Linguistic information is not conveyed by the phonetic content of speech alone. Features such as pitch, intensity and timing also play an important role. Such features are referred to as prosody or prosodic features.

Prosodic features convey information about the mood of the speaker and about the emphasis placed on specific words. Information such as this is useful in the higher level stages of a recognizer (see Figure 1.2) and may be able to resolve some ambiguities of meaning.

When using prosodic features to assist the interpretation of speech, it is often the pitch of the speech that best indicates word emphasis rather than intensity, as might be expected. Sound duration and intensity, however, can be used as prosodic features to detect the end of phrases (by a reduction in intensity) and stressed words (through a slight increase in word duration). An example of the use of prosody to resolve ambiguity is considered in Lea [10]. He notes the difference between the uninterrupted, falling pitch sentence:

"What is that in the road ahead?"

and the interrupted, rising pitch double question:

"What is that in the road? A head?"

illustrating the possibility of using prosodic features (in this case, mainly intonation and timing) to overcome ambiguity.

The incorporation of prosodic knowledge into speech recognizers is discussed by a number of authors including: Carbonell [5], Komatsu et al. [8], Lea [10] and Fitch [6]. In these studies, the typical prosodic cues are the fundamental frequency, F0, and syllabic duration.
2.3.4 Occasion & Time of Day

Stress and nervous tension, brought on by the formality or strangeness of an occasion, create significant changes in our voice. This can often prove embarrassing for those who demonstrate speech recognizers as the equipment on display is almost certain to have been trained on unstressed speech. Although the unit may have been performing successfully during training, the pressure of public demonstration often affects the users voice enough to seriously reduce the recognition performance.

Speech quality varies throughout the day, too, often being quite poor in the early morning (remember how your voice sounded the last time you were woken by the telephone?).

2.3.5 Context

In Section 2.3.1, the effect of coarticulation on speech variation was introduced. Coarticulation effects, even for a single speaker, change depending upon the context in which a phrase is spoken thus affecting the phonetic quality of the utterance. For example, simple words such as “yes” and “no” can take on many inflections and durations depending on the context of use — see Figure 2.2 below.

![Figure 2.2: Even simple words such as “yes” and “no” can take on many inflections and durations depending on the context in which they are used. Inflections shown are illustrative approximations to the frequency of the speaker’s voice in each case.](image)

---

*A presenter of a popular television technology programme suffered from variation due to occasion whilst demonstrating a voice-controlled car telephone. Having successfully trained the unit on her unstressed, calm voice prior to broadcast, the unit failed completely to recognize her more nervous voice during the live show.*
2.3.6 Speaker Sex and Age

Two speech sounds, one spoken by a male and the other by a female, may be judged to be the same phoneme by a trained phonetician and yet may display quite different characteristics under acoustic analysis. These differences are most easily observed in the location of the energy concentrations in the spectrograms, which are shifted towards higher frequencies for the female speaker. Also, it is found that

- male speech exhibits a lower fundamental frequency (frequency of voiced excitation)
- male vowels tend to have narrower formant bandwidths, and
- male vowels have a first harmonic of lower amplitude [3].

Variation between male and female speakers is obviously a serious problem for recognition schemes based on frequency spectra. Speaker adaptation techniques and larger template databases have been used to solve the problem. However, there are other approaches, based on alternative forms of speech analysis, which are not as sensitive to speaker sex in the way that time-frequency spectral analysis is. For example, Bladon [3] suggests transforming the speech spectrum using an auditory filter which models human auditory “processing”.

Also, it is found that children’s voices generally possess higher fundamental frequencies, similar to female voices. In fact, differences in phonetic quality are assumed to exist across age groups and should be taken into account in speaker independent ASR.

2.3.7 Speech Rate

Probably the most well-known variability in speech concerns changes in speech rate or tempo. The rate at which speech events occur — during sounds, words and sentences — varies almost constantly. For most speech recognition methods derived from template matching schemes, variations in speech rate need to be removed before matching can proceed. In Sections 4.3 and 4.5, it will be seen how normalization of the speech utterance has been attempted in the past; first linearly and, recently, by non-linear methods with a detailed treatment of non-linear normalization in Section 4.9. Today, temporal normalization of speech for recognition is commonplace.

2.3.8 Noise & Environment

Real-life speech recognizers are expected to work in noisy surroundings. Offices, factories and telephone lines all impose their own type of noise on the environment in
which a recognition system operates.

Each source of noise is different in form and intensity ranging from background speech in an office to impulsive white noise on the telephone line. The main problem that arises in noisy environments is that of poorer discrimination between periods of non-speech and noise, weak fricatives or low-amplitude voiced sounds. This gives rise to increased problems in word-boundary spotting and, consequently, reduced recognition performance.

The effect of noise on the speech recognizer can be moderated in a number of ways:

- close-talking, head-mounted microphones
- background noise subtraction (by monitoring the spectral characteristics of the noise during quiet periods)
- backward-searching algorithms at the phoneme or word-level to avoid rejection of weak fricatives and low-amplitude voiced sounds.

In the presence of noise, speakers often change their speech accordingly which adds further variability. This is an example of how the speaker's environment is a key factor in determining speech quality. The tonal quality of our utterances is quite different when we are in a crowded room from that when we are speaking in a quiet area.

2.3.9 Word Boundaries

Continuous speech recognition is made more difficult by the absence of clear word boundaries. Typically, these are instances when two or more words are merged together, often habitually. For example, in the two phrases

“lighthouse keeper” and “light housekeeper”

the location of the word boundary is clearly critical if the correct meaning is to be determined!

Furthermore, in the sentence

“Did you hit it to Tom?”

the boundaries between almost all of the words (except between “you” and “hit” possibly) have been lost or corrupted in some way. Indeed, in everyday, continuous speech, it is rare to hear the words “did” followed by “you” with any recognizable boundary.
To illustrate this, spectrograms of the continuously spoken phrase “did you” together with those for the component words spoken in total isolation is shown in Figure 2.3.

By comparing the continuous phrase (top) with the individual utterances (bottom), two characteristics can be observed. Firstly, the short burst at the end of the individual word “did” has been lost in “did-you”. Secondly, the spectrogram at the beginning of “you” in the upper diagram has changed to a different sound which will not map reliably onto a standard template for “you”.

Figure 2.3: Spectrograms of the continuously spoken common phrase “did you” together with those for the component words “did” and “you”, spoken in total isolation. They show how the detection of word-boundaries is hampered when sounds merge.

Phonologists have defined many of the common problems regarding word boundaries (Klatt [7]). However, a solution to the problem is not straightforward and often relies heavily on the presence of contextual information. For example, the string of phonemes /grɛtɛip/ may be interpreted as either “grey tape” or “great ape” but con-
text ought to be able to resolve this problem!

In Klatt [7], reliance on context is reduced by adding extra options to the lexicon. For example, as well as storing the phonemic strings /d İ d/ and /y u/, for words “did” and “you” respectively, an extra word-boundary phonological rule

\[ /d \# y/ \Rightarrow [3] \]

transforms the word-final phoneme /d/ and the word-initial phoneme /y/ into the phonetic segment [3] — he calls this phonological recoding.

2.3.10 Homophones, Syntactic & Semantic Ambiguity

Homophones are words which sound the same but have different spelling and meaning; “peace” and “piece”, for example. Ambiguities can usually be overcome by referring to the context of the word.

The sentence “the man sat on the chair with the wooden leg”, is an example of syntactic ambiguity; the reader cannot be absolutely sure whether the man or the chair has the wooden leg! In this case, the ambiguity is said to be permanent because, even when the sentence is complete, the correct syntax still cannot be resolved. It is also possible for a sentence to have temporary syntactic ambiguity. Temporary ambiguity, a characteristic of the temporal nature of speech, occurs when an utterance is given different syntactic interpretations during different parts of the phrase. A good example of temporary syntactic ambiguity is the sentence “The old man the boats”. Until the end of the third word, “man” is expected to be a noun. Then, when the rest of the sentence appears, the listener is firstly confused and then switches to a different syntactical interpretation for which “man” is a verb.

If an utterance has an unambiguous syntax, but can be interpreted in more than one way; it contains semantic ambiguity. The phrase “The council decided to subsidize the port” (from Ainsworth [1]) suffers from this problem.

2.3.11 Common Solutions to Speech Variabilities

In most cases, speaker variabilities can be alleviated by careful design of the reference template store, including extra templates if necessary. However, there is a limit to the success and feasibility of these solutions and so a better approach might be to develop a recognition scheme which can adapt to variations in the speaker’s voice. Since variations due to the occasion, time of day, environment and context are often temporary, speaker adaptation methods may yield more reliable, practical and elegant solutions.
2.4: Practical Problems

<table>
<thead>
<tr>
<th>Expected performance</th>
<th>Significance</th>
<th>Number of Utterances</th>
</tr>
</thead>
<tbody>
<tr>
<td>90%</td>
<td>95%</td>
<td>4000</td>
</tr>
<tr>
<td>99%</td>
<td>95%</td>
<td>40000</td>
</tr>
<tr>
<td>99.9%</td>
<td>95%</td>
<td>400000</td>
</tr>
</tbody>
</table>

Table 2.1: Thomas et al. performed experiments to determine the theoretic size of training database required to achieve given levels of performance with various levels of statistical significance.

In the matching criterion developed later (in Section 5), speaker adaptation is incorporated at the acoustic level in an attempt to overcome some of the problems discussed above.

Having covered the main inherent problems in ASR, the discussion now turns to the practical problems encountered in the design of ASR systems.

2.4 Practical Problems

2.4.1 Size of the Training Database

A speaker independent ASR system is trained on a database of utterances spoken by many people. The size of the database is clearly important if the recognizer is to be exposed to all forms of speech variability during training and testing. Ainsworth [1] suggests that for a vocabulary list of size, $V$, with the list of words spoken in $W$ orders (to balance any contextual effects) over different days, with $G$ regional variations, for both sexes, for $S$ speakers and with $R$ repetitions of each list, the ideal number of utterances in the database would be of the order:

$$\text{number of utterances} = 2VWGSR. \quad (2.1)$$

In practice, databases have to be much smaller than this. The consequence is a lowering in the statistical significance of the results — a view which has been quantified in the work of Thomas et al [17]. He suggests that, when calculating the number of test tokens to use, the size of a randomly constructed database should be based upon the level of significance and the expected performance (figures for their problem are reproduced in Table 2.1).
2.4.2 Choice of Basic Recognition Unit

The choice of base unit for the recognition of speech is an important one. The list of candidates includes words, syllables, demisyllables, diphones, phonemes and acoustic sub-word units. Unfortunately, there is no general agreement as to which unit is the best. Each unit has its advantages and disadvantages which need to be considered in any given problem. A summary of the merits of each unit is included in Table 2.2 on page 23.

Words Transcription of speech will conclude with a series of word units. Machine commands, too, are usually given as discrete words: "open", "close", "stop", "start", for example. It is not surprising, then, that many speech recognizers employ whole word matching techniques using the word as their base unit. There are two main disadvantages with this method. Firstly, owing to the difficulty in detecting word boundaries, the input words generally have to be spoken in isolation, which is not natural and not always desirable. Secondly, since the number of words in the English language is very high, any large vocabulary system would have to match the input speech to enormous numbers of stored templates. For a small vocabulary system, however, whole words may be used successfully as the base recognition units thus eliminating the need for an intermediate level of recognition (using phonemes, syllables, etc.).

Syllables Syllables possess some advantages over words as a recognition unit. Firstly, the syllable has a fixed structure, $C_iV C_f$, where $C_i$ is an initial consonant or consonant cluster, $V$ is a vowel or dipthong and $C_f$ is a final consonant or consonant cluster (see Ainsworth [1], pp. 49, for a list of initial and final consonant clusters in British English). Also, segmentation of consonants and vowels is performed reliably by observing the smaller intensity of consonants compared with vowels. Ambiguities arise, however, when it becomes difficult to distinguish between initial and final consonant clusters. The number of syllables has been found to be of the order of 10,000 (Moser [14]), which is a lot smaller than the number of words in the language. However, this figure is still prohibitive in a large number of applications. An important advantage of using syllables as recognition units is that they contain much coarticulation information.

Demisyllables The problem of template storage is partially solved by using demisyllables instead of syllables. A demisyllable consists of just the initial consonant/vowel pair, $C_iV$, or the final consonant/vowel pair, $V C_f$. This reduces the number of stored templates to around 2000. Recognition based on demisyllables
2.4: Practical Problems

has been investigated by Rosenberg et al. [15]. Although the error rate for the
demisyllable recognizer was twice as bad as the corresponding whole word rec-
ognizer (which achieved a 10-15% error rate), the storage space for demisyllable
templates was considerably less.

Allophones Allophones represent a set of phones bearing the same information. Cer-
tain allophones are easy to identify acoustically and, indeed, some word bound-
daries are indicated by them. However, automatic recognition of allophones in the
speech signal is not yet a reliable procedure. Other disadvantages are the large
number of allophones and their dependence upon coarticulation effects.

Phonemes Phonemes are attractive for use as base recognition units because there
are only 40-60 phonemes in the English language (see the algorithm of Kopec
and Bush [9]). Unfortunately, some sounds belong equally well to more than
one phoneme, causing ambiguity. Also, the detection of phoneme boundaries is
difficult.

Diphones The phoneme segmentation problem is partially overcome by using di-
phones. Diphones are segments of acoustic information lying between the centres
of adjacent vowels and consonants. Their advantage is that they contain some
coaarticulation and transitional information. There are around 1000-2000 diphones
requiring a fairly large template store.

Tri-phones Recently, an increasing amount of interest has been shown in the use of
tri-phones as a basic recognition unit. In cases where coarticulation effects and
other influences go beyond the immediate phoneme or phoneme pair, tri-phones
can often reduce ambiguity. Unfortunately, there is a significant price to pay in
terms of the size of the template dictionary, and hence the amount of training
material required.

Acoustic Sub-Word Units Even though linguists and phonetitions have developed ex-
tensive rules for combining most of the base units mentioned above, there is little
reason to believe that these are the best for recognizing speech by machine. More-
over, the detection of linguistic units is not trivial, the primary problem being the
coaarticulation effects across word boundaries. Recently, an alternative approach
has developed which is to use acoustic sub-word units (ASUs).

Acoustic sub-word units need not possess a one-to-one correspondence with lin-
guistic units. Indeed, no linguistic knowledge is used in the segmentation pro-
cess. Instead, utterances are segmented based on acoustic features observed in
the speech signal, ignoring any phonetic preconceptions. There are, therefore, a number of advantages associated with ASUs:

- reliable segmentation of the speech signal
- reduced template storage space compared with word-based recognizers for large vocabulary systems
- uses clustering methods to construct reference templates.

A typical design procedure for an ASU-based speech recognizer might involve:

1. Segmentation of the speech signal into acoustic segments.
2. Clustering of the segments into ASP reference classes.
3. Labeling of the reference ASUs.

The choice of event within the acoustic signal is chosen to ease segmentation, clustering and subsequent classification stages. In Wilpon et al. [18], for example, acoustic events are defined as either steady state or transient sounds. Steady state sounds are periods of similar spectral activity and transient sounds are sequences of rapid characteristic changes. Detection of these events is achieved by firstly identifying anchor points - points of maximum spectral variation. Transient periods are located by examining the spectral variation contour near to the anchor points. Steady state segments are detected by finding a sequence of frames with spectral change values below some threshold. This, and other studies ([11], [16] and [2]) show that ASUs are a good choice for base recognition unit achieving excellent recognition scores [18]. The next phase in ASU-based recognizers seems to be centred around the development of a good method for mapping all acoustic segments to the word level.
### Table 2.2: Summary of basic recognition units for speech recognition showing the number of units needed in a typical dictionary of everyday speech and comments on the relative advantages and disadvantages of each unit.

<table>
<thead>
<tr>
<th>Speech Unit</th>
<th>Approx. size of dictionary</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>word</td>
<td>10,000+</td>
<td>Ultimate goal transcription</td>
<td>Boundaries difficult to detect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Dictionary size prohibitive</td>
</tr>
<tr>
<td>syllable</td>
<td>10,000</td>
<td>Fixed structure, C1VCf</td>
<td>Some C1/CV ambiguity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reliable segmentation</td>
<td>Dictionary size prohibitive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contains coarticulation information</td>
<td></td>
</tr>
<tr>
<td>demi-syllable</td>
<td>2,000</td>
<td>Reduced dictionary size</td>
<td>Higher error rate [15]</td>
</tr>
<tr>
<td>allophone</td>
<td>100</td>
<td>Certain allophones easily identified</td>
<td>Segmentation unreliable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Large number of allophones</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Coarticulation dependent</td>
</tr>
<tr>
<td>phoneme</td>
<td>40-60</td>
<td>Small dictionary size</td>
<td>Several sounds map onto one phoneme</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Boundary detection difficult</td>
</tr>
<tr>
<td>diphone</td>
<td>1,000-2,000</td>
<td>Segmentation easier than phonemes</td>
<td>Fairly large dictionary size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Some coarticulation information</td>
<td></td>
</tr>
<tr>
<td>tri-phone</td>
<td>10,000</td>
<td>Much coarticulation information</td>
<td>Large dictionary size</td>
</tr>
<tr>
<td>acoustic sub-word unit</td>
<td>-</td>
<td>Reliable automatic segmentation</td>
<td>No existing grammar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reduced template storage</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Template clustering possible</td>
<td></td>
</tr>
</tbody>
</table>

2.4.3 End-Pointing

End-pointing is concerned with detecting the beginnings and ends of speech units. End-pointing is commonly employed prior to the template matching phase in isolated word recognition. It is not usually necessary in continuous speech recognition, especially if the pattern matching scheme includes a method for dealing with beginning and end conditions [4].

Most end-pointing techniques involve detecting changes in the total signal energy, the distribution of energy with frequency and a spectral derivative term. In reality, these parameters are not entirely reliable [13], being affected by background noise, weak fricatives and low-amplitude voiced sounds. Hence, to extract significant amounts of reliable training utterances, it is usually necessary to check the end-points by hand.

If the problems mentioned so far are too difficult to overcome, then an alternative approach is to constrain the task in some way. This chapter continues by describing some of the constraints which are possible.
2.5 Constraining the Problem

In the previous section, we have seen that the ultimate goal of a completely faultless, continuous speech, large vocabulary, speaker independent automatic speech recognizer has many inherent and practical difficulties. For the moment, we will disregard the question of whether such a goal will ever be achievable. Instead, we shall turn to systems for constrained applications, which have been available for many years (see Section 4.8) providing reliable solutions in offices and industrial situations.

Usually, the speech recognition task is constrained by using:

- **Isolated word recognition** — words separated with pauses (rather than continuous recognition),
- **Speaker dependence** — the system is trained to recognize just one speaker only (rather than speaker independence designed to operate for any number of speakers),
- **A small vocabulary** — tens of words (rather than a large vocabulary which may contain many thousands of words for everyday speech applications),
- **Environmental restrictions** — operation in a studio or other quiet area (rather than in noisy office or over a telephone line) (see page 15 “Noise & Environment” for details and solutions).

Designers and users of speech recognition systems today consider the above constraining factors in order to achieve reasonably error-free recognition (between 95% and 99% performance figures, for example). Increased recognition performance arises through the removal (or reduction) of variability effects described above. Another way to constrain the problem but still provide the user with some degree of freedom in their speech is to use word spotting techniques.

2.5.1 Isolated and Continuous Speech Recognition

In Section 2.3.9 on page 16, we discussed the problem of detecting word boundaries. Of course, this problem is removed if the speech input is segmented by instructing the operator to insert pauses between words. Such a system is referred to as an isolated word recognizer.

Isolated word recognition is quite adequate in a number of situations — the input of machine commands, for example. Certainly, in the past, some highly reliable recognizers have appeared for constrained tasks of this nature. However, human word segmentation is rather artificial and, obviously, a more desirable goal for systems of
general application is to achieve recognition of continuous speech. However, as we have mentioned, word segmentation in continuous speech recognition is often prone to ambiguity. One solution is to partially constrain the problem by storing templates of word couplets or triplets such as "book-flight", "one-two-three", etc. In this way, short phrase recognition is possible without being concerned about word boundaries; these being already incorporated into the pre-stored model templates.

### 2.5.2 Speaker Independent & Speaker Dependent Systems

A truly *speaker independent* ASR system should respond successfully to any speaker without having been specifically trained on that speaker. For many tasks, however, speaker *dependent* systems are quite adequate, being trained on a single speaker and, therefore, intended for use solely by that person.\(^2\)

Speaker-to-speaker variabilities have already been described above (page 11 onwards). Speaker dependent recognizers suffer less from these factors and therefore tend to achieve higher recognition rates.

A method which boasts some of the benefits of both speaker dependence and independence is *speaker adaptation*. Speaker adaptation relies on the fact that, in many applications, the same user speaks for the duration of the dialogue. As another user commences speaking, the system adapts in a short while to the characteristics of the new voice.

### 2.5.3 Vocabulary Size

It is generally believed that the larger the vocabulary, lexicon or dictionary, the more likely it is that recognition errors will occur. This is especially true if the vocabulary contains similar sounding words. Some tasks, of course, can be successfully accomplished with a small vocabulary; spoken digit/letter recognition for example.

In the past, vocabulary sizes were limited to tens of words. Over the last ten years or so, a steady increase in the reliable size of the vocabulary has been seen. Perhaps the most dramatic demonstration of a low error rate large vocabulary recognizer occurred when the SPHINX system was developed [12] — a reported recognition rate of 96% on a 997-word task. However, most small-vocabulary systems will tend to out-perform large-vocabulary ones and so, for very high performance, small vocabularies are used.

---

\(^2\) Often, a speaker dependent system will operate with a different speaker from the training candidate, but performance is usually reduced.
2.5.4 Word Spotting

Word spotting overcomes the problem of forcing the speaker to pronounce words in an isolated fashion while still allowing the recognition system to operate as a discrete utterance recognizer. This is possible by continuously attempting to match the initial portion of the reference templates with the incoming speech until a good comparison is found. Once found, normal template matching is carried out and a decision as to what was said is made. However, as the incoming speech is continuous and connected, the template matching process is subject to coarticulation and boundary problems.

2.6 Summary

In this section, a general overview of the key issues in speech recognition by machine has been presented. Discussion was divided into:

- inherent issues in speech recognition which are largely independent of the approach taken including natural variabilities in human speech and linguistic ambiguities;

- practical problems, which arise from implementing speech recognizers based on the common framework of template matching with a pre-stored set of reference templates (Figure 1.2);

- constraining the problem in order to overcome some of the difficulties which have been highlighted by employing isolated word recognition, speaker dependence, a small vocabulary, environmental restrictions and word-spotting.

Inherent Issues:

- The effect of neighbouring sounds on articulation, or coarticulation, tends to modify the basic phoneme sound depending upon the context in which it is spoken.

- For true speaker independence, speech recognizers will have to cope with many accents and dialects.

- Extra information is carried by speech in the form of variations in pitch, intensity and timing — collectively termed prosody.

- Speech quality changes depending upon the occasion (particularly for a nervous or tense speaker), the time of day and the context.

- Significant spectral differences exist between males and females, with the characteristic formants in female speech tending to be shifted towards higher frequencies (see page 34 regarding formants).
For template matching, *speech rate* normalization is a central task, successfully approached so far by non-linear (dynamic programming-based) techniques.

- In real-life situations, speech recognizers have to cope with many types of *noise* and changes of *environment* which can seriously affect discrimination capabilities.

- In continuous speech recognition, a major, and largely unsolved, problem is the reliable recognition of *word boundaries*.

- At the higher level of recognition, ambiguities such as *homophones*, *syntactic* and *semantic* ambiguities pose difficult problems for the unconstrained, large-vocabulary recognizer.

**Practical Problems:** A core framework has been adopted upon which speech recognition systems are designed (Figure 1.2). As a result, a number of *practical* and implementational problems can be identified.

- Increasingly, speech technologists are realizing that the creation of reliable and accurate speech models requires much larger *training databases* than had previously been used (Table 2.1).

- An on-going discussion exists concerning the best *basic recognition unit* to use — new candidates are occasionally suggested which tend to gain interest. Each unit, of course, has its associated advantages and disadvantages (Table 2.2).

- Allied to the choice of recognition unit is the problem of *end-pointing*. End-pointing is the task of segmenting an utterance into basic recognition units; this could be *linguistically based* (finding word-boundaries or phoneme boundaries) or *acoustically based* (finding boundaries between acoustic sub-word units, ASUs).

**References**


Chapter 3

Digital Speech Analysis

3.1 Introduction

In Section 1.3, it was mentioned that a typical speech recognition system can be split into a number of levels from the acoustic input signal to the linguistic output. The first stage, pre-processing, is concerned with the sampling and analysis of the time-varying speech waveform and aims to convert the raw time-frequency speech signal into a more compact representation and to extract the relevant waveform features.

Here, a brief introduction to the essentials of this process is presented concerned entirely with the analysis of the speech signal in the discrete-time and frequency domains.

Firstly, the main characteristics of speech in the time-domain are described and the common source-filter model for speech production is presented. Then, the main elements of digital (discrete-time) speech analysis are described. Factors surrounding the dissection of the signal into frames are included such as windowing.

Parametric representations of the speech signal are described with particular reference to mel-frequency cepstral coefficients (MFCCs) and, finally, feature extraction considerations and pattern analysis terminologies are introduced. A block diagram of the whole pre-processing stage is shown in Figure 3.1.

3.2 The Source-Filter Model of Speech Production

Speech sounds fall into two categories: voiced sounds — the “e” sound in “bee” and the “ar” sound in “car” — and unvoiced sounds — “s” in “sizzle”. The waveforms of voiced sounds are periodic in nature with a fundamental frequency up to 400Hz and with reduced energy above 4kHz. Unvoiced sounds, however, produce random variations up to a frequency of 10kHz or more, but with reduced amplitude. An example of both
Figure 3.1: Pre-processing is the first stage in most speech recognition systems. The aim is to convert time-varying speech signals into a compact representation which still retains the majority of the useful information content.
3.3: Digital Speech Analysis & Windowing

Types of speech is contained in the word "sizzling", a portion of which is displayed in Figure 3.2.

![Figure 3.2: First portion of the spoken word “sizzling”, showing examples of unvoiced (“s”) and voiced (“i”) speech.](image)

Current speech analysis is based upon a source-filter model of speech production. This consists of an excitation convolved with the impulse response of a vocal tract filter. The model represents voiced sounds and unvoiced sounds by using a periodic waveform and white noise respectively for the filter excitation signal as illustrated in Figure 3.3.

### 3.3 Digital Speech Analysis and Windowing

Signal processing of speech signals was formerly undertaken by analogue methods. Today, digital techniques are employed almost exclusively which gives rise to numerous, and well-known, advantages [6].

Conversion of the analogue speech waveform into a digital representation follows the common process outlined in Figure 3.4. Regular measurements of the signal amplitude are made by the sample-and-hold device and held constant for enough time
to allow conversion to a digital entity. The anti-aliasing filter (low-pass filter) has a bandwidth of $f_{c}$ which is set to approximately half the sampling rate, $f_s$, to prevent the repeated responses at $nf_s$, $n = 1, \ldots \infty$ from overlapping.\footnote{In the ideal case of zero sample-and-hold time, frequency responses would have equal amplitude for all values of $n$ up to $\infty$, but for a non-zero sample-and-hold time, the frequency response is scaled by the $s^{\frac{\alpha}{2}}$ function.}

In almost all cases, it is sufficient to restrict the speech bandwidth to 4kHz (or 6.4kHz maximum) as little useful information appears at higher frequencies [12]. If the bandwidth exceeds 6.4kHz, recognition performance may even deteriorate. Thus, the typical sampling frequency range is 8-13kHz.

Voiced speech, as mentioned above, has a periodic nature with a fundamental

---

Figure 3.3: The source-filter model for speech analysis which generates speech by convolving periodic functions (voiced speech) or white noise (unvoiced speech) with the impulse response of a vocal tract filter.

---

Figure 3.4: Outline of the analogue-to-digital conversion process.
3.3: Digital Speech Analysis & Windowing

Figure 3.5: Digitized input speech (portion of "sizzling") is divided into frames and usually pre-multiplied by a window function (Hamming window in this case). Then, transformation into the frequency domain occurs on a frame-by-frame basis; successive frames can be overlapped as shown here.

Frequency rarely greater than 400Hz (this figure is believed to be set by the maximum rate at which articulators in the vocal passage can be changed). Two consequences arise from this observation:

1. speech can be represented using parameters which vary at this fundamental frequency;

2. over a "short-time" interval (say 10ms) speech parameters may be assumed to remain constant.

Such a short-time interval is called a frame. The number of samples per frame varies according to the type of analysis required and the sampling frequency, but is typically either 128, 256 or 512.

Each frame is successively transformed using the Fast Fourier Transform [2] (FFT) for analysis in the frequency domain. However, the frame length rarely coincides with a multiple of the signal period and this introduces errors in the frequency response of the speech signal. Thus, frames are usually pre-multiplied by a window function, such
as a Hamming window (Figure 3.5),

\[ w(n) = \begin{cases} 
0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) & 0 \leq n \leq N - 1 \\
0 & \text{elsewhere}
\end{cases} \quad (3.1) \]

where \( N \) is the frame length and \( n = 0, \ldots, N - 1 \).

Straightforward truncation to the frame length causes "ripples" in the frequency domain. The windowing procedure reduces these "ripples" because the sidelobes of the frequency response of the window function are smaller than that for a rectangular window (truncation) (see Harris for a detailed analysis of windowing functions [7]).

Successive frames are sometimes overlapped by half a frame (correlated) to avoid overlapping of the window's main lobe in the frequency domain [10, 1, 11] and to provide some degree of speech continuity and correlation.

The resulting frequency (power density) spectra for voiced and unvoiced sounds are shown in Figure 3.6 on page 35 showing clearly the periodic nature of voiced speech and the randomness of unvoiced speech.

Considering the plot of the voiced sound (top), peaks in the envelope of the spectrum can readily be seen called formants. Formants — of which only the first three or four can be identified to any useful degree of reliability — characterize the vocal tract filter response of the source-filter model and are often used as a parameter for speech recognition.

The length of the frame determines the resolution in time and frequency; a short frame gives good time resolution and can thus identify short-time variations but achieves poor frequency resolution; a long frame has the opposite effect.
3.3: Digital Speech Analysis & Windowing

Figure 3.6: Frequency spectra of voiced (top) and unvoiced (bottom) speech taken from the first "i" and "s" in "sizzling" respectively. Peaks in the spectral envelope (dotted curve) of the voiced sound plot (top) correspond to the vocal tract filter frequency response. (256-sample Hamming window used in each case).
3.4 Parametric Representation of the Speech Signal

In the majority of speech recognizers the speech signal is represented parametrically, instead of using the raw time-amplitude or frequency-amplitude acoustic signal. Thus, each utterance is represented by fewer values and yet the information content remains largely unaltered. Recognizers usually represent the spectral envelope with about 8-14 coefficients [12].

The choice of representation is important. Firstly, it should be easy to compute the speech parameters reliably in real time. Secondly, it is wise to choose the appropriate representation for the task and benefit from the improved recognition performances which have been shown to exist [4], [8].

Usually, the parametric representations are based upon one of the following popular techniques:

- reduced Discrete Fourier Transform (DFT)
- bandpass filter banks
- linear prediction analysis
- formant analysis
- linear discriminant functions
- mel-frequency cepstral analysis.

Many recognizers pre-emphasis high frequencies prior to parameterization so that high and low frequencies receive equal weighting, but this does not always improve performance [9].

It has been shown, in comparative studies [4], that mel-frequency cepstral coefficients (MFCCs) perform well and are consequently one of the most popular parametric forms in use today.

3.4.1 Mel-Frequency Cepstral Coefficients

Mel-frequency cepstral coefficients (MFCCs) are based on a frequency- and amplitude-warped DFT magnitude spectrum and are intended to reflect the non-uniform spectral distribution of phonemic information. The procedure for calculating them is as follows:

Apply mel-spaced filter bank to DFT magnitude spectrum: In order to follow the bark or critical band scale. A mel-spaced filter bank consists of filters which are
equi-spaced up to 1kHz (100Hz bandwidth) and log-spaced above 1kHz (500-1000Hz bandwidth) as shown in Figure 3.7. Effects of varying filter bank parameters are discussed in [3]. In [4], twenty such filters were used.

![Figure 3.7: Mel-spaced filter banks which are applied to the magnitude output of a DFT to obtain mel-frequency cepstral coefficients.](image)

Perform an inverse DFT on the filter bank outputs. If the log-energy output of each of the twenty filters is $X_k$, $k = 1, \ldots, 20$, then $M$ MFCCs, $c_n$, $n = 1, \ldots, M$, can be obtained,

$$c_n = \sum_{k=1}^{20} X_k \cos \left[ n(k - \frac{1}{2}) \frac{\pi}{20} \right] \quad n = 1, \ldots, M \quad (3.2)$$

The $c_0$ coefficient represents the average energy in the speech frame and is often discarded to implement some form of amplitude normalization.

### 3.5 Pattern Analysis and Feature Extraction

Before the recognition phase, the parametric representation is sometimes processed further to bring out relevant acoustic features and remove parameters which may be detrimental to recognition (see Appendix A for experimental validation of this). Methods for achieving a reduction in the number of parameters include transformations and optimal feature selection techniques based on search strategies such as the branch and bound algorithm [5].
Eventually, a set of parameters or features are found which represent each speech frame adequately. If there are \( N \) such features then they can be viewed as an \( N \)-dimensional feature vector — a point in \( N \)-dimensional space. Thus, each frame can be plotted in the space as a point. Frames containing similar speech should produce points which are close together — they are said to form clusters — as shown in Figure 3.8.

![Figure 3.8: Each speech frame can be plotted as a feature vector in \( N \)-dimensional space. Similar frames tend to form clusters.](image)

The advantage of viewing the speech signal in this way is that pattern analysis principles can be applied. For example, assuming the feature vectors follow a Gaussian distribution, clusters can be modeled by their mean, \( \mu \), and covariance, \( \Sigma \) (Figure 3.9)

\[
\mu = \frac{1}{N} \sum_{j=1}^{N} y_j
\]

\[
\Sigma = \frac{1}{N} \sum_{j=1}^{N} (y_j - \mu)(y_j - \mu)^T
\]

where \( y_j \) is the \( j \)-th point in the cluster.

The benefit of using this terminology is that measures of similarity, or distance measures, can be employed along with the host of tools developed in the pattern analysis field. The common measure of similarity is the Mahalanobis distance

\[
M(y) = (y - \mu)^T \Sigma^{-1} (y - \mu)
\]
3.6: Summary

In this chapter, the basic elements of digital speech analysis, or pre-processing, in speech recognition systems have been described:

- the source-filter model of speech production which separates excitation and vocal-tract filtering to model voiced and unvoiced speech;

- an outline of the process of converting analogue speech signals into digital values;

and its identity-covariance version, the Euclidean distance ($\Sigma = I$, where $I$ is the unit matrix). Classification of unknown speech frames can then be based upon statistical decision theory, which is centred around the Bayes decision rule [5].

However, the jump from the clustering of speech frames to using pattern analysis methods in the recognition of whole words (groups of frames) is not straightforward. The number of frames produced during the utterance of a particular word will differ from speaker to speaker. As we shall see later, it is necessary to perform normalization of the number of frames per utterance if these decision-theory comparison techniques are to be used.

3.6 Summary

Figure 3.9: Assuming the feature vectors follow a Gaussian distribution, clusters of speech frames can be modeled by their mean, $\mu$, and covariance, $\Sigma$. 
- windowing and the dissection of speech into frames;
- digital frequency analysis;
- parametric representation of speech into a compact form with particular reference to mel-frequency cepstral coefficients;
- an introduction to basic pattern analysis terminology and feature extraction.

References


Chapter 4

A Review of Approaches to Automatic Speech Recognition

4.1 Introduction

Many approaches to the problem of automatic speech recognition have been employed over the last forty years. In this Chapter, the most popular and common approaches are reviewed with particular attention given to the contribution made by individuals and groups. This review, together with some examples of commercial and experimental recognizers, aims to provide a concise overview of developments in ASR to date.

We consider, firstly, the ingenious early attempts at recognizing speech by machine. Then, the move towards pattern recognition and template matching is covered followed by the incorporation of linguistic knowledge in the 1970s. The popular methods of today are introduced, namely: dynamic time warping (DTW), hidden Markov models (HMMs) and artificial neural networks (ANNs) with, finally, a detailed introduction to the principles of DTW as a basis for the new work presented in Chapters 5 - 7.

4.2 Early Attempts

Attempts to translate speech into other forms began in the 1950s. Dreyfus-graf [27] reported work on his phonograph which attempted to translate speech into stenolike symbols. At about the same time, recognizers based on distinctive features were proposed. Distinctive features (of which there are around twelve) are used to represent phonemes in a more efficient way than using phonemes alone.

The first attempt at developing a word-based automatic speech recognizer was reported by Davis et al. [23] in 1952 tackling the problem of spoken digit recognition. He took the approach of splitting the speech signal into two frequency bands, F1 and
F2, counting zero crossings (to give an approximate measure of the first and second formant frequencies) and then segmenting into 100/500 Hz bands. A matrix was then constructed with the sub-divided frequency spectrum forming an F1–F2 plane. Classification proceeded by calculating the time spent by the signal in each element and cross-correlating this figure with reference patterns. This system achieved 98% correct classification performance on a single speaker and was later improved upon by Dudley and Balashek [28].

Minor improvements were made by Olson and Belar [78] who attempted to reduce the effect of speech variability by using eight filters with amplitude compression to produce a crude spectrogram. Their aim was to decode ten syllables into letters and their system achieved 99% correct recognition with careful pronunciation.

### 4.3 Template Matching and the Application of Pattern Recognition

With the growth in the use of digital computers came the ability to evaluate more complicated speech recognition methods. A common theme to these methods was template matching. Today, a large proportion of speech recognition systems are based on some form of template matching scheme. The method firstly requires the generation of reference templates, which represent good approximations of each utterance in the vocabulary, by some averaging or clustering technique. These reference templates are then compared, or matched, with the current input speech waveform. A decision on which template best matches the current input is based on some comparison criterion, usually a distance metric, such as the Euclidean distance (equation (4.5) on page 57).

Methods based on pattern recognition techniques also attempted to tackle the problem of speech and speaker variability by normalizing some speech parameter. Forgie and Forgie [32] aimed to reduce the effect of inter-speaker variability by normalizing formant frequencies with the fundamental frequency. The resulting formant-based vowel recognizer achieved 93% correct recognition when tested with 21 male and female speakers.

One of the first attempts to normalize the duration of speech waveforms was reported by Denes and Mathews [24]. They used a 17 channel spectrum analyser as the input to a computer to solve a digit recognition problem. Utterances were averaged and stored as templates. During classification, the stored templates were cross-correlated with the unknown speech input. On a test with six male speakers and one female speaker, the method achieved a 6% recognition error rate. This compared with a 12% error rate without the time normalization technique.
The problem of recognizing sequences of speech has been approached by a number of people (e.g. Sakai and Doshita [97]). These efforts were based on an approach reported by Wiren and Stubbs [112]. Wiren and Stubbs constructed a binary classification tree to successively sub-divide the speech into linguistic units and finally phonemes.

In 1975, Sambur and Rabiner [98] developed a digit recognizer with self-normalising acoustic parameters. Excellent results were reported using this system: 5.6% error rate in a noisy environment and 2.7% under quiet conditions.

4.4 Incorporating Linguistic Knowledge

During the 1970s, speech recognition research moved towards the incorporation of linguistic knowledge. This activity was exemplified in the five year Speech Understanding Research (SUR) project funded by ARPA (Advanced Research Projects Agency), the primary focus of attention being continuous speech. Out of this came recognizers such as HEARSAY, HARPY and DRAGON.

HEARSAY [89] used pragmatic (world) knowledge from a wide range of sources to deduce the likely set of utterances. An additional aim was to design flexible knowledge sources (e.g. phonological, prosodic, lexical and syntactic) and control strategies. This would enable new sources of knowledge to be added to or to replace existing members of the knowledge base. The DRAGON [5] and HARPY [67] projects integrated a number of these knowledge sources into a network (based on the beam search technique in HARPY).

Tappert et al. [103] also used syntactic knowledge in a tree-based recognizer. In his system, branches represented possible sentence matches constructed from a 250-word vocabulary. Using a finite state grammar, the recognizer achieved 91% word accuracy and 54% for sentences.

Another speech recognizer of the 1970s, SPEECHLIS [114], incorporated semantic knowledge. Thus the appearance of a key word triggered a search for other words which were to be expected as a result of the occurrence of the key word.

The effect of neighbouring acoustic features usually causes problems in speech recognizers. However, environmental changes in pronunciation were used to improve performance by Oshika et al. [79]. Similarly, Lea [59] extracted prosodic features for recognition purposes and made suggestions for their use.
4.5 Dynamic Time Alignment

Dynamic time alignment (or Dynamic Time Warping [DTW]) aims to overcome one of the prime causes of variability in speech; the global and local variation in speech rate. DTW preserves the general shape of the speech waveform, but removes temporal differences during the template matching process (see Figure 4.1). Until the introduction of DTW algorithms, only linear normalization of speech utterances was performed, if at all.

Figure 4.1: Dynamic time alignment (or warping — DTW) of a hypothetical speech signal. DTW preserves the general shape of the speech waveform, but removes temporal differences during the template matching process (from Peacocke and Graf).

Non-linear time alignment methods have been based mainly on the dynamic programming algorithms of Bellman [11]. First applications of dynamic programming to speech were reported by Vintsjuk [107] in the late 1960's and by Velichko and Zagoruyko [106] in 1970.

Much of the development of dynamic time alignment methods at present is based upon the work of Sakoe and Chiba [96] who optimized a number of dynamic programming algorithms using Japanese digit recognition as a test vehicle.

For connected digit recognition, Sakoe [95] suggested a two-level dynamic programming algorithm which was later developed by Bridle and Brown [17].

Many ASR systems have been produced based upon the dynamic programming method and a number of papers published which report the results of some successful
modifications to the original method (Paliwal et al. [80] Rabiner et al. [86], and Das [22] for example).

To overcome the need to time-align the input utterance with every reference template, Raswan and Fahmy [88] proposed a method whereby the utterance is time-aligned only once with a single reference pattern or TAP (Time Alignment Pattern). Then, selection of the best reference template is made by computing the distance between the pre-warped speech input and the other reference templates.

4.6 Hidden Markov Models

In the early 1980s, interest grew in using stochastic models for speech recognition. In particular, the use of hidden Markov models (HMMs) became popular and described in a number of tutorial papers of which the one by Levinson et al. [62] is well known.

The theory of HMMs was published in a series of papers by Baum [8]-[9] et al. and applied to speech processing by Baker [5] and Jelinek et al. [47]. The underlying idea in this stochastic approach is to model the speech signal as a set of definite states. Each state has its own probability distribution and a set of state transition probabilities which model the behaviour of the signal in time. As a result, every possible state sequence can be assigned a probability of occurrence (Figure 4.2).

![Diagram of HMM States and Transition Probabilities](image)

**Figure 4.2:** The underlying idea in the hidden Markov model approach to ASR is to represent the speech signal as a set of definite states which can be frames, linguistic units, words, etc. Each state has its own probability distribution and a set of state transition probabilities which model the behaviour of the signal in time (after Picone).

It has been found that when this process is applied to each level of the recognition process (e.g. acoustic feature, word and sentence level), excellent results can
be achieved. Recent and comprehensive introductions to HMMs with application to speech recognition are provided by Rabiner [85], Poritz [83] and Picone [82].

There have been a number of ASR systems based on HMMs produced to date. Some of the best known systems include the SPHINX system by Lee [48] at Carnegie Mellon University, TANGORA by IBM (Bahl et al) [3], the AT&T Connected Digit Recognizer (Rabiner et al [87]) and the Texas Instruments Connected Digit Recognizer (Doddington [26]). SPHINX is a speaker-independent, large vocabulary system with reported recognition rate of 96% on a 997-word task. TANGORA accepts English sentences drawn from a 20,000-word vocabulary achieving around 5% error rate on a several-speaker dependent task and 0.7% error rate.

Since their introduction, HMMs have been developed extensively and have been proven to be one of the most successful approaches to the machine recognition of speech to date.

4.7 Artificial Neural Networks

Artificial neural networks are architectures, processors or systems which usually contain distributed and parallel processing elements configured in an attempt to model biological neural networks. Early work in the development of mathematical models for artificial neural networks was performed by McCulloch and Pitts [69], Hebb [36], Rosenblatt [92] and Widrow and Hoff [110].

After an initial surge of interest in this field, research effort slowed, for a number of reasons [72], and gave way, in the 1970s, to approaches based on high level knowledge. In the early 1980s, interest was rekindled through the work of Hopfield [39, 40, 41], Kohonen [52], Rumelhart et al. [93], Rumelhart and McClelland [94], Sejnowski [100], Feldman [30] and Grossberg [34].

Techniques for modeling the human brain were first applied to the speech recognition task in the early- to mid-sixties [77]. Despite some initial enthusiasm, the majority of research effort was redirected towards other approaches such as the incorporation of linguistic knowledge. However, the revival of interest in artificial neural networks has led to a multitude of new work in the application of neural net models to speech processing. In the field of speech recognition, the multi-layer perceptron, trained with the back-propagation algorithm (Figure 4.4) of Rumelhart et al, and Kohonen's self-organizing feature maps [52] (Figure 4.3) have emerged as being the most successful.

Self-Organizing Feature Maps (SOFMs): Kohonen has developed a hardware speech recognizer for Finnish words based on a feature map [53] — the learning strategy for
feature maps is shown in Figure 4.3. The feature map will eventually contain a set of reference vectors which will act as class templates. To begin with, however, the map is initialized with random vectors. During the unsupervised\(^1\) learning, or training, phase, some points in the map are selected and moved towards the current training utterance. Points are chosen if they are in a neighbourhood whose centre is marked by the closest map point, \(c\), to the current training sample; the measure of "closeness" is determined with a Euclidean distance measure. The size of the neighbourhood and the amount of movement towards the training sample is decreased with each sample. The most important aspect in SOFMs is that the feature map points self-organism, without

---

\(^1\)The training utterance class labels are not used
knowledge of the training utterance class labels, into phonemic clusters as the learning phase progresses. The trained map is used as a nearest-neighbour [25] phoneme recognizer with the decisions fed into a rule-based post-processing stage. More recently, Kohonen has developed a further stage to the feature map recognizer called LVQ2 [54]. It has been shown by McDermott and Katagiri [70] that LVQ2 outperforms other neural network algorithms on the /b/, /d/, /g/ phoneme recognition task. Further and more detailed coverage of Kohonen's SOFMs can be found in Appendix A where some results of a study using them is presented.

Multi-Layer Perceptrons (MLPs): Many applications of MLPs to speech recognition have been made [64]. An MLP consists of layers of nodes, each performing a simple weighted sum and thresholding function (Figure 4.4).

![Multi-layer Perceptron Neural Network](image)

**Figure 4.4:** The structure of a multi-layer perceptron neural network consists of layers of nodes, each performing a simple weighted sum and thresholding function.

The output layer contains as many nodes as there are classes and one node will fire to signify the class membership of the current input pattern. Sometimes, there are $\log_2 m$ output nodes, where $m$ is the number of classes. In this case, a binary combination of the output nodes will fire.

Training begins by presenting a known pattern, $x, z = [x_1, x_2, \ldots, x_N]$ at the input and noting the error at the output between the desired and actual output values. A desired output value would be 1 (fire) for a correct classification and 0 or -1 (no-
fire) for an incorrect classification. The back-propagation training algorithm updates the inter-node weights $w_{ij}$ according to the error at the nodes in the output layer. Weight updating continues in this way until convergence is reached; this usually takes a significant amount of time (see Appendix A).

A number of tutorial papers have been written as a result of this new wave of interest in neural nets [63, 43, 105]. These papers have been supplemented with a number of comparative studies of neural net and conventional classifiers [68, 42, 20, 10]. Such comparative studies have reported the superior classification performance of neural net methods over traditional methods and also highlighted some of the disadvantages. For example, MLP training algorithms are exceedingly expensive in terms of computer time, although major advances in the learning rate have been achieved by Haffner et al [35]. However, on a more positive note, MLPs have been shown to cope well with noisy inputs in a variety of applications [14, 76].

A criticism often made towards MLPs is the lack of sound understanding regarding the internal representations. Indeed, this fact may explain why many researchers discontinue work on MLPs when they fail to improve significantly on their initial results. Some good papers have been published which attempt to provide a better understanding of the discriminant surfaces formed [63, 66] and work continues by a core of dedicated proponents of neural-nets to improve matters in this area [90].

Time-Delay Neural Networks: Currently, one of the most important lines of research into MLPs is based on time-delay neural networks (TDNNs) originated by Waibel [108, 109, 99, 35, 15, 57]. The TDNN incorporates temporal information from the speech signal by scanning the input frame with an MLP. Results have shown this method to be very successful on phoneme and digit recognition tasks [15, 57, 99, 108, 109].

Attempts have also been made to combine neural networks with established speech recognition techniques, such as dynamic programming [15], and with other disciplines, such as expert systems [57]. It is also not surprising that efforts have been made to combine neural net approaches, Kokkonen and Torkkola [55], for example, who combine and MLP with an SOFM.

Artificial neural networks have become one of the most important research areas in speech recognition and, as a result, a number of major research projects have appeared in the UK we have the BT Connectionist Project and active research at Cambridge University and at the Royal Signals and Radar Establishment. However, research is still very active in the more traditional and well-established fields of DTW and HMMs.
4.8 Automatic Speech Recognition Systems

Since the early 1970's, complete automatic recognition systems of various complexity have been available. Some systems have been designed as commercial products for applications such as hands-free factory work [65], office work [16, 4], etc. Others were experimental, demonstrating and evaluating methods. More recently, a number of integrated circuits have been developed [45, 84, 33] allowing bolt-together systems to be designed. Examples of some ASR systems that have been produced in the last 20 years are shown in Table 4.1.

Among this list are some ASR systems that have become famous in the field of speech recognition. HARPY, HEARSAY, HWIM, for example, were products of the five-year Speech Understanding Research (SUR) project funded by the Advanced Research Projects Agency (ARPA) of the USA.

More recently, much interest has been aroused by the development of MLP-based recognizers such as SPHINX and TANGORA.

Landmark

So far, this review of ASR approaches has provided an overview of the methods developed, and the important work reported, since the early 1950s. Now, we look more closely at DTW as an example of how time-normalization and template matching are carried out. DTW is commonly implemented using dynamic programming (DP). Here, the basic idea behind non-linear normalization of speech using DP is presented.

Firstly, the basic concepts of DP and the associated terminology are introduced with reference to a simple example. Then, the application of DP to discrete utterance and connected speech recognition is described including the constraints imposed on the DP technique by the speech recognition problem.
Table 4.1: Selected examples of commercial and experimental automatic speech recognition systems that have been produced in the last 20 years.
4.9 Non-Linear Time Alignment Using Dynamic Programming

4.9.1 Non-Linear Time Alignment

Early attempts to normalize speech used linear normalization along the time axis, with limited success. However, since the mid- to late-sixties, non-linear time-normalisation demonstrated improved recognition performance and, today, is used almost exclusively.

It is easy to see why non-linear time-normalization is better than linear normalization at removing changes in speech rate by comparing the two methods on a typical problem. In Figure 4.5 from Silverman and Morgan [102], a comparison of linear and non-linear normalization is shown by warping the word "speech". In the figure, an utterance of the word along the x-axis is being normalized against a reference pattern, or prototype, of the same word along the y-axis. The portions of speech corresponding to /ee/ and /sh/ are clearly different while other portions are similar in duration.

![Figure 4.5: Comparison of linear and non-linear time normalization applied to an utterance of the word "speech" showing the superior performance of the non-linear method applied to a typical ASR example problem (after Silverman and Morgan).](image)

The majority of non-linear normalization schemes in use today are based on dynamic programming which is discussed in the next section.
4.9.2 Introduction to Dynamic Programming

Dynamic Programming is a mathematical concept used for the analysis of sequential decision processes. It has a long and well-established history which can be traced back to the mid-1940s and was popularized by the works of Bellman [11] and Bellman and Dreyfus [12]. The concept is based upon a simple property of multi-stage decision processes — The Principle of Optimality:

**The Principle of Optimality:** An optimal policy [or path] has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy [path] with regard to the state resulting from the first decision [12].

In simple terms, this implies that the final path depends upon the initial state and upon making only optimal decisions at intermediate stages along the way.

For an introduction to the concepts of DP, consider the worked example from Silverman and Morgan [102] which is summarized below.

The example attempts to construct the best four-word sentence from the output of a simple isolated word (discrete utterance) recognizer having a four-word vocabulary, cat, fat, sat, that. The sixteen possible states through which the path could pass are shown in Figure 4.6 with their associated probabilities of occurrence, \( P(i, n) \) (word \( i \), \( i = 1, \ldots, 4 \) at time \( n \)), at the bottom-right of each state. A table of transition probabilities for the utterance of word \( i \), at time \( n + 1 \), given that word \( j \) is uttered at time \( n \), \( Q(i|j) \), is also shown in the same figure (a "silence" word has also been added).

Representing the sentence as a sequence of word numbers organized into a vector \( i \) and defining a correctness measure, \( C \), quantifying the possibility that the sentence was uttered

\[
C(i) = \sum_{n=1}^{4} Q(i_n|i_{n-1}) P(i_n, n)
\]

the problem is then to find the sentence \( i_{\text{opt}} \) which maximizes the measure \( C \),

\[
i_{\text{opt}} = \arg\max_i C(i),
\]

where the abbreviation \( \arg\max \) signifies that \( i_{\text{opt}} \) is the particular value of \( i \) which maximizes \( C \).

Dynamic programming allows this maximization process to be performed recursively. Thus, if \( g(i, j) \) is defined as the maximum value of \( C \) for a
Figure 4.6: A state transition diagram for the cat, fat, sat, that example showing probabilities of occurrence of each word (bottom-right) and some explored paths. A table of transition probabilities from one word to the next is also shown.

The probability that word \( j \) is uttered at time \( n+1 \) given that word \( i \) is uttered at time \( n \) is shown in the table below:

<table>
<thead>
<tr>
<th>Word at ( n+1 )</th>
<th>Silence</th>
<th>Cat</th>
<th>Fat</th>
<th>Sat</th>
<th>That</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>0.10</td>
<td>0.05</td>
<td>0.50</td>
<td>0.10</td>
<td>0.45</td>
</tr>
<tr>
<td>Fat</td>
<td>0.40</td>
<td>0.10</td>
<td>0.20</td>
<td>0.30</td>
<td>0.45</td>
</tr>
<tr>
<td>Sat</td>
<td>0.10</td>
<td>0.50</td>
<td>0.10</td>
<td>0.30</td>
<td>0.05</td>
</tr>
<tr>
<td>That</td>
<td>0.40</td>
<td>0.35</td>
<td>0.20</td>
<td>0.30</td>
<td>0.05</td>
</tr>
</tbody>
</table>

For a sentence of \( i \) words and ending with word \( j \), then

\[
g(i, j) = \max_{1 \leq k \leq 4} [(g(i-1, k) + Q(j|k) \cdot P(j, i)]
\]  
(4.3)

This recursion is best explained with reference to Figure 4.6.

Starting at the \textit{init} state at the bottom-left hand corner of the graph, the four paths to the first column (time 1) are produced using equation 4.3. Values for \( g(i, j) \) are displayed in ellipses adjacent to each explored state. If we then hypothesize that, at time 2, the partial sentence ends in fat (state A) then, from the graph, there are four possible paths leading to this word (shown dotted). The thick dotted line represents the best of the four paths (largest value of \( g(i, j) \)), and is calculated as follows:

\[
g(2, 2) = \max_{1 \leq k \leq 4} [g(2, k) + Q(2|k) \cdot P(2, 2)]
\]  
(4.4)

The other three partial paths can be discarded as they will not lead to a better path overall which passes through state (2, 2). This is the essence of the dynamic programming scheme. Applying equation 4.3 from left-to-right...
will lead to a set of four best paths which reach the four terminal words at time 4 as shown in Figure 4.7.

Figure 4.7: The completed dynamic programming search for the cat, fat, sat, that problem showing the back-tracking, second-pass phase (dark line) to recover the words of the best sentence.

Dynamic programming is said to be a “two-pass” process: a forward pass and a backward pass. In fact, the backward pass is only necessary if the words comprising the best-sentence are required, rather than simply the best score. In this example, the backward pass has been performed (dark line in Figure 4.7) requiring the storage of the best scores, and their associated words, at each stage. As a result, we obtain the best, or most likely, sentence: that fat cat sat.

4.9.3 Dynamic Programming Applied to Isolated-Word Recognition

The extension of the dynamic programming idea to non-linear time alignment of low-level speech signals is relatively straightforward. Firstly, consider an alignment space, or matrix (Figure 4.8), similar to the state transition diagram in the previous example. The reference template for the particular utterance is placed along the y-axis — one sample per matrix location — and the candidate input utterance along the x-axis. Paths through the matrix represent stretched and compressed matches between these utterances. The optimal match can be found by employing DP from the bottom-left
56 Review of Approaches to ASR

Figure 4.8: The template is placed along the y-axis and the candidate input utterance along the x-axis. Paths through the matrix represent stretched and compressed (time-warped) matches between these utterances.

hand corner of the matrix to the top-right hand corner. For isolated-word recognition, these end-points are assumed to be known.

Looking at this another way, the candidate utterance is mapped onto the prototype template using a non-linear transformation (Figure 4.9), seen as a path through the matrix. This path has been computed recursively using DP.

The application of DP to speech template matching usually proceeds in the following manner:

1. Compute local, feature-to-feature distances,
2. Compute cumulative distances (accumulated path length),
3. Obtain path-length-normalized cumulative distance (for connected speech recognition only),
4. (Optional) Back-track to obtain the best-match warped waveform.

Computing Local Distance Measures. In order to implement the recursive DP algorithm, a suitable similarity, or distance measure needs to be defined to enable two
4.9: Dynamic Programming in ASR  

Figure 4.9: The candidate utterance is mapped onto the prototype template using a non-linear transformation which has been computed recursively using dynamic programming.

feature vectors to be compared. Commonly, the Euclidean distance $d(i, j)$ is used

$$d(i, j) = (y_i - \mu_j)^T(y_i - \mu_j).$$  \hfill (4.5)

between the $i$-th sample of the speech pattern to be classified, $y_i$, and the $j$-th sample of the stored template $\mu_j$. This distance calculation is visualized in Figure 4.10.

Computing Cumulative Distance Measures. This is effectively the total path length accumulated as the matrix is traversed (Figure 4.10),

$$g(i, j) = d(i, j) + \min \begin{cases} g(i - 1, j) \\ g(i - 1, j - 1) \\ g(i, j - 1) \end{cases}$$  \hfill (4.6)

and is the basic recursive calculation in the DP algorithm. This is one specific form of the recursive calculation (based on a Sakoe-Chiba constraint); there are others which are discussed below.
Figure 4.10: Top: the first step in the DP method — calculation of the local distance between a sample in the input speech signal and another sample in the template. Bottom: calculating the cumulative distance $g(i, j)$ from the local distances, $d(i, j)$. 
4.9.4 Constraints

When applying DP to the speech problem, it has been found that some constraints need to be applied. These are resolved into

A Global Constraint to regulate the overall allowable stretching and compression of the utterances;

Local Constraints to govern local decisions such as the allowable number of features to skip (continuity), the maximum number of predecessors in the cumulative distance calculation and the preservation of the natural order of events (monotonicity);

A number of different constraints have been proposed of which those suggested by Itakura [46] and Sakoe and Chiba [96] are the most famous.

(A) The Itakura Constraints

Global Constraint. The Itakura global constraint sets the maximum stretching and/or compression factor to two, restricting the slope of the matching path (Figure 4.11) to between 2 and 1/2. For isolated word recognition, the ends of the match can be locked to (1, 1) and (NZ, N) which assumes silence at either end of the word. This forces the search space to assume the parallelogram shape of Figure 4.11. It is worth noting that if the number of samples in the template, N, is twice (or half) the number of samples in the input candidate utterance, NZ, then the search space reduces to a diagonal line only.

Local Constraints. The Itakura local constraints are shown in Figure 4.11. The normal situation (left) allows three possible predecessors. In this case, horizontal skipping is disallowed but a single vertical skip is allowed setting a maximum slope of two. However, if two consecutive horizontal paths are chosen, the global constraint is violated and an overall horizontal path could occur. Hence, the second local constraint (Figure 4.11, right) is imposed to prevent two such horizontal paths occurring.

As a result, the recursive DP equation which employs Itakura constraints can follow three paths, A, B and C, and is calculated as follows:

\[
g(i, j) = d(i, j) + \min \begin{cases} 
g(i - 1, j) & \text{A if predecessor not A} 
g(i - 1, j - 1) & \text{B} 
g(i - 1, j - 2) & \text{C} \end{cases} \tag{4.7}
\]

These constraints are said to be asymmetric — the matching of every sample in the input utterance (horizontal axis) is forced and yet a skip is allowed in the template.
Figure 4.11: Top: the Itakura global constraint sets the maximum stretching and/or compression factors to two thus restricting the slope of the matching path to between 2 and $\frac{1}{2}$. Bottom: Itakura local constraints allow three possible predecessors disallowing horizontal skipping or two consecutive horizontal paths.
This is good for implementation as the same storage locations can be used for the new cumulative path values as were used for the old ones — a true in-place algorithm.

(B) The Sakoe-Chiba Constraints

**Global Constraint.** The global constraint suggested by Sakoe and Chiba states that the DP search space should be limited to a band of width $M$ at 45° (Figure 4.12).

![Diagram](image)

**Figure 4.12:** The Sakoe and Chiba global constraints are a band of width $M$ which rises at 45°. This can be modified for isolated word recognition (parallelogram) to have slopes of 2 and 1/2.

For isolated word recognition, the band can be modified to a parallelogram with restricted end-points such that the slope is limited to be within the range 2 to $\frac{1}{2}$.

**Local Constraints.** Sakoe and Chiba have defined a family of local constraints — both symmetric and asymmetric. Examples of the symmetric ones are shown in Figure 4.13.

In Figure 4.13(a), it is noticed that no local slope constraint is imposed thus allowing long horizontal or vertical paths to occur within the global "band" constraint. Slopes of 2 or $\frac{1}{2}$ are imposed by the constraints in Figure 4.13(b) and slopes of 3 or $\frac{1}{3}$ by the constraints in Figure 4.13(c). Additionally, path weights can be incorporated to favour some predecessors more than others yielding a DP equation (for the constraints
Figure 4.13: Examples of the Sakoe-Chiba family of local constraints which contains both symmetric and asymmetric constraints — symmetric constraints only are shown here.

in Figure 4.13(a))

\[
g(i, j) = \min \begin{cases} 
g(i-1, j) + w_1d(i,j) 
g(i-1, j-1) + w_2d(i,j) 
g(i, j-1) + w_3d(i,j) \end{cases} \tag{4.8}
\]

with weights \(w_1\) to \(w_3\) taking values of 0.5, 1 or 2, typically.

Sakoe-Chiba constraints do not allow in-place algorithms — 2, 3 and 4 storage vectors are needed for the local constraints in Figure 4.13(a), (b) and (c) respectively.
4.9.5 Dynamic Programming Applied to Connected Speech Recognition

In discrete utterance recognition, a number of assumptions about the input speech were possible allowing a relatively simple application of DP. In connected speech recognition (CSR), these assumptions cannot be made. Typically this means

- no a priori knowledge is available regarding the number of words in the input string,
- the beginning and end points of the portion of speech to be matched are unknown,
- the vocabulary is often unknown (despite having a fixed size template database).

The problem is compounded by the difficulty in detecting word boundaries. To overcome this, the stored template is matched against subintervals of the input phrase with overlaps and gaps between successive matches allowed. Matching along the template time axis is favoured.

In general, approaches to CSR using DP are split into early decision and deferred decision types. Early decision methods perform continuous DP matching in a left-to-right manner and produce decisions about what is being said as the phrase is uttered.

Figure 4.14: The main problem in DP applied to CSR is the absence of a clear endpoint. Here, a number of possible “optimal” paths have been developed to match a template to a subinterval of a connected speech input.
Deferred decision approaches wait until a pause in the speech is detected before generating a decision.

The main problem in DP applied to CSR is the absence of a clear endpoint. As a result, the calculation of the optimal match is much more computationally intensive. For example, in Figure 4.14, a number of possible "optimal" paths have been developed to match a template to a subinterval of a connected speech input. The aim is, therefore, to reduce the number of calculations which need to be performed and/or reduce the "best-match" candidate storage space.

A number of algorithms which address this aim have been produced. The most famous of which are the NEC algorithm by Sakoe, Chiba and Kato [49, 50], the Level Building algorithm by Myers and Rabiner [74, 75] and the Bridle et al algorithm [17, 19].

### 4.10 Summary

In this chapter, a number of famous approaches to solving the speech recognition problem have been introduced:

- Early attempts based on signal processing techniques,
- The application of template matching and pattern recognition,
- The incorporation of high-level linguistic knowledge,
- Dynamic time alignment/warping,
- hidden Markov models,
- Artificial neural networks.

In order to prepare the way for the new theory in the following chapters, a more detailed description of DTW was given. The basics of DP for DTW, were described, with reference to an illustrative example. Then, the implementational details of DP applied to isolated word recognition and connected speech recognition were covered including the local and global constraints commonly employed.

### References


[73] H. M. Moser (1969), One-syllable words, (Merill, Columbus OH).


Chapter 5

The Matching Criterion

5.1 Introduction

From the preceding chapters, it is clear that the process of recognizing an input speech signal requires the matching of the signal to a number of pre-stored reference templates. It has been shown that this process is impeded by local and global variations along the time axis. The problem has been approached, with some success, by applying dynamic programming (DP) algorithms as described previously in Section 4.9.

However, traditional DP techniques assume that frames in the speech waveform are statistically independent creating template models based on the first order statistics of the speech waveform alone. In other words, traditional methods base the recognition process on criteria computed from frequency-amplitude measurements acquired in a single frame of speech.

In this chapter, a new method for dynamic time alignment of speech utterances is introduced where each class of speech entity being matched is modeled by a conventional template but with a full covariance matrix representing the correlations between frames. By attempting to incorporate the dependencies between frames, full utilization of the discriminatory information content of the data should be possible.

Furthermore, the method attempts to remove variations in the mean level of the speech signal which should reduce the effects of some speaker variation such as the differences in male, female and children's speech, caused by shifts in the frequency spectra (see Section 2.3.6 on page 15).

Firstly, a framework for the solution is presented, addressing the main criteria of the method. Then, the matching criterion is derived from first principles and, finally, developed into a recursive form facilitating sequential computation.
5.2 A Framework for the Solution

The main criteria which need to be addressed when developing the new method are based upon the original list of aims set out at the beginning of the thesis (Chapter 1). This forms a solution framework which defines the tasks addressed in the following chapters:

1. The primary objective of the method is to incorporate statistical dependence between frames in the speech models, within the template matching scheme.

2. The matching criterion should ideally be insensitive to the common forms of variability exhibited in speaker independent recognition (Section 2.3); in particular, the differences in the mean frequency levels of the utterances in the main speaker groups; males, females and children. Careful choice of parametric representation is thus also important.

3. It is necessary to develop a recursive form of the matching criterion which facilitates sequential computation and allows an updated criterion value to be calculated as and when new speech frames become available — a form of early decision algorithm (see Section 4.9.5).

4. A search strategy needs to be devised which optimizes the matching criterion.

5. A method of creating the full template models (inferencing the first and second order statistics — mean and covariance respectively) from suitable training data needs to be formulated.

6. Decisions on the method of segmentation, the choice of basic recognition unit and the size of templates need to be made.

Criteria 1–3 will be addressed in the following sections of this chapter, criterion 4 in Chapter 6, criterion 5 in Chapter 7 and criterion 6 in Chapter 8.
5.3 Problem Formulation

Conventionally, speech is considered as a set of frames which have been converted into, say, \( L \) parameters, often based in the frequency domain. These parameters may be MFCCs, formant frequencies, LPCs, or similar. Each parameter describes a sampled waveform in time, a stylised representation of which is shown in Figure 5.1.

![Graph showing speech frame parameters](image)

**Figure 5.1:** Conventionally, speech is considered as a set of frames, each consisting of \( L \) parameters often based in the frequency domain. Each parameter describes a sampled waveform in time.

The problem under consideration is that of comparing an unknown pattern, representing a segment of continuous speech, against a set of prototype patterns, or reference templates, producing a criterion function value for each match which gives a quantitative measure of the “goodness of fit”. The pattern, in this case, is comprised of speech frame parameters arranged in a vector, commonly called a pattern vector. Therefore, for \( L \) parameters, there will be \( L \) pattern vectors characterizing the utterance.

In the derivation below, just one parameter is considered initially. This means that we are considering just one waveform in Figure 5.1. Later on, the development generalizes to embody all parameters in such a way that any relationship which may exist between them is preserved.

In essence, speech template matching comprises pattern matching and time alignment/warping transformation activities. In formulating the problem, the pattern matching function is based upon classical Bayesian statistical decision theory yielding a familiar decision rule for normally distributed speech models corresponding to fixed-length utterances. Then, the problem of coping with different length utterances is considered...
by incorporating a time-axis warping transformation.

5.3.1 The Pattern Matching Problem

Suppose the dictionary of speech model templates contains \( m \) basic speech units (which may be words, phonemes, diphones, etc.). Then, the ultimate aim of the recognition system is to categorize an unknown speech input into one of these \( m \) classes. At the pattern matching level, this may lead to categorizing segments of speech into one class or providing a small number of candidate classes for higher level discrimination. For the moment, it will be assumed that each class is represented by one speech unit model alone although the technique can easily be extended to use multi-modal class models.

The waveform described by one parameter from the \( i \)-th class template can be represented as an \( N \)-dimensional vector of samples,

\[
P_i = \begin{bmatrix} \mu_{i1} \\ \vdots \\ \mu_{iN} \end{bmatrix} = [\mu_{i1}, \ldots, \mu_{iN}]^T. \tag{5.1}
\]

One should remember that this is not a speech frame. It is the time-varying sampled waveform of one parameter from each successive frame in the template.

Of course, speech is inherently variable in many aspects and thus a particular realization of a speech pattern, \( y \), from a given class, \( \omega_i \), will not be identical to the \( i \)-th class template model. It is therefore more appropriate to model each class of patterns by a distinct probability distribution. It is assumed that the distribution is Gaussian and that the template, \( \mu_i \), is the expected value

\[
\mu_i = \frac{1}{\Omega_i} \sum_{b=1}^{\Omega_i} y_b \quad y_b \in \omega_i \tag{5.2}
\]

where \( \Omega_i \) is the number of patterns in class \( \omega_i \) and \( y \) denotes an unknown pattern to be classified

\[
y = [y_1, \ldots, y_N]^T \tag{5.3}
\]

which describes the path of one of the \( L \) speech frame parameters in time. It is assumed, for the moment, that the dimensionality of the vector \( y \) (the length of the unknown utterance) is equal to that of the templates. Later, this will be extended to the more usual case for an utterance with arbitrary length.

Conventionally, as we have already said, in conventional DP approaches, speech frames are considered to be statistically independent and of equal variance — in other words, there are assumed to be no inter-frame dependencies. In this case, the Euclidean distance between an unknown input speech pattern vector and the \( i \)-th class template
can be used as a matching criterion. This assumption effectively implies that the i-th class covariance matrix, \( \Sigma_i \), is an identity matrix, \( \Sigma_i = I \). However, if inter-frame statistical dependence and unequal variances are taken into consideration the covariance matrix should be assumed to have a general form,

\[
\Sigma_i = \frac{1}{\Omega_i} \sum_{b=1}^{\Omega_i} (y_b - \mu_i)(y_b - \mu_i)^T.
\] (5.4)

Under the assumption of speech waveforms (sequences of frames) being normally distributed with a general covariance matrix, the Euclidean metric is not applicable. Instead, an appropriate matching criterion must be derived.

Let us define a few standard terms:

- \( P(\omega_r) \) — the a priori probability of the occurrence of class \( \omega_r \), usually assumed to be equal for all classes as a first approximation;
- \( P(\omega_r|y) \) — the a posteriori probability of the occurrence of class \( \omega_r \), given that \( y \) has already been observed;
- \( p(y|\omega_r) \) — the conditional class probability density function (PDF) of class \( \omega_r \)

\[
p(y|\omega_r) = \frac{1}{\sqrt{(2\pi)^N | \Sigma_r |}} e^{-\frac{1}{2} [(y-\mu_r)^T \Sigma^{-1}_r (y-\mu_r)]};
\] (5.5)

- \( p(y) \) — the mixture class PDF.

The Bayes minimum error decision rule [2] states that, for minimum classification error, pattern \( y \) should be assigned to class \( \omega_i \) if

\[
P(\omega_i|y) = \max_r P(\omega_r|y).
\] (5.6)

The Bayes equation for combining probabilities states that

\[
P(\omega_i|y) = \frac{P(\omega_i)p(y|\omega_i)}{p(y)}
\] (5.7)

and substituting equation (5.7) into equation (5.6) (note \( p(y) \) cancels) we obtain the decision rule — assign pattern \( y \) to class \( \omega_i \) if

\[
p(y|\omega_i)P(\omega_i) = \max_r p(y|\omega_r)P(\omega_r).
\] (5.8)

Substituting equation (5.5) into equation (5.8) we get,

\[
\frac{P(\omega_i)}{\sqrt{(2\pi)^N | \Sigma_i |}} e^{-\frac{1}{2} [(y-\mu_i)^T \Sigma^{-1}_i (y-\mu_i)]} = \max_r \frac{P(\omega_r)}{\sqrt{(2\pi)^N | \Sigma_r |}} e^{-\frac{1}{2} [(y-\mu_r)^T \Sigma^{-1}_r (y-\mu_r)]}.
\] (5.9)
Taking natural logarithms of both sides, multiplying by -2 and noting that a maximizing function becomes a minimizing function when multiplied by a negative quantity, we get the decision rule

\[ \text{assign } y \text{ to } \omega_i \text{ if } \]

\[ (y - \mu_i)^T \Sigma_i^{-1} (y - \mu_i) + \log |\Sigma_i| - 2 \log P(\omega_i) = \min_r ((y - \mu_r)^T \Sigma_r^{-1} (y - \mu_r) + \log |\Sigma_r| - 2 \log P(\omega_r)). \]  

(5.10)

Denoting the right hand side of equation (5.10) by \( J_r(y) \), a short form expression of the rule can be written

\[ y \rightarrow \omega_i \text{ if } J_i(y) = \min_r J_r(y). \]  

(5.11)

5.3.2 Time Alignment Transformation

So far, the dimensionalities of the unknown pattern and of the template have been assumed to be identical. In practice, of course, the unknown speech input will have a variable length and thus the vector representing it will have a variable size. Generally, it can be shown that utterances from class \( \omega_i \) will have the same general shape as the class template, but will be subject to non-linear time-axis deformation, seen clearly in Figure 5.2 which shows two utterances of the letter “b”. These variations manifest themselves locally as a set of timing differences and globally by an overall difference in utterance length.

The Decision Rule

The sampled version of the speech signal to be recognized is denoted by \( Z \). There is a large set, \( Y \), of possible candidate normalized waveforms \( y \in Y \) of dimensionality \( N \) that can be generated from \( Z \). Each normalized waveform \( y \) can be obtained by a suitable time-axis warping transformation \( W(Z) \) of the original signal, i.e.

\[ y = W(Z). \]  

(5.12)

The optimal normalized waveform \( y_{opt} \) is the one that yields the best match over all possible classes \( \omega_r \) and for all \( y \). Thus the rule to be used can be stated as

\[ \text{assign } Z \text{ to } \omega_i \text{ if } J_i(y_{opt}) = \min_r \min_y J_r(y). \]  

(5.13)

Consider now the matching criterion function \( J_r(y) \) from equation (5.10),

\[ J_r(y) = (y - \mu_r)^T \Sigma_r^{-1} (y - \mu_r) + \log |\Sigma_r| - 2 \log P(\omega_r). \]  

(5.14)
5.3: Problem Formulation

Figure 5.2: The letter "B" spoken by two different speakers clearly shows the global and local timing differences that exist despite the general shape of the utterances being similar.

Since the last two terms are independent of $y$, the minimization of $J_r(y)$ over all possible $y$ involves only

$$M_r(y) = (y - \mu_r)^T \Sigma_r^{-1} (y - \mu_r)$$

known as the Mahalanobis distance. The decision rule in equation (5.11) can be defined as

assign $Z$ to $\omega_i$ if $J_i(y_i) = \min_r J_r(y_r)$

where $y_r$ is the candidate pattern yielding the best match for class $\omega_r$, i.e.

$$M_r(y_r) = \min_y M_r(y).$$
Classification Procedure

The procedure for classification of waveform \( Z \) can thus be summarized as follows.

1. For each class \( \omega_r \) use the Mahalanobis distance (equations 5.17 and 5.15) to find the best time-warping transformation \( y_r \).

2. Select the most probable class according to the decision rule (5.16).

It is recalled that the entities \( y, \mu_i \) and \( \Sigma_i \) represent respectively the utterance, template and covariance matrix from one of \( L \) such entities characterizing the model for each basic recognition unit. Incorporation of all \( L \) parameter waveforms may be achieved in a number of ways. However, it should be borne in mind when choosing an approach that it is desirable to preserve any relationship that may exist between the waveforms in each model. In view of this, the full matching criterion for matching waveforms \( y^i, j = 1, \ldots, L \), to the \( i \)-th class model has been chosen as

\[
M(y) = \sum_{j=1}^{L} (y^j - \mu_{ij}^j)^T (\Sigma_{ij}^{-1}) (y^j - \mu_{ij}^j).
\]  

(5.18)

Computational Feasibility

Direct minimization of criterion \( M_r(y) \) over all possible time-axis warping transformations \( y = W(Z) \) involves an exhaustive search. In practice, such a search would not be computationally feasible. However, in Chapter 6 it will be shown that the optimal warping function can be found using the highly efficient \textit{branch and bound} algorithm.

For the moment, it is sufficient to note that any matching criterion used in conjunction with the branch and bound algorithm must be computable in a recursive manner. A recursive form of the criterion function is developed in Section 5.5 for this purpose.
Landmark

So far in this chapter, the development of the basic matching criterion has been presented, as defined by action (1) of the solution framework in Section 5.2. The essential features of the derivation so far can be reviewed as follows.

- The key problem has been described as that of matching unknown speech patterns to prototypes, or class models, called templates.

- Speech classes, $\omega_i$, are modeled by Gaussian distributions with mean, $\mu_i$ (the $i$-th class template), and general covariance, $\Sigma_i$.

- A classification procedure for assigning an unknown waveform $y$ to class $\omega_i$ was developed from the Bayes minimum error decision rule leading to the Mahalanobis distance being used as a matching criterion.

- Incorporation of the entire set of $L$ parameter waveforms characterizing each class has been achieved within the matching criterion, thus retaining any relationship that may exist between them.

Attention now focuses on the second part of the original solution framework; that of mean level compensation.

5.4 Mean Level Compensation

5.4.1 The Need For Mean Level Compensation

When considering time-alignment (or warping) and matching, the primary criterion tends to be the shape of the waveform. Although different speakers pronounce identical words differently, the basic shape of the waveform is assumed to be similar. This is the basis upon which the majority of speech recognition systems are designed, of course. However, variations between male, female and children's speech have been found to manifest themselves as shifts in mean level of the frequency components of each frame [1]. A change in the mean level does not affect the shape of the waveform, but most matching criteria are sensitive to it and the Mahalanobis distance is no exception. Without compensation, the criterion will treat two waveforms which differ solely in their mean levels as being two distinct speech entities. To observe the effect that shifts in the mean level of parameter frequencies can have, observe the recognition error performance graph in Figure 5.3. This shows the percentage error for a basic dynamic programming algorithm when applied to a speaker independent spoken letter task (letters “bdg”) in which a number of shifts in the mean level of the frame parameters
Figure 5.3: The effect of shifts in the mean level of the frame parameter frequencies on recognition error for a spoken letter ("bdg") task. The recognizer was a straightforward dynamic programming algorithm.

were applied. It can be seen that an improvement in recognition error has been possible by applying a mean level shift of 0.7. For this task, MFCCs were used. If formant frequencies are used as a parameterization, the problem is even more acute (see page 30 of Bladon [1]).

The effect of mean level differences could easily be removed if an exhaustive search procedure was employed to solve the minimization problem in equation (5.19). Then, the Mahalanobis distance would be computed directly for the complete waveform \( y^i \) from which its mean level had been subtracted. However, the dynamic time alignment procedure requires decisions to be made on a frame-by-frame basis in order to select the correct path. To do this, sequential mean level compensation is required.

A decision at each stage \( k \) about the constant level \( S_k \) to be subtracted from the current waveform, \( k \) frames long, needs to be made. Intuitively, it may seem appropriate to remove from the segment the mean level at the \( k \)-th stage. However, it will be seen that the situation is more complicated than this.
5.4: Mean Level Compensation

5.4.2 Compensation Scheme

The derivation of the compensation scheme will be confined to finding the best match under just one hypothesis, i.e. for just one class and for one given utterance. For this reason, the class index \( r \) will be dropped from \( M_r, \mu_r \) and \( \Sigma_r \). The aim is therefore to find, among all candidate time-axis warped waveforms, \( y \), that waveform \( y_{opt} \) which minimizes the Mahalanobis distance \( M \) between the unknown utterance and the template of the hypothesized class. Formally,

\[
M(y_{opt}) = \min_y M(y) \tag{5.19}
\]

where \( M(y) \) is given in equation (5.15).

The question arises of how to treat each of the \( L \) parameter waveforms which constitute a segment of speech? One approach would involve removing the appropriate mean level from each waveform regardless of that removed from the others. Another method would involve the removal of the same constant level from all component waveforms. For the latter of the two techniques, the resultant mean level compensation would be optimized over all waveforms, and yet, any inter-waveform information would be preserved.

Consider the stylised formant tracks for two hypothetical utterances in Figure 5.4. In such a case where two vowel-like sounds differ purely by their inter-waveform spacing, the former of the above approaches to mean level compensation would lead to the matching criterion yielding similar values for both utterances when matched.
The Matching Criterion

Figure 5.5: One way to compensate for shifts in the waveform mean level is to remove the current estimate of the waveform mean at each stage, \( k \).

with some similarly shaped template. However, if we employ the latter approach and thus only remove the same, globally optimal, mean from all component waveforms, inter-waveform spacing is preserved and these hypothetical utterances would not yield identical matching scores. For this reason, the latter technique is used in the compensation scheme, a development of which proceeds as follows.

Denoting the vector of the first \( k \) frames of \( y^j \) as

\[
y^j[k] = [y^j_1, \ldots, y^j_k]^T \quad j = 1, \ldots, L
\]

(5.20)

where \( y^j_n \) are the components of the pattern vector \( y^j \) in equation (5.3) (Figure 5.5). Similarly, \( \mu^j[k] \) denotes the partial template composed of the first \( k \) components of vector \( \mu^j \). Further, let \( \Sigma^j[k] \) be the covariance matrix in the \( k \)-dimensional space. The Mahalanobis distance between \( y^j[k] \) and \( \mu^j[k] \) can be written as

\[
M[k] = \sum_{j=1}^{L} \left( y^j[k] - \mu^j[k] \right)^T \left( \Sigma^j[k] \right)^{-1} \left( y^j[k] - \mu^j[k] \right).
\]

(5.21)

A constant signal (mean offset) \( S_k \) is then removed from all of the waveforms in the speech segment to generate the modified signal

\[
y^j[k]' = y^j[k] - S_k \cdot e[k]
\]

where \( e[k] \) is a \( k \)-dimensional unit vector (see Figure 5.5). The constant level \( S_k \) which gives the minimum value of the criterion function \( M[k] \) can now be found.
Dropping the \([k]\) notation for the moment, the Mahalanobis distance as a function of \(S\) can be written as

\[
M(S) = \sum_{j=1}^{L} (y^j - \mu^j)^T (\Sigma^j)^{-1} (y^j - \mu^j)
\]

\[
= \sum_{j=1}^{L} [y^j - S e - \mu^j]^T (\Sigma^j)^{-1} [y^j - S e - \mu^j].
\]  \hspace{1cm} (5.22)

Expanding,

\[
M(S) = \sum_{j=1}^{L} (y^j - \mu^j)^T (\Sigma^j)^{-1} (y^j - \mu^j)
+ \sum_{j=1}^{L} S^2 e^T (\Sigma^j)^{-1} e
- 2 \sum_{j=1}^{L} S e^T (\Sigma^j)^{-1} (y^j - \mu^j).
\]  \hspace{1cm} (5.23)

Differentiating 5.23 with respect to \(S\), the condition for a stationary point is obtained,

\[
\frac{\partial M(S)}{\partial S} = 2S \sum_{j=1}^{L} e^T (\Sigma^j)^{-1} e
- 2 \sum_{j=1}^{L} e^T (\Sigma^j)^{-1} (y^j - \mu^j) = 0.
\]  \hspace{1cm} (5.24)

Thus,

\[
S \sum_{j=1}^{L} e^T (\Sigma^j)^{-1} e = \sum_{j=1}^{L} e^T (\Sigma^j)^{-1} (y^j - \mu^j)
\]  \hspace{1cm} (5.25)

and rearranging, the optimum value of \(S\) becomes

\[
S = \frac{\sum_{j=1}^{L} e^T (\Sigma^j)^{-1} (y^j - \mu^j)}{\sum_{j=1}^{L} e^T (\Sigma^j)^{-1} e}.
\]  \hspace{1cm} (5.26)

This corresponds to a minimum of function \(M(S)\) as its second derivative at this point is positive

\[
\frac{\partial^2 M(S)}{\partial S^2} = 2 \sum_{j=1}^{L} e^T (\Sigma^j)^{-1} e \geq 0.
\]  \hspace{1cm} (5.27)

Substituting from (5.26) into (5.23) the minimum value of \(M(S)\) is found to be

\[
M(S_{opt}) = \sum_{j=1}^{L} (y^j - \mu^j)^T (\Sigma^j)^{-1} (y^j - \mu^j) - \frac{\left( \sum_{j=1}^{L} e^T (\Sigma^j)^{-1} (y^j - \mu^j) \right)^2}{\sum_{j=1}^{L} e^T (\Sigma^j)^{-1} e}.
\]  \hspace{1cm} (5.28)
Reintroducing the notation for describing $k$-dimensional quantities and denoting

$$\xi_k = e^T[k] \sum_{j=1}^{L} (\Sigma^j[k])^{-1} e[k] \quad (5.29)$$

$$\varepsilon_k = e^T[k] \sum_{j=1}^{L} (\Sigma^j[k])^{-1} (y^j[k] - \mu^j[k]) \quad (5.30)$$

the optimal value of the criterion function can be rewritten as

$$M[k](S_{opt}) = M[k] = \sum_{j=1}^{L} (y^j[k] - \mu^j[k])^T (\Sigma^j[k])^{-1} (y^j[k] - \mu^j[k]) - \varepsilon_k^2. \quad (5.31)$$

The last term in equation (5.31) can be considered as a mean level correction factor.

**Special Case: Identity Covariance Matrix**

It is interesting to note that if the covariance matrix is an identity matrix, i.e. $\Sigma^j[k] = I$ then

$$\varepsilon_k = k \cdot L \quad (5.32)$$

and

$$\xi_k = k \sum_{j=1}^{L} (\gamma^j[k] - \alpha^j[k]) \quad (5.33)$$

where $\gamma^j[k]$ and $\alpha^j[k]$ are the mean levels of the input speech waveform and template segments respectively,

$$\gamma^j[k] = \frac{1}{k} \sum_{b=1}^{k} y^j_b \quad (5.34)$$

$$\alpha^j[k] = \frac{1}{k} \sum_{b=1}^{k} \mu^j_b \quad (5.35)$$

and the correction factor then becomes

$$\frac{\varepsilon_k^2}{\xi_k} = \left[ \sum_{j=1}^{L} k(\gamma^j[k] - \alpha^j[k]) \right]^2. \quad (5.36)$$

Thus, mean level compensation is achieved, as expected, when the mean levels of the input speech and the template are identical.
Landmark

In this last section, action (2) in the outline framework has been addressed, summarized as follows:

- Mean level compensation aims to remove speaker-to-speaker variations, to which the Mahalanobis distance is sensitive, by removing a level $S_k$ from the speech waveforms. The primary criterion then becomes the shape of the waveform.

- The dynamic time alignment procedure requires decisions to be made on a frame-by-frame basis in order to select the correct path. To do this, sequential mean level compensation is implemented by removing a level $S_k$ from the speech waveform up to frame $k$.

- In the compensation process, intra-frame (frequency-based) relationships are preserved by removing the level $S_k$ from all component waveforms.

- The resulting expression for the mean-level compensated matching criterion value $M[k]$ up to the $k$-th frame is found to be the original uncompensated criterion function minus a correction factor, $\varepsilon_k^2/\xi_k$.

In order to minimize the above criterion (5.31), it is necessary to develop a recursive form (action 3).

5.5 Recursive Form of the Criterion Function

For an understanding of the requirements of a recursive form of the criterion function, consider the following scenario. The criterion function has been evaluated at the $k$-th stage of the recognition process. An extra frame is observed, $y^j[k+1], j = 1, \ldots, L$, necessitating a comparison of segments $y^j[k+1]$ and $\mu^j[k+1]$, where

\begin{align*}
y^j[k+1] &= \begin{bmatrix} y^j[k] \\ y_{(k+1)}^j \end{bmatrix} \quad \text{(5.37)} \\
\mu^j[k+1] &= \begin{bmatrix} \mu^j[k] \\ \mu_{(k+1)}^j \end{bmatrix}. \quad \text{(5.38)}
\end{align*}

Clearly, it would be advantageous if the new criterion function value at the $(k+1)$-st stage could be computed from the current value at stage $k$. Below, a scheme is developed for performing such a recursive calculation.
Let the new \((k + 1)\)-dimensional inverse covariance matrix be

\[
(\Sigma^j[k + 1])^{-1} = \begin{bmatrix}
\sigma_{11}^j & \sigma_{12}^j & \cdots & \sigma_{1(k+1)}^j \\
\sigma_{21}^j & \sigma_{22}^j & \ddots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
\sigma_{(k+1)1}^j & \cdots & \cdots & \sigma_{(k+1)(k+1)}^j
\end{bmatrix}
\]

Then, designate by \(\tilde{\sigma}^j_{(k+1)}\) the \(k\)-dimensional vector obtained from the \((k+1)\)-st column, \(\sigma^j_{(k+1)}\), by deleting the \((k + 1)\)-st element, \(\sigma^j_{(k+1)(k+1)}\).

\[
\tilde{\sigma}^j_{(k+1)} = [\sigma^j_{1(k+1)}, \sigma^j_{2(k+1)}, \ldots, \sigma^j_{k(k+1)}]^T.
\]

Then, we can write

\[
\sigma^j_{(k+1)} = \begin{bmatrix}
\tilde{\sigma}^j_{(k+1)} \\
\sigma^j_{(k+1)(k+1)}
\end{bmatrix}.
\]

In [2] (page 266–267) it is shown how \((\Sigma^j[k + 1])^{-1}\) can be easily obtained from \((\Sigma^j[k])^{-1}\) in the following manner

\[
(\Sigma^j[k + 1])^{-1} = \begin{bmatrix}
(\Sigma^j[k])^{-1} + \frac{\tilde{\sigma}^j_{(k+1)}(\tilde{\sigma}^j_{(k+1)})^T}{\sigma^j_{(k+1)(k+1)}} & \tilde{\sigma}^j_{(k+1)} \\
(\tilde{\sigma}^j_{(k+1)})^T & \sigma^j_{(k+1)(k+1)}
\end{bmatrix}.
\]

Recalling the criterion function from equation (5.31), the value at the \((k + 1)\)-st stage is given by

\[
M[k + 1] = \sum_{j=1}^{L} (y^j[k + 1] - \mu^j[k + 1])^T (\Sigma^j[k + 1])^{-1} (y^j[k + 1] - \mu^j[k + 1]) - \frac{\xi^j_{k+1}}{\xi^j_{k+1}}.
\]

where

\[
\xi_{k+1} = e^T[k + 1] \sum_{j=1}^{L} (\Sigma^j[k + 1])^{-1} e[k + 1]
\]

\[
\varepsilon_{k+1} = e^T[k + 1] \sum_{j=1}^{L} (\Sigma^j[k + 1])^{-1} (y^j[k + 1] - \mu^j[k + 1])
\]
From [2], using the recursive form of the \((k + 1)\)-dimensional inverse covariance matrix developed above, the individual terms of (5.43) can be expanded and rewritten in a recursive form as follows:

**Term 1:**

\[
\sum_{j=1}^{L} (y^j[k + 1] - \mu^j[k + 1])^T(\Sigma^j[k + 1])^{-1}(y^j[k + 1] - \mu^j[k + 1])
\]

\[
= \sum_{j=1}^{L} (y^j[k] - \mu^j[k])^T(\Sigma^j[k])^{-1}(y^j[k] - \mu^j[k])
\]

\[
+ \sum_{j=1}^{L} \frac{[(\sigma^j_{(k+1)})^T(y^j[k + 1] - \mu^j[k + 1])^2}{\sigma^j_{(k+1)(k+1)}};
\]

(5.46)

**Term 2:**

\[
\varepsilon_{k+1} = \text{tr} \left[ \sum_{j=1}^{L} (y^j[k + 1] - \mu^j[k + 1])e^T[k + 1](\Sigma^j[k + 1])^{-1} \right]
\]

\[
= \text{tr} \left[ \sum_{j=1}^{L} (y^j[k] - \mu^j[k])e^T[k](\Sigma^j[k])^{-1} \right]
\]

\[
+ \sum_{j=1}^{L} \frac{(\sigma^j_{(k+1)})^T(y^j[k + 1] - \mu^j[k + 1])e^T[k + 1]e_{(k+1)}}{\sigma^j_{(k+1)(k+1)}}
\]

\[
= \varepsilon_k + \sum_{j=1}^{L} \frac{(\sigma^j_{(k+1)})^T(y^j[k + 1] - \mu^j[k + 1])e^T[k + 1]e_{(k+1)}}{\sigma^j_{(k+1)(k+1)}}
\]

(5.47)

where \(\text{tr}[A]\) is the trace of matrix \(A\);

**Term 3:**

\[
\xi_{k+1} = \xi_k + \sum_{j=1}^{L} \frac{[(\sigma^j_{(k+1)})^Te[k + 1]]^2}{\sigma^j_{(k+1)(k+1)}}
\]

(5.48)

Introducing the simplifying notation

\[
\eta^i_{(k+1)} = (\sigma^i_{(k+1)})^Ty^i[k + 1] - \mu^i[k + 1])
\]

(5.49)

\[
\rho^i_{(k+1)} = (\sigma^i_{(k+1)})^Te[k + 1]
\]

(5.50)

and using (5.31), \(M[k + 1]\) in equation (5.43) can be expressed as
\[ M[k + 1] = M[k] + \frac{\xi_k^2}{\xi_k} + \frac{\varepsilon_{k+1}^2}{\xi_{k+1}} + \sum_{j=1}^{L} \frac{\eta_{(k+1)}^2}{\sigma_{(k+1)}^2} \]  \hspace{1cm} (5.51) \\

where, in terms of notation (5.49) and (5.50), \( \xi_{k+1} \) and \( \varepsilon_{k+1} \) can be computed recursively as

\[ \xi_{k+1} = \xi_k + \sum_{j=1}^{L} \frac{(\rho_{(k+1)}^j)^2}{\sigma_{(k+1)}^j} \]  \hspace{1cm} (5.52) \\

\[ \varepsilon_{k+1} = \varepsilon_k + \sum_{j=1}^{L} \frac{\eta_{(k+1)}^j \rho_{(k+1)}^j}{\sigma_{(k+1)}^j} \]  \hspace{1cm} (5.53) 

Since \( \xi_k \) and \( \Sigma_{(k+1)(k+1)}^j \) are positive, the right hand side of equation (5.51) will always be non-negative and the criterion function is therefore **monotonically increasing**. This is an essential property for use in a branch and bound algorithm.

Equation (5.51), fulfilling action (3), is the key to the new dynamic time alignment method. It is a recursive form of the original matching criterion (equation (5.31)) which facilitates the calculation of criterion values as and when new frames become available. This recursive form has an elegant feature, explained below, which allows efficient storage of the speech model statistics.

### 5.6 Speech Model Storage

Close inspection of the recursive criterion function expression (5.51) and its associated equations (5.49) - (5.53) will reveal a novel facet. It requires just the last column of the inverse covariance matrix \( (\Sigma^j[k])^{-1} \). Thus, to compute the criterion function for all \( N \) stages merely requires the storage of

1. \( N \) vectors with dimensionalities of 1, 2, 3, \ldots, \( N \) representing the last (\( k \)-th) column of \( (\Sigma^j[k])^{-1} \) as \( k \) increases,

2. the \( r \)-th class template, \( \mu_r^j \),

for each frame parameter, \( j = 1, \ldots, L \), and for each model class, \( \omega_r \), \( r = 1, \ldots, m \). Thus, the amount of storage space required for the covariance information for each class is reduced to just \( N(N + 1)/2 \) per parameter.

For simplicity, the \( r \)-th class special triangular model matrix of vectors from \( (\Sigma_r^j[k])^{-1} \) will be denoted \( \chi_r^j \), \( j = 1, \ldots, L \). Thus, for each class, speech models will
5.7: Summary

In this chapter, the matching criterion for a new method for dynamic time alignment of waveforms has been presented which is directly applicable to the problem of speaker independent continuous speech recognition. The key feature of the approach is that it utilizes the full discriminatory information content of the data by employing full co-variance matrices in the matching process. Recognition of unknown speech waveforms is achieved by template matching with a database of speech models comprised of both the first and second order statistics.

- Firstly, a solution framework was defined identifying the important criteria that the solution should meet.

- Then, the matching criterion was derived, within a Bayesian framework, to exploit inter-frame statistical dependencies.

- As the shape of the speech waveforms is believed to be an important feature, the criterion was developed to include a scheme for mean level compensation. Compensation (or removal) of the mean level differences between the unknown
input and the template acts to remove speaker-to-speaker differences which can cause problems.

- To allow sequential computation of the matching criterion, a recursive form of the formula is developed which incorporates an elegant method for storing the inverse matrices in the speech models. It is shown that the recursive formula thus derived is monotonically increasing with time (speech frame number) and is therefore suitable for use with a branch and bound minimization procedure.

In the next chapter, details of the method employed to implement and optimism the above matching criterion are presented.

References


6.1 Introduction

The matching criterion developed in the previous chapter allows the matching of a speech utterance against a reference template, taking into account inter-frame variances and removing mean frequency levels on a frame-by-frame basis.

In this chapter, an algorithm for implementing the matching criterion is suggested based upon the branch and bound algorithm. It allows the minimization of the criterion and yields the optimal time-warping transformation between the input and the template.

First, the reasons why a dynamic programming algorithm cannot be used are explained, essentially because the new matching criterion is not separable. Then, the search problem is defined. It is shown that an exhaustive search procedure is, naturally, computationally prohibitive and that a better approach is to use the computationally efficient branch and bound algorithm.

In order to tailor the branch and bound algorithm to the speech recognition task, some control strategies are suggested based on global and local constraints and thresholds.

In Section 6.6 the details of the implementation approach are explained with reference to an illustrative example. For those requiring specific algorithmic detail, a formal statement of the branch and bound algorithm is presented in Section 6.6.1.

Finally, a brief analysis is carried out of the factors involved in developing the new method in hardware.
6.2 Optimizing the Matching Criterion: Why Not Use Dynamic Programming?

The method of speech template matching by dynamic programming was described in Section 4.9 for finding the best match between an input speech signal and a set of templates. However, the DP algorithm cannot be used to implement the new matching criterion. The DP approach relies upon the Principle of Optimality which, we recall, states that the global optimal decision (or path) depends upon the initial state and upon making only optimal decisions at each intermediate stage. When minimizing a conventional criterion function, this principle holds because the function can be expressed in terms of additive components each of which is a function of measurement acquired in a single frame of speech — the criterion function is separable. At each intermediate stage, the DP algorithm computes a local Euclidean distance \( d(i, j) \) which is independent of the path chosen up to that particular stage. This is evident when examining the basic form of the recursive DP calculation from Section 4.9.3,

\[
g(i, j) = d(i, j) + \min \begin{cases} g(i - 1, j) \\ g(i - 1, j - 1) \\ g(i, j - 1) \end{cases} \tag{6.1}
\]

where \( i \) and \( j \) are frame indices for the input speech pattern and the template respectively.

Path decisions based on computing the new criterion function described in the previous chapter will be influenced by the current explored path. This can be seen more clearly by studying the terms in the recursive calculation of the criterion function which was derived in in Section 5.5,

\[
M[k + 1] = M[k] + \frac{\varepsilon_k^2}{\xi_k} + \frac{\varepsilon_{k+1}^2}{\xi_{k+1}} + \sum_{j=1}^{L} \frac{(\eta_{k+1}^j)^2}{\sigma^2_{(k+1)(k+1)}} \tag{6.2}
\]

where

\[
\eta_{k+1}^i = (\alpha_{k+1}^i)^T (y^i[k + 1] - \mu^i[k + 1]) \tag{6.3}
\]

\[
\rho_{k+1}^i = (\alpha_{k+1}^i)^T e_{k+1} \tag{6.4}
\]

\[
\xi_{k+1} = \xi_k + \sum_{j=1}^{L} \frac{(\rho_{k+1}^j)^2}{\sigma^2_{(k+1)(k+1)}} \tag{6.5}
\]

\[
\varepsilon_{k+1} = \varepsilon_k + \sum_{j=1}^{L} \frac{\eta_{k+1}^j \rho_{k+1}^j}{\sigma^2_{(k+1)(k+1)}} \tag{6.6}
\]
It can be seen that the terms $e_k$ and $e_{k+1}$ are functions of either $y[k]$, $y[k+1]$, $\mu[k]$ or $\mu[k+1]$. These measurements are acquired at every point along the speech waveform (up to frame $k$) and the resulting criterion function is not separable. In other words, its value at state $k$ cannot be expressed in terms of the value at stage $(k - 1)$ and a correction factor depending only upon the $k$-th stage. Thus, the dynamic programming algorithm cannot be used to minimize the criterion function.

Minimization of the criterion function over all possible paths could be achieved by an exhaustive search procedure. However, as we shall see, this is computationally prohibitive and so a much more efficient strategy is developed based upon the branch and bound algorithm.

### 6.3 The Search Problem

The general problem under consideration is that of matching a segment of connected speech, represented by vector $Z$ of dimensionality, $N_Z$, with a template $\mu$ of dimensionality $N$

\[
\mu = [\mu_1, \ldots, \mu_N]^T
\]

\[
Z = [Z_1, \ldots, Z_{N_Z}]^T.
\]

As we know, the number of frames, $N_Z$, of the candidate speech signal will differ from the nominal dimensionality $N$, as the duration of natural speech is subject to local and global variations. To make a fair comparison between $Z$ and $\mu$, the speech signal must be subjected to some form of time alignment transformation which compensates for timing differences between the two matched waveforms. In practice, the alignment process simply implies that for long utterances, $N_Z > N$, some components of vector $Z$ are omitted while for short utterances, $N_Z < N$, some are repeated\(^1\).

### 6.3.1 Computational Complexity of Exhaustive Search

In principle, the optimal match between $Z$ and $\mu$ can be found by exhaustive search. That is, all possible warping transformations can be found which map $Z$ onto $\mu$ and the best one (i.e. the one that achieves the lowest matching criterion value) selected.

When dealing with a speech signal, only an admissible subset of transformations is considered, rather than considering all possible mappings. This subset reflects the constraints applied to the speech template matching problem (Section 4.9.4). This means that, for each frame in the template $\mu$, only $\ell$ frames from the candidate signal are considered in the warping transformation.

---

\(^1\)As well as repeating frames, frame insertion can be carried out by interpolation.
Figure 6.1: The constrained warping transformation process can be considered as a tree search problem where, for each frame in the template, there are a number of possible frames in the input signal which it can be matching with. These are called successor pairs which are located at successor nodes.

The transformation process can be considered as a tree search problem, a portion of which is shown graphically in Figure 6.1. As with most tree searches, the black dots are called nodes, which are connected by branches, and $\ell$ is called the branching factor.

In order to assess the computational complexity of such a search process, suppose that the $k$-th stage of the warping transformation has been reached. That is, a match between the $k$-th frame, $\mu_k$ and the $p$-th frame, $Z_p$, of the input speech has been made. $\mu_k$ and $Z_p$ denote the $k$-th and $p$-th components of vectors $\mu$ and $Z$ respectively and $p \geq k$. The possible successor pairs of corresponding frames are constrained to $[\mu_{k+i}, Z_{p+i}]$, $i = 2, \ldots, \ell$ (see Figure 6.2). In other words, there are $\ell$ possible frame-pair matches that can be considered for the $(k + 1)$-st stage of the warping process. Each of the candidate pairs the $(k + 1)$-st stage has again $\ell$ possible successors that can be selected for the $(k + 2)$-nd stage, and so on.

It is apparent that the number of possible combinations grows exponentially as a function of the stage of the warping transformation. Constraining the first frame of the template always to be associated with the first frame of the matched waveform, then at the second stage there will be $\ell$ candidate frame-pairs, in the 3rd stage, $\ell^2$ pairs, and so on. In the $N$-th stage, the number of $(N - 1)$-tuples of the frame-pairs, $\beta$, will have grown to

$$\beta = \ell^{(N-1)}. \quad (6.9)$$
6.4: The Branch and Bound Algorithm

Figure 6.2: A match between the k-th frame, \( \mu_k \), and the p-th frame, \( Z_p \), of the input speech has been made. The possible successor pairs of corresponding frames are constrained to \([\mu_{k+i}, Z_{p+i}]\), \( i = 2, \ldots, \ell \).

Clearly, even for a moderate size problem, an exhaustive search would not be computationally feasible. In the following section, a more efficient search method will be introduced.

6.4 An Introduction to the Branch and Bound Algorithm

Traditional implementations of the dynamic time alignment process use dynamic programming to reduce the number of explored nodes in the search tree. To minimize the matching criterion presented here, a tree search algorithm is required which reduces the number of nodes to be explored and, unlike the DP algorithm, also allows backtracking, for the reasons outlined in Section 6.2. The branch and bound algorithm meets these requirements.

The branch and bound algorithm is a depth-first tree search technique which guarantees to find the optimal solution (warping transformation) and yet explores much fewer nodes than an exhaustive search procedure. The only proviso is that the matching criterion function being minimized is monotonically increasing with the depth of the tree which, in Section 5.5 (page 88), it was shown to be. The reason for this will become clear.

The basic idea in the branch and bound algorithm is to prune branches and subtrees which, as a result of the monotonicity property, will never lead to an optimal solution. Consider the example search tree displayed in Figure 6.3.

In this example, \( \ell \), the number of successor nodes, has been set to 2, and the number of levels (the number of frames in the template) \( N \) set to 4. The total number of warping transformations that would be evaluated if an exhaustive search was carried out is \( 2^3 = 8 \). Each branch of the tree represents one of these transformations and each node stores the corresponding cumulative matching criterion value \( M[k, p] \) for the
Figure 6.3: Example tree search using the branch and bound algorithm.

particular path, where $k$ is the template frame index (the level of the tree) and $p$ is the input speech frame index of that particular match.

Suppose the tree is traversed along the right-most branch first calculating the criterion function value $M[k, p]$ at each stage, $k$, (nodes 1, 2 and 3). At each level, $\ell$ nodes are expanded corresponding to matches between the first $k$ frames of the template with $\ell$ sequences of input speech frames. At each level, explored nodes are arranged in ascending order from right to left. When the leaf node is reached (node 3), the criterion function value $M_B$ is stored as the current best value. Then, the search backtracks up the tree to the nearest unexplored node (node 2) and then moves forward (downward) along the right-most branch evaluating the criterion value at node 4. Back-tracking occurs again (to node 2 and node 1) and then forward to the next right-most branch, evaluating $M[k, p']$.

Now, the criterion function has already been shown to be monotonically increasing with the depth of the tree. That is, the value of $M[k, p']$ increases as the tree is descended. Also, the branches originating from any node of the tree are searched in ascending order of the criterion function value at its successor nodes. So, the search only continues forward if the current value $M[k, p']$ is less than the current best $M_B$. Otherwise, if $M[k, p'] > M_B$, the current search path should be terminated because $M[k, p']$ will never decrease as more nodes are explored (monotonicity property). Hence the path currently being evaluated will never lead to a better match than the one responsible
for achieving the best match score \( M_B \).

When the search eventually reaches another leaf node, at which \( M[k, p'] < M_B \), \( M_B \) is updated with the value at this leaf node and this becomes the new best path. This is the essence of the branch and bound algorithm. The search continues in a similar fashion until there are no more valid nodes to expand.

On page 105, a formal statement of the algorithm is given but, for now, attention focuses on the control of the search strategy by means of global and local constraints.

### 6.5 Control of the Search

The branch and bound algorithm, as it stands, is not tailored to the speech problem. It does not make use of the nature of the speech signal and may, therefore, allow unnecessary searches. Also, it does not provide a simple means of obtaining early decisions or fast, slightly sub-optimal, decisions. In short, it does not incorporate any control. Control can be implemented, among other ways, through the use of global and local constraints (Section 4.9.4) and by thresholding. From Section 4.9.4, we recall that the roles of constraints are:

- **A Global Constraint** to regulate the overall allowable stretching and compression;

- **Local Constraints** to govern local decisions such as the allowable number of features to skip (continuity), the maximum number of predecessors in the cumulative distance calculation and the preservation of the natural order of events (monotonicity);

Thresholds, on the other hand, offer a different form of control which is particularly useful when the process is performing either very well or very badly. The degree of control obtainable from thresholds has the advantage of being adjustable. Unfortunately, they are often difficult to set correctly and are sensitive to changes in the environment. It should be noted that many control strategies result in a sub-optimal algorithm.

In this section, some strategies are suggested which should allow sufficient control of the branch and bound algorithm without severe loss of optimality.

#### 6.5.1 Global Constraints

The implementation of the matching criterion described so far is intended to be applied in a connected speech recognition environment. Under this assumption, the global control of the search procedure will not be governed directly by the word length. In other words, if the input speech is assumed to contain several words, a particular match
between a template and a candidate input will not be expected to terminate at the ends of both speech signals.

Therefore, the global constraint on the branch and bound algorithm is simply an overall stretching and compression factor. This factor is governed by the range of variability expected in the utterance rate of the original speech. The task of quantifying this global constraint relies upon determining the dynamic range of utterance rates over which the algorithm is to work.

As we know, similar utterances will be spoken at different rates by different speakers. Thus, the dimensionality (length) of a particular segment of speech, \( Z \), will depend upon the utterance rate of the original speech; the faster the speaker, the smaller the number of frames representing a speech entity and visa versa. In order to design a speech recognition system which can cope with a dynamic range of utterance rates defined by the ratio of maximum and minimum numbers of frames, \( N_{Z_{\text{min}}} \) and \( N_{Z_{\text{max}}} \) respectively, it is helpful to define some simple waveform parameters. The frame repeat rate, giving information regarding the fastest speaker, can be defined as

\[
R_R = \frac{N}{N_{Z_{\text{min}}}}
\]  

and a comparative measure of slowest utterance with the standard template as

\[
R_s = \frac{N_{Z_{\text{max}}}}{N},
\]

the speech record length factor where \( N \) is the template length. While the record length factor \( R_s \) can, in general, be any real number greater than unity, a practical frame repeat rate \( R_R \) must be an integer, as only whole frames can be repeated or inserted.

As the speaker utterance rate is not known beforehand, we must consider a speech record of at least \( R_s N \) frames to cater for the slowest possible speaker. The simplest way to cope with a fast speaker is to either repeat each frame \((R_R - 1)\) times or interpolate between frames and insert \((R_R - 1)\) frames per frame.

The product, \( R_R R_s \), is an alternative representation of the dynamic range of a non-linear time-axis warping algorithm. In practice, the dynamic range of warping transformations should be an integer since the number \( \ell \) of the possible successor pairs can only be an integer number. Therefore, \( \ell \) should be set so as to satisfy

\[
R_R R_s + 1 > \ell \geq R_R R_s.
\]  

In this implementation, a pre-processed pattern, \( Z' \), is generated from the first \( N \) frames of the raw input speech pattern \( Z \). Linear interpolation is performed between these \( N \) frames which inserts \((R_R - 1)\) frames per frame and results in a pre-processed pattern of \( \ell N \) frames in length (see Figure 6.4). Pattern matching operates between the template \( \mu \) and the pre-processed pattern \( Z' \).
6.5: Control of the Search

6.5.2 Local Constraints and Thresholds

Local constraints aim to control the low-level decision making process at each step in the branch and bound search. As with DP local constraints, their form endeavours to exploit knowledge about the speech signal and about the type of warping transformation that is most likely to occur in the context of speech recognition. In addition, local constraints are used to keep the computational overhead down to a feasible level without compromising the performance of the algorithm.

In this branch and bound implementation of the matching criterion, six local constraints and thresholds are employed: a successor node excursion restriction, a monotonicity condition, a continuity condition, a path weighting, a "bad-path" rejection threshold and a "good-path" acceptance threshold.

Successor Node Excursion Restriction

The successor node excursion restriction defines the maximum number of successor nodes $\ell' \in (1, \ell)$ which can be expanded from any given node in the search tree. An upper limit value $\ell$ is assigned to this restriction which is equivalent to the branching factor $\ell$ in the previous analysis on page 93.

In order to model the characteristics of the speech signal more realistically, $\ell'$ is constrained dynamically depending upon the number of frames, $I_{i-1}$, skipped at the previous $(i - 1)$-st stage. In this way, some form of continuity is introduced into the search path which discourages sharp transitions in the warping function.
This local constraint performs essentially two tasks. It aims to

- avoid the expansion of too many nodes at any level in order to restrict the maximum number of frames which may be skipped. This is closely related to the global stretching and compression factor;

- reduces the complexity of the search space by limiting the branching factor at each level of the tree.

**Monotonicity Condition**

A monotonicity property aims to preserve the natural order of events. In this implementation, this simply means that, for any path down the search tree, the frame index $p$ increases.

**Continuity Condition**

The continuity condition $I_{\text{min}}(I_{i-1})$ defines the minimum number of frames to be skipped at the $i$-th level in the search tree. Like the successor node restriction, it is also set dynamically depending upon the number of frames which were skipped at the $(i-1)$-st level (see Table 6.1). The primary purpose of this constraint is to ensure that neither too few nor too many frames are skipped at each point in the warping transformation which would lead to a discontinuous warping function.

**Path Weighting**

The successor node excursion restriction and continuity condition ensure that the search tree expands in a sensible fashion and also discourage large transitions in the warping function. To further influence the search, an optional *path weighting* can be introduced,

$$M[k, p] = M[k, p] + w(I_i)$$

(6.13)

where the weights $w(I_i)$ are a function of the frame index increment, $I_i$.

This is particularly useful when successive frames have similar values and therefore will produce similar matching criterion scores. By weighting a desirable path

<table>
<thead>
<tr>
<th>Frame index increment, $I_k$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of successor nodes, $\ell'$</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Minimum skip, $I_{\text{min}}$</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.1: The maximum number of successor nodes at each level and the minimum allowable skip are governed by the number of frames skipped at the previous level. Typical values are shown here for frame increments of 1 to 5.
score slightly in favour of other paths, sufficiently dissimilar matching scores arise which limit the number of unnecessary path explorations. A desirable path might be that which favours choosing frames which are as close to the current point in the input utterance and, conversely, an undesirable path is one which skips over a number of frames.

The "Bad-Path" Rejection Threshold and the "Good-Path" Acceptance Threshold

Rejection Threshold  In some cases, a particular search path will be expanded which incurs a high matching criterion value — i.e. a bad match is attempted. This situation could arise, for example, at the outset of a search, when no "best-match score" exists or when two dissimilar utterances are matched. In such cases, a number of unproductive paths may need to be explored before the algorithm settles onto a good path if, indeed, one exists.

To avoid this situation occurring, a "bad-path" rejection threshold $T_r(k)$ is incorporated into the search procedure. This involves simply examining the criterion value at each node and terminating the search if the value exceeds $T_r(k)$. This threshold increases as a function of the depth of the tree, $k$.

Acceptance Threshold  In a similar vain to the rejection threshold, some searches produce excellent matching scores very early on. This may occur, for example, when two very similar utterances are compared. In such a case, it seems pointless to continue with the tree search simply to reach the optimum matching score. It may, after all, be only slightly better than score achieved via the initial paths which may have been good enough to make a fairly accurate decision.

Implementation of this idea is straightforward consisting of a comparison of the matching criterion value at a leaf node with an acceptance threshold, $T_a$. If the criterion value is less than $T_a$, the search is terminated and the current path is stored as the best match warping transformation.

Effect of Acceptance and Rejection Thresholds on the Speed of Execution  It will be seen in Chapter 8 that judicious choice of the acceptance and rejection thresholds can greatly reduce the average computation time for a typical match without seriously affecting the recognition performance. Clearly, the global optimality of the algorithm no longer exists but the practical advantages will often outweigh this deficiency.
6.6 Details of the Branch and Bound Implementation

In this section, the specific details of the branch and bound search procedure are explained taking the basic idea outlined in Section 6.4 and adding the control strategies described above. The intention is to show how the implementation works in practice when applied to real speech signals and to provide a stepping stone to the more formal statement of the algorithm below. Indeed, the reader need not study the formal algorithm unless specific algorithmic detail is required as all the salient details are mentioned here.

Let us assume that the matching process is currently at stage \( k \). That is, the first \( k \) components of the template have been matched against some portion of the input speech signal. This corresponds to being at level \( k \) in the branch and bound search tree. In order to illustrate this explanation more clearly, we will construct a quantitative example and assume \( k = 3 \). We will denote, also, the index of the speech pattern component paired with template component \( \mu_k \) by \( g^k_{s(k-1)} \) with index \( s(k) = \ell' \leq \ell \) being the number of candidate sequences at the \((k+1)\)-st level. It is a function of the number of frames skipped at the previous level, \( I_k \) — the successor node excursion restriction — see Table 6.1.

In order to continue the matching procedure, it is necessary to evaluate the match-
6.6: Implementation Details

In terms of the tree search, extending candidate sequences corresponds to expanding \( \ell' \) successor nodes \([k + 1, q_r^{k+1}], r = 1, \ldots, \ell'\), where the matching score for each candidate match is stored. A portion of the search tree showing this expansion step is displayed in Figure 6.6.

Let \( \ell' = 3 \) and define an active node at level \( k \), \( p = q_r^{k-1} = q_2^3 \) corresponding to the current best-match warping transformation up to level \( k \) (level 3) where \( s(k - 1) = s(2) \) specifies the number of unexplored branches at the 3rd level.

The input speech frame indices \( q_r^{k+1} \) are determined from the current active node:

\[
q_r^{k+1} = p + I_i
\]  

(6.14)

where \( I_i \) is a frame index increment, \( I_i \in (I_{\text{min}}(I_{k-1}), \ell) \). The order of the frame index increments \( I_i, i = 1, 2, \ldots, \ell' \), should be such that the matching scores for the successor nodes are arranged in ascending order,

\[
M[k + 1, p + I_1] \leq \cdots \leq M[k + 1, p + I_i] \leq \cdots \leq M[k + 1, p + I_{\ell'}].
\]  

(6.15)

If path-weighting is used (Section 6.5.2), the criterion values \( M[k + 1, p + I_i] \) at node

Figure 6.6: A portion of the branch and bound search tree showing the evaluation of \( \ell' \) successor nodes.
$[k + 1, p + I_i]$ will be modified to

$$M[k + 1, p + I_i]' = M[k + 1, p + I_i] + w(I_i)$$  \hspace{1cm} (6.16)$$

where the weights are a function of the frame index increment, $I_i$.

Now, assuming the tree is traversed along the right-most branch, the next step is to compare the largest new criterion function value, $M[k + 1, q_{k+1}^4] = M[4, q_3^4]$ with the best match so far defined as $M_B$. If $M[4, q_3^4] < M_B$ then depth-first tree generation takes place to the next level — i.e. $k = k + 1 = 4$. A new active node $q_3^4$ is set and three more nodes are evaluated as shown in Figure 6.7.

![Figure 6.7: If the largest successor node value $M[4, q_3^4]$ is less than the best match $M_B$ then depth-first tree generation takes place to the next level. However, if (as here) $M[k, q_{k+1}^4] > M_B$ we know that no other $k$-th level node will achieve a lower value of $M[k, p]$ and thus the algorithm backtracks and prunes the remaining paths at the $4$th level.]

To prevent the expansion of very bad paths, the option to reject the current path is provided by comparing $M[4, q_3^4]$ with the rejection threshold, $T_r(k + 1)$. 
Suppose, after this expansion step, that the largest successor node value (the rightmost node value) $M[5, q_{4}] \geq M_{B}$. Then, since the criterion function is monotonically increasing with the depth of the tree, a better value than $M_{B}$ will not be found along the current path. Also, since the successor nodes have been arranged in ascending order, no other currently explored criterion value at this level will be less than $M_{B}$. The other paths at this level are pruned by decrementing the number of unexplored nodes at the 4th level, $s(3) = s(3) - 1 = 2$ and the search backtracks to the previous level $k = k - 1 = 3$. Now, the criterion function value at node $[4, q_{2}]$ is compared with $M_{B}$: $M[4, q_{2}] \geq M_{B}$?

The search process proceeds in a similar fashion — depth-first tree generation, expand nodes and compute criterion function, check against bound $M_{B}$, backtrack and explore along right-most branch — until there are no more valid nodes to expand. At a leaf node, the option to accept a (possibly) sub-optimal path is provided by comparing $M_{B}$ with the acceptance threshold, $T_{a}$; if $M_{B} < T_{a}$, the algorithm is terminated. When all the valid nodes are explored, the warping transformation of the current best match $q_{s(k)}^{k+1}$, $k = 0, 1, \ldots, (N - 1)$ is stored.

Below, a formal statement of the algorithm is provided which facilitates a software implementation. The reader not interested in the formal details may proceed to Section 6.7 without loss of continuity.

### 6.6.1 Formal Statement of the Branch and Bound Algorithm

#### The Branch and Bound Algorithm

**Step 1. Initialization**

Initialize the template index, $k$, the input speech index, $q_{s(k-1)}^{k}$, the number of un-pruned successor nodes $s(k - 1)$, the "bad-path" rejection threshold, $T_{r}(k)$, and the best match criterion function value, $M_{B}$ as follows:

\[
\begin{align*}
    k & = 1 \\
    q_{s(k-1)}^{k} & = 1 \\
    s(k - 1) & = 1 \\
    T_{r}(k) & = T_{r_{\text{min}}} \cdot k \\
    M_{B} & = \infty.
\end{align*}
\]
Step 2. Successor Node Determination

Given the upper-limit value of the number successor nodes to be expanded, \( \ell \), set the maximum number of successor nodes to expand at this stage to \( \ell' \in (1, \ell) \) using Table 6.1.

Given also the local constraint \( I_{\min(k-1)} \) defining the minimum number of frames to skip, then:

Determine \( \ell' \) successor nodes, \([k + 1, q_r^{k+1}], \ r = 1, \ldots, \ell'\)

where \( q_r^{k+1} \) satisfies the local constraint

\[
q_{s(k-1)}^k + I_{\min(k-1)} \leq q_r^{k+1} \leq q_{s(k-1)}^k + \ell'
\]

and

\[
M[k+1, q_r^{k+1}] \leq \cdots \leq M[k+1, q_r^{k+1}] \leq \cdots \leq M[k+1, q_l^{k+1}]
\]

where

\[1 \leq r \leq \ell'.\]

Set the number of unpruned branches, \( s(k) = \ell' \).

Step 3. Check Bound for the Candidate Successor Node

If \( s(k) = 0 \) then go to Step 5.

If the current criterion value is greater than the best match value,

\[
M[k + 1, q_{s(k)}^{k+1}] \geq M_B
\]

or the cumulative path score exceeds the "bad-path" rejection threshold,

\[
M[k + 1, q_{s(k)}^{k+1}] > T_r(k + 1)
\]

then go to Step 5.

Step 4. Increment: Depth-first Tree Generation

Set \( k = k + 1 \). If \( k = N \) then go to Step 6. Otherwise, go to Step 1.

Step 5. Backtracking

Set \( k = k - 1 \). If \( k = 0 \) or \( M_B < T_a \), the acceptance threshold, then terminate the algorithm. Otherwise, \( s(k) = s(k) - 1 \).

Go to Step 3.
Step 6. Bound Updating

Set

\[ M_B = M[k + 1, q_{a(k)}^{k+1}] \]

Store the indices \( q_{a(k)}^{k+1}, k = 0, \ldots, N-1 \) of the speech vector components giving the current best match and return to Step 5.

6.7 Development In Hardware

6.7.1 Arithmetic Operations

At present, development of speech recognizers in hardware is undertaken largely on digital signal processors (DSPs). These devices are optimized for fast multiplications and common signal processing schemes such as pipelining, etc.

When implementing an algorithm on a DSP, it is useful to know the number of additions, multiplications and divisions that the algorithm performs in the basic recursive operation. Divisions should be avoided if possible as they can take several machine cycles to perform. However, subtractions are can be considered to be equivalent to additions.

The basic recursive calculation in this new matching scheme is

\[
M[k + 1] = M[k] + \frac{\varepsilon_k^2}{\text{Term1}} + \frac{\varepsilon_{k+1}^2}{\text{Term2}} + \sum_{j=1}^{L} \frac{(\eta_{k+1}^i)^2}{\text{Term3}}. \tag{6.17}
\]

Parameters \( \varepsilon_k \) and \( \xi_k \) need no calculation as they would have been stored at the previous level. \( \varepsilon_{k+1} \) and \( \xi_{k+1} \) are computed recursively:

\[
\varepsilon_{k+1} = \varepsilon_k + \sum_{j=1}^{L} \frac{\eta_{k+1}^j \rho_{k+1}^j}{\sigma_{(k+1)(k+1)}^j}
\]

\[
\xi_{k+1} = \xi_k + \sum_{j=1}^{L} \frac{\rho_{k+1}^j)^2}{\sigma_{(k+1)(k+1)}^j}
\]

where

\[
\eta_{k+1}^i = (\sigma_{k+1}^i)^T (y^i[k + 1] - u^i[k + 1]) \tag{6.18}
\]

\[
\rho_{k+1}^i = (\sigma_{k+1}^i)^T e_{k+1}.
\]
The terms $\eta^j_{k+1}$ and $\rho^j_{k+1}$ need only be calculated once for each evaluation of the matching criterion. So, for each parameter, $\eta^j_{k+1}$ takes $(k + 1)$ subtractions and $(k + 1)$ multiplications to calculate and $\rho^j_{k+1}$ requires $(k + 1)$ additions, where $k$ is the template frame index (tree level).

**Term 1** To evaluate Term 1 requires just 1 multiplication and 1 division as $\epsilon_k$ and $\xi_k$ have already been calculated.

**Term 2** If $L$ is the number of parameters per frame, then to calculate $\epsilon_{k+1}$ and $\xi_{k+1}$ each require $L$ multiplications, $L$ divisions and $L$ additions giving a total for Term 2 of

- $2L$ multiplications,
- $2L$ divisions and
- $2L$ additions.

**Term 3** Term 3 requires

- $L$ multiplications,
- $L$ divisions and
- $(L - 1)$ additions

to calculate.

Combining these terms takes another 3 additions giving the total number of operations required to evaluate the criterion function at the $k$-th stage:

- $L(k + 4) + 1$ multiplications,
- $3L + 1$ divisions,
- $L(k + 1)$ subtractions and
- $L(k + 4) + 2$ additions.

### 6.7.2 Storage

As we mentioned above, intermediate values of $\epsilon_k$ and $\xi_k$ are stored to avoid unnecessary calculations. At each stage, $t'$ nodes are explored and the corresponding criterion function value calculated. It seems sensible to store these values and use them in the back-tracking stages. The total storage requirements for the matching criterion can therefore be broken down into:
6.8 Summary

A search strategy for implementing and minimizing the new matching criterion has been developed in this chapter. The main points of discussion were as follows:

- The new matching criterion is not separable and thus the tradition DP algorithm cannot be used to minimize it.
- The general problem involves template matching and time alignment — an exhaustive search procedure could be used but is computationally prohibitive.
- The branch and bound algorithm, which allows backtracking, guarantees to find the optimal warping transformation and is computationally efficient.
- In order to tailor the search strategy to the speech recognition problem, control is incorporated by adding global and local constraints and thresholds.
- The details of the method are described with reference to an illustrative problem followed by a formal statement of the algorithm.

To conclude, the branch and bound algorithm allows the elimination of the unproductive searching of many candidate warping transformations without compromising the global optimality of the solution yielded by the algorithm.
Chapter 7

Inferencing of Model Statistics

7.1 Introduction

The speaker-adaptive, statistically independent speech template matching criterion developed so far requires an accurate set of speech class models before it can be used. These models are slightly more complex than those used in conventional DP approaches as they need inter-frame variance information in the form of special inverse covariance matrices.

In this chapter, the basic strategy for inferencing these models is proposed. It is shown how this preliminary approach creates linearly dependent matrices which create problems in the matching process. Through experiments, these problems are exposed and a solution proposed based on eigenvector analysis. An improved inferencing strategy is suggested which is used in Chapter 8 to test the new matching scheme.

7.2 Preliminary Inferencing Strategy

The basic inferencing strategy follows a simple procedure which should give models for each class of speech being represented. These models consist of the class template, \( \mu \) and the special matrix \( X \).

The strategy produces the models by iteratively normalizing (warping) each of the training utterances to a “current-best” set of reference models and then averaging these normalized waveforms for each class (see Figure 7.1). At the beginning of the process, a particularly “good” example of an utterance from each class is selected and hand-normalized to make a starting template of the desired length. A first estimate of the model matrix \( X \) is chosen to be the identity matrix. Afterwards, the columns of matrix \( X \) are calculated by taking the last column of the \( k \times k \), \( k = 1, \ldots, N \) inverted sub-matrices taken from the top left-hand corner of the full \( N \)-dimensional covariance matrix \( \Sigma \). The inferencing procedure operates over the \( L \) frame parameters separately.
7.2: Preliminary Inferencing Strategy

Current Template/Model Matrix

- TI

Training Utterances

- time-align

Length-normalized Utterances

utterance 1
utterance 2
utterance 3
utterance ω

New average template/model matrix

Figure 7.1: The speech model inferencing strategy produces the models by iteratively normalizing the length of each of the training utterances to a "current-best" set of reference models and then averaging these normalized waveforms for each class.
Inferencing terminates either when classification scores converge for the training set or the desired number of iterations have been performed. The preliminary inferencing strategy is shown in Figure 7.2.

![Flow diagram of the preliminary speech model inferencing process.](image)

**Figure 7.2: Flow diagram of the preliminary speech model inferencing process.**

### 7.3 Problems with the Preliminary Inferencing Strategy

The inferencing strategy in the previous section does display some problems when tested on real data. These problems manifest themselves when the inversion of the covariance matrix take place. It is believed that as the statistical dependencies between
the template and the utterance are gradually built into the covariance matrices with each iteration, linear dependencies occur. This effect can be illustrated by looking at the eigenvalues of the covariance matrix.

In Figure 7.3 the training characteristics of this inferencing strategy are shown for the spoken letter “b” in a speaker-independent context. In the upper graph, the two smallest eigenvalues of an eight-dimensional covariance matrix are shown decreasing to small values (top) over just four iterations of the training data. Not surprisingly, this gives rise to an increasingly poor recognition error score (bottom graph) when the models are used in a “b,d,g” recognition task. Obviously, this problem must be overcome for the proposed method to work.

### 7.4 An Improved Inferencing Strategy

From the previous experiment, it seems as though the basic problem with the preliminary inferencing strategy lies in the linear dependencies that are generated in the class covariance matrices. Having identified this, it would seem obvious to remove these dependencies by monitoring the behaviour of the eigenvalues of each class matrix and removing the offending rows and columns from the matrix. However, once these rows and columns are removed, the reduced-dimensionality matrix which remains is not suitable for use in the method described.

The mismatch between the length of the template and the size of the reduced-dimensionality matrix can be resolved by either reducing the length of the template or by reconstituting a full-dimension matrix. Of these solutions, reduction in template length is probably the least satisfactory because, for example, if the matrix dimensionality is reduced at every iteration, the template length and matrix size run the risk of being reduced to excessively small values.

So, in this section, the problem of reconstructing a full-rank covariance matrix from a reduced-rank version is addressed.

Consider an $N \times N$, rank-deficient covariance matrix, $\Sigma$, generated from the inferencing strategy above. The corresponding eigenvalues and eigenvectors of $\Sigma$ are calculated and those eigenvectors with eigenvalues approaching zero are discarded\(^1\).

---

\(^1\)The point at which an eigenvalue is said to have reached “zero” is discussed later.
Figure 7.3: As the statistical dependencies between the template and the utterance are built into the covariance matrices with each iteration, linear dependencies occur. This can be seen here as the two smallest eigenvalues of the covariance matrix decrease to zero (top) giving rise to an increasingly poor recognition error score (bottom) [results are for the first MFCC of the spoken letter “b” in a “b,d,g” recognition task].
The remaining $K$ non-zero eigenvalues are arranged in a $N \times K$ matrix, $U$

$$U = \begin{pmatrix}
e_{11} & e_{21} & \cdots & e_{K1} \\
e_{12} & e_{22} & \cdots & e_{K2} \\
\vdots & \vdots & \ddots & \vdots \\
e_{1N} & e_{2N} & \cdots & e_{KN}
\end{pmatrix}. \quad (7.1)$$

Using this matrix $U$, the covariance matrix can be transformed into the reduced-dimensionality ($K \times K$) space matrix, $V$,

$$V = U^T \Sigma U. \quad (7.2)$$

Thus, we have a new, reduced-dimensionality matching criterion function,

$$(x - w)^T V^{-1} (x - w) \quad (7.3)$$

where, given the unknown speech data, $y$, and the template, $\mu$,

$$x = U^T y \quad (7.4)$$
$$w = U^T \mu. \quad (7.5)$$

Substituting equations (7.4) and (7.5) into equation (7.3), the full-rank matching criterion can be recovered,

$$(y - \mu)^T \underbrace{UV^{-1}U^T}_{Y^{-1}} (y - \mu). \quad (7.6)$$

The matrix $UV^{-1}U^T$ is the reconstituted version of the original rank-deficient inverse covariance matrix. This new matrix is denoted as

$$Y^{-1} = U [U^T \Sigma U]^{-1} U^T. \quad (7.7)$$

Once created, $Y^{-1}$ is used instead of $\Sigma^{-1}$ for creating the class model matrix, $X$. See Figure 7.4 for a illustrative view of this process.
Let eigenvalue threshold, $t_v = 0.3$

cut out eigenvectors with eigenvalues $< t_v$
to produce $U$ matrix

\[
K \left[ U^T \right] \times N \left[ \Sigma \right] \times \left[ U \right]_N \rightarrow K \left[ V \right]
\]

Invert

\[
N \left[ U \right] \times K \left[ V^{-1} \right] \times \left[ U^T \right]_K \rightarrow N \left[ V^{-1} \right]
\]

the replacement for $\Sigma^{-1}$

Figure 7.4: The problem of rank-deficient matrices can be overcome by employing this inverse covariance matrix reconstruction technique based on eigenvector analysis.
7.4.1 Control of Inverse Covariance Reconstruction Technique

There are a number of issues to consider before applying the inverse covariance reconstruction technique:

- At what point should the technique be introduced — i.e. how small do the eigenvalues have to be before their corresponding eigenvectors are discarded?

- What happens when more than one eigenvalue approaches zero?

- To form the special model matrix $X$, inverses of the sub-matrices of $\Sigma$ need to be calculated. Should the reconstruction technique be applied to all sub-matrices or only those which are over a given size?

- How should any threshold be modified for variations in sub-matrix size?

Let us consider each point in turn.

From the graphs on page 114, it can be seen that the recognition scores are deteriorating long before the smallest eigenvalues reach zero. This suggests that the reconstruction technique should be applied before the inferencing procedure breaks down. One of the easiest methods of determining the point at which to introduce the technique is to set some eigenvalue threshold, $t_v$, and to run exhaustive tests in which $t_v$ is varied and the effect on recognition performance observed. The results of this test (in Chapter 8 on page 139) show that there is a small preferred range of values for $t_v$ — too small and the inferencing strategy has already passed "crisis-point", too large and the technique removes too much information from the covariance matrix.

It is quite possible that several of the eigenvalues of the covariance matrix approach zero simultaneously. It seems sensible, therefore, that the improved inferencing strategy should discard more than one eigenvector if necessary. However, the strategy should ensure that, in extreme cases for example, enough eigenvectors are retained to allow adequate reconstruction.

When forming the $X$ matrix, if any of the sub-matrices of $\Sigma$ are rank-deficient, then the inverse is difficult to obtain. So, the improved inferencing approach is applied to all sizes of sub-matrix inversion, not just the final stage.

Thresholds, $t_v$, need to be modified with sub-matrix size, $k$, as the normalized eigenvalues will be different. The relationship between eigenvalue and sub-matrix size is inversely proportional and so a simple $\frac{1}{k}$ scaling of the threshold is performed.
7.4.2 The Improved Inferencing Strategy

The improved inferencing strategy overcomes the problem outlined in section 7.3 regarding the creation of rank-deficient covariance matrices. Before creating the special model matrix, \( X \), and at each stage, \( k \), it checks the eigenvalues of each covariance sub-matrix and applies the reconstruction technique developed above if the eigenvalues fall below the weighted threshold, \( \gamma \). In some cases where full-rank covariance matrices are formed, direct inversion can be performed. The complete and revised inferencing strategy thus becomes:

---

Model Inferencing Strategy
---

**Step 1. Initialize Models**

Set iteration counter, \( i = 1 \).

Choose a single utterance from each of the \( m \) speech classes in the training set to be the initial template, \( \mu_r^{[i]} \), \( r = 1, \ldots, m \).

Create the upper triangle of an identity covariance matrix to be the initial inverse matrix, \( X_r^{[i]} \), of each of the \( m \) speech class models.

Set the eigenvalue threshold, \( t_e \).

**Step 2. Initialize Class Index, \( r = 1 \)**

**Step 3. Initialize Utterance Counter**

Set utterance counter, \( t = 1 \), and clear new template, \( \mu_r^{[i+1]} = 0 \cdot I \).

**Step 4. Normalize**

Normalize training utterance \( y_r(t) \) with the branch and bound algorithm using \( \mu_r^{[i]} \) and \( X_r^{[i]} \) as references to produce normalized pattern, \( \tilde{y}_r(t) \).

Store \( \tilde{y}_r(t) \) for calculation of covariances.

Add \( \tilde{y}_r(t) \) to the new-template accumulator:

\[
\mu_r^{[i+1]} = \mu_r^{[i+1]} + \tilde{y}_r(t).
\]
If there are more utterances in this class, i.e., if \( t < \Omega_r \), the number of patterns in class \( r \), then increment, \( t = t + 1 \) and repeat this step otherwise continue.

**Step 5. Calculate New Class Template**

Calculate and store the new averaged class template,

\[
\mu_r^{[i+1]} = \frac{\mu_r^{[i+1]}}{\Omega_r}.
\]

**Step 6. Calculate New Covariance Matrix**

\[
\Sigma_r^{[i+1]} = \frac{1}{\Omega_r} \sum_{t=1}^{\Omega_r} (\hat{y}_r(t) - \mu_r^{[i+1]}) (\hat{y}_r(t) - \mu_r^{[i+1]})^T.
\]

Let the sub-matrix size, \( k = 1 \).

**Step 7. Inverse Covariance Reconstruction Decision**

Let the notation \( A[row, col] \) mean the sub-matrix in the upper right hand corner of matrix, \( A \). Compute the eigenvalues and eigenvectors of the \( k \times k \) sub-matrix, \( \Sigma_r^{[i+1]}[k, k] \). If the smallest eigenvalue is smaller than the scaled threshold, \( \frac{t_k}{k} \), then use the inverse covariance reconstruction technique — go to Step 8 — otherwise, go to Step 9 and do not apply the reconstruction technique.

**Step 8. Perform Inverse Covariance Reconstruction of Covariance Matrix**

Remove all eigenvectors of the \( k \times k \) sub-matrix, \( \Sigma_r^{[i+1]}[k, k] \), which have corresponding eigenvalues below \( \frac{t_k}{k} \) to produce the reduced-dimensionality eigenvector matrix \( U \) (see previous derivation). Disallow removal of more than \( (k - 1) \) eigenvectors. Then, perform the reconstruction technique described above (page 113) to obtain the inverse matrix, \( \Sigma^{-1} \).

Copy \( \Sigma^{-1} \) to \( (\Sigma_r^{[i+1]}[k, k])^{-1} \).

Go to Step 10.

**Step 9. Invert Sub-Matrix**

Invert the \( k \times k \) sub-matrix of \( \Sigma_r^{[i+1]} \) to form the new matrix, \( (\Sigma_r^{[i+1]}[k, k])^{-1} \).
Step 10. Create New $X^{[i+1]}_r$ Matrix

Copy the $k$-th column of $(\Sigma^{[i+1]}_r[k,k])^{-1}$ to the $k$-th column of the new model matrix,

$$X^{[i+1]}_r[c,k] = (\Sigma^{[i+1]}_r[c,k])^{-1} \quad c = 1, \ldots, k.$$  \hspace{1cm} (7.8)

If the sub-matrix size $k < N$, the full matrix dimension, then increment $k = k + 1$ and go to Step 7, otherwise store the new model matrix $X^{[i+1]}_r$ and continue.

If there are more classes, increment $r = r + 1$ and go to Step 3, otherwise continue.

Step 11. Performance Evaluation

Test the models by classifying the entire training set using the branch and bound algorithm with $\mu^{[i+1]}_r$ and $X^{[i+1]}_r$, $r = 1, \ldots, m$, as class models.

Step 12. Termination

If the recognition performance has not converged or the desired number of iterations has not been reached, increment $i = i + 1$ and go to Step 2, otherwise terminate the iteration process.

A flow diagram for this inferencing process is shown in Figure 7.5.
An Improved Inferencing Strategy

7.4: An Improved Inferencing Strategy

- Initialize models with seed utterance and identity covariance
- Normalize length of current training utterance using branch & bound algorithm with current template model
- Store normalized utterance and add to template accumulator
- Any more training utterances?
  - Yes
  - No
- Calculate average template from accumulator and compute covariance matrix
- Let sub-matrix size, \( k = 1 \)
- Find eigenvalues/eigenvectors of \( k \times k \) sub-matrix of covariance
- Smallest eigenvalue < \( t_v \) ?
  - No
  - Yes
- Perform reconstruction to obtain new inverse covariance
- Invert sub-matrix
- Copy k-th column of inverse covariance matrix to model matrix \( X \)
- \( k < N \) ?
  - Yes
  - No
- More classes?
  - Yes
  - No
- Evaluate performance
- Reached required performance? (Yes/No)
- Terminate

Figure 7.5: Flow diagram of the speech model inferencing process.
7.5 Summary

In this section, an inferencing mechanism for creating the models needed to apply the new template matching scheme was presented.

- Firstly, a preliminary inferencing strategy was suggested which iteratively normalized and averaged the training data to produce updated model statistics, $\mu$ and $\Sigma$.

- A problem arose with this approach as linear dependencies were introduced into the covariance matrix making the calculation of the matching criterion increasingly inaccurate.

- A solution was suggested which involves transforming the covariance matrix into a lower-dimensional space using an eigenvector matrix in which the linearly dependent eigenvectors have been removed. A full-rank version of the inverse covariance matrix is recovered by reusing the eigenvector matrix.

- This inverse covariance reconstruction technique is controlled by means of an eigenvalue threshold, carefully chosen by experiment, governing those eigenvectors which are to be used.

- The section concludes with a detailed description of the complete and revised inferencing strategy.

In the following chapter, experiments are described which aim to validate the new speech template matching technique.
Chapter 8

Experiments & Results

8.1 Introduction

In this chapter, some speech recognition experiments are described which test the new speech template matching scheme, the results of which validate the method, show a marked improvement over a dynamic programming approach and suggest further courses of research.

These experiments are carried out on data comprised of nearly 8000 utterances of the 26 letters of the British English Alphabet spoken by 104 speakers, roughly half of which are male and the other female. The speakers, taken from four age groups, provided three repetitions of each utterance — the first two used for training and the remainder for testing. The results of these experiments reveal information concerning:

- the extent of the matching error correlations between frames in a speech signal and the improvement in recognition performance obtained from using these correlations in a template matching scheme,

- the relative difference in performance for consonant and vowel sounds — consonant sounds showing the greatest improvement over a DP approach,

- the effect of the various control strategies upon recognition performance and speed of execution,

- the behaviour of the template inferencing strategy,

- the effect of varying training set size on performance.
8.2 The Experimental Data

All experiments described in this Chapter have been performed on a database of spoken utterances of the letters of the British English Alphabet. This was prepared by British Telecommunications PLC (BT) at their research laboratories in Martlesham Heath, Ipswich, UK (BTRL) as part of the BT Connectionist Project. The author would like to thank BT for authorizing the use of this data and for allowing the use of an abridged version of the documentation to create the following description of the data.

Recording

The database was recorded at BTRL using BT employees as test speakers. It is comprised of speech from a total of 53 male and 51 female speakers and there are three repetitions of each letter from each speaker. Recording was undertaken in a silent cabinet using a high quality handset, sampled at 20kHz, and digitally converted using a 16bit A/D converter and stored on disk.

End-Pointing

Speakers were prompted in a random order to say each letter. After the prompt appeared, two seconds of data was recorded to disk. When this data-gathering exercise was complete, an automatic end-pointing routine was used to obtain putative end-points for each utterance. Each (end-pointed) utterance was then checked by a human operator and the end-points adjusted if necessary. Any "bad" utterances were discarded at this stage (an utterance was marked "bad" if the wrong word was said or if an utterance was clipped by the recording window). After this process was completed, a total of 7976 utterances remained. BT state that there are known to be a small number of utterances (less than 10) in which an end-pointing error occurred.

MFCC Analysis

The Mel-frequency cepstral coefficient (MFCC) database (Section 3.4.1) used in the following experiments was prepared from the end-pointed time-domain data. Firstly, a number of frames were calculated between the end-points of each utterance (see Section 3.3). Each frame was 512 samples long (25.6ms) with successive frames overlapped by 256 samples (12.8ms). Then, to form MFCCs, each frame was pre-emphasized, windowed using a Hamming window and an FFT performed. From the resulting discrete magnitude spectrum, the energy values in 26 overlapping Mel-spaced frequency bands
were calculated (Section 3.4.1). The MFCCs are obtained by applying a discrete cosine transform to the logarithm of these energy values. For this database, each frame is characterized by 8 MFCCs. The zeroth MFCC (related to the frame energy) is not included in the parameterization.

It is recalled that three repetitions of each letter were recorded. In the following experiments, the first two repetitions of utterances are used for training with the remaining utterance used for testing. Each speaker was characterized by sex (m/f) and age group (age-range code 1-4).

8.3 The Algorithms & Experimental Procedure

In the following experiments, two algorithms are used to classify utterances from the BT database; the branch and bound algorithm with full-covariance matching criterion and a dynamic programming (DP) algorithm for comparative purposes. Both algorithms are implemented in Fortran 77 on a Sun4 workstation (SparcStation). The branch and bound software implements the formal statement of the algorithm described in Section 6.6.1 while the DP software is based upon a Viterbi algorithm [2, 1] and is similar to the schemes described in Sections 4.9.3 and 4.9.5.

To allow a fair comparison between the branch and bound and DP algorithms, a mean-level compensation scheme has been added to the DP software. This feature is implemented by iteratively shifting each MFCC frame value in the input speech by a small amount and performing a classification for each shift value. The decision procedure chooses the best match which results from each shifted utterance. This method is not ideally suited for implementation in hardware or for efficient software implementation. However, it does enable the effects of statistical dependency in the branch and bound method to be fairly compared with a statistical independent approach, such as dynamic programming, without the improvement in recognition performance achieved by mean-level compensation being included.

Creation of template speech models for each class (speech model inferencing) for the branch and bound method is carried out as described on page 118. For the DP method, inferencing is similar except that a DP algorithm is used in the normalization and classification steps (Steps 4 and 11 respectively) and that, obviously, no creation of the model matrix X or the covariance matrix $\Sigma$ is necessary. Thus, the inferencing scheme for the DP experiments becomes:
DP Template Inferencing Strategy

**Step 1. Initialize Templates**
Set iteration counter, $i = 1$.

Choose one utterance from each of the $m$ speech classes in the training set to be the initial template, $\mu_r^{[i]}$, $r = 1, \ldots, m$.

**Step 2. Initialize Class Index, $r = 1$**

**Step 3. Initialize Utterance Counter**
Set utterance counter, $t = 1$, and clear new template, $\mu_r^{[i+1]} = 0 \cdot I$.

**Step 4. Normalize**
Normalizing the length of training utterance $y_r(t)$ with the DP algorithm using $\mu_r^{[i]}$ as a reference template to produce the normalized pattern, $\tilde{y}_r(t)$.

Add $\tilde{y}_r(t)$ to the new-template accumulator:

$$\mu_r^{[i+1]} = \mu_r^{[i+1]} + \tilde{y}_r(t).$$

If there are more utterances in this class, i.e. if $t < \Omega_r$, the number of patterns in class $r$, then increment, $t = t + 1$ and repeat this step, otherwise continue.

**Step 5. Calculate New Class Template**
Calculate and store the new averaged class template,

$$\mu_r^{[i+1]} = \frac{\mu_r^{[i+1]}}{\Omega_r}.$$ 

If there are more classes, i.e. if $r < m$, then increment $r = r + 1$ and go to Step 3, otherwise continue.

**Step 6. Performance Evaluation**
Test the new templates by classifying the entire training set using the DP algorithm with $\mu_r^{[i+1]}$, $r = 1, \ldots, m$, as class templates.
Step 7. Termination

If the recognition performance has not converged or the desired number of iterations have not been reached, increment $i = i + 1$ and go to Step 2, otherwise terminate the iteration process.

8.3.1 Basic Recognition Unit & Template Selection

In Section 5.2, it was mentioned that decisions on the method of segmentation, choice of basic recognition unit and the template size need to be made. Earlier in the thesis, a review of the various basic recognition units was presented (Section 2.4.2) outlining the relative merits of each unit. For these early studies presented here into the effects of inter-frame correlations, an attempt to segment the database into basic linguistic units (such as phonemes, tri-phones, etc.) was thought unnecessary, and unwise, too. As mentioned previously, these segmentation schemes introduce boundary decision errors themselves. Indeed, the new matching scheme itself, with further development, may suggest more suitable basic recognition units which do not map directly onto traditional linguistic entities.

Given that the data is already segmented into discrete word units, whole-word template matching would seem be a logical task to perform. However, whole word recognition is not performed for two reasons. Firstly, it is found that the time taken to match whole word templates against spoken letter utterances is too long when implemented in a high-level language for both the branch and bound and dynamic programming methods. Secondly, this work is intended to be applied to the problem of connected continuous speech recognition (CSR) in the long term. In Section 4.9.5 the differences between the search strategies for CSR and for isolated word recognition were highlighted. One difference is that little or no attempt is made to fix the matching process to end-points in the input speech waveform. With the finite speech record length data available, it is not possible to perform whole-word CSR as the matching process would soon run out of input frames.

The approach taken in these experiments is to generate a template by hand which models some feature at the beginning of each group of spoken letters — the initial portions of the consonants “b”, “d” and “g”, for example. The chosen feature is selected by observing distinctive characteristics of the MFCC waveforms. The boundaries of these features are not rigidly fixed. However, it should be possible to extract simi-
lar boundaries by an automatic technique similar to that employed to create acoustic sub-word units (Section 2.4.2). In practice, this approach gives a template of around eight frames in length and is usually chosen to be the first eight frames of speaker 1, repetition 1 for each speech class (letter) taken from training set (see Figure 8.1).

![Graph](image)

**Figure 8.1:** For the first iteration of the inferencing process, a seed template of around eight frames in length is made from the first eight frames of speaker 1, repetition 1 for each speech class (letter) taken from training set (this example is a letter "b").

**A Note on the Presentation of Results:** The results presented in the forthcoming section are set out in question-and-answer format. The questions are posed in a straightforward and simple manner. The answers bring out the essential aspects of the new matching criterion — recognition performance, speed, etc.
8.4 Matching Error Correlations Between Frames

Question: Are there any correlations in the matching errors between frames in a speech signal?

To answer this, over 400 utterances of the letter “e” were normalized using the inferencing scheme described on page 118. The covariance matrix of these normalized utterances was computed exactly as described in the scheme. In Figure 8.2 (left) this covariance matrix is shown as a grey-level image of squares — each square is a matrix location, its value being represented as a shade of grey (black for small values, white for large).

From this figure, it is obvious that quite strong and long-range correlations do exist in speech. It is also worth noting that, for this vowel sound, the correlations are of a similar strength along the entire length of the template (top-left to bottom-right).

As above, over 400 utterances of the letters “b” and “d” (consonant sounds), “e” and “u” (vowel sounds) were normalized and their covariances calculated. In Figure 8.3 it can be seen that the covariances of the consonant sounds show less broad correlations at the beginning of the utterance (top-left) when compared with the vowel
sounds. This is as one would expect — it is also worth noting that correlations exist across a wide number of frames for the latter portions of the consonant sounds.

Figure 8.3: The extent of the correlation between frames seems to be slightly different for each sound type — vowel sounds and consonant sounds — as these four covariance matrices show. At the very beginning of each consonant sound (top-left of the matrix), correlations are weaker between adjacent frames but this grows as the utterance progresses. Covariances are shown for the first MFCC value only.
Question: Does the use of a general covariance matrix in the matching criterion improve recognition performance? Furthermore, how does the performance vary for consonant sounds and vowels?

The new matching scheme described in the previous chapters aims to exploit the statistical dependence between frames. This dependence has been shown to exist in the covariance matrices on the previous pages. The hypothesis put forward in Chapter 1 for the thesis was that better acoustic level recognition performance could be achieved with such a matching criterion.

To test this hypothesis, a set of three experiments were constructed using the alphabet database. The aim of these experiments was to create template models of the letters “b”, “d”, “g” (experiment 1), “p”, “t”, “v” (experiment 2) and “a”, “e”, “i”, “o”, “u” (experiment 3) and use them to classify the test set utterances from the same classes. Examples of each utterance are shown in Figures 8.4 and 8.5. The best results from these experiments are tabulated below.

<table>
<thead>
<tr>
<th>Recognizer</th>
<th>Incorrectly Classified Utterances (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;bdg&quot;</td>
</tr>
<tr>
<td>Dynamic Programming</td>
<td>16.6</td>
</tr>
<tr>
<td>Branch &amp; Bound (identity cov.)</td>
<td>16.1</td>
</tr>
<tr>
<td>Branch &amp; Bound (full cov.)</td>
<td>12.5</td>
</tr>
</tbody>
</table>

Table 8.1: A comparison of the number of incorrectly classified utterances for three template matching schemes. The table shows the results of separate experiments which attempted to recognize the spoken letters “bdg”, “ptv” and “aeiou”. The figures shown are the minimum values obtained over at least 20 iterations of the entire training and testing data sets.

The figures shown are the best values which were obtained after at least twenty iterations of the entire training database. For consonant sounds, they show a significant improvement in recognition error over the conventional dynamic programming approach. It can be seen also that results for the branch and bound algorithm with identity covariance are similar to DP, which is to be expected as both techniques assume statistical independence between frames.

The new approach shows only marginal improvement when tested on the vowel sounds “a”, “e”, “i”, “o” and “u”. When the vowel waveforms in Figure 8.5 are compared with the consonant sounds in Figure 8.4, the most striking difference is the lower dynamic range and fewer transients present in the vowels, particularly in the first few MFCCs. This may suggest that frame-to-frame correlations are most effectively used on transient signals. For vowel sounds, it does not seem as though these correlations provide significantly more discriminatory information. Indeed, recognition scores are
Figure 8.4: Initial portions of utterances of the spoken letters "b", "d", "g", "p", "t" and "v" taken from the BT database showing how the MFCC values change with time. Notice how the consonant sounds show greater dynamics than the vowel sounds overleaf (key on next page).
Figure 8.5: Central portions of the letters “a”, “e”, “i”, “o” and “u” taken from the BT database showing how the MFCC values change with time.
quite respectable for all methods when tested on the vowels which suggests that the maximum discriminatory information is being extracted from vowels anyway.

The most important conclusion, however, which can be drawn from the entire set of results is that the new matching criterion which includes statistical dependence between frames does allow improved recognition scores at the acoustic level and, for most sounds, provides significant improvement — up to 34% improvement in the tests described.

8.5 Control Strategies & Speed of Execution

Question: Can the acceptance/rejection thresholds be used to speed up the branch and bound algorithm without severe loss of performance? How does it compare with DP?

Question: What effects do the acceptance and rejection thresholds have individually on the speed of execution?

In Chapter 6 some strategies were introduced to control the branch and bound search algorithm and tailor it to the speech problem. It is recalled that an acceptance threshold was designed to detect “good paths” and a rejection threshold used to detect “bad paths”, both of them terminating the search early and thus preventing unnecessary tree searching. Of course, both thresholds remove the algorithm’s global optimality which can be seen in Figure 8.6 as the error rate worsens when the influence of each threshold increases.

However, the main reason for using these thresholds is to control the execution time of the branch and bound algorithm. This is most useful for hardware implementation or real-time software applications as the speed/performance characteristics can be defined for the given problem. In Figure 8.6 (top), the effect of varying the reject threshold on these characteristics can be seen. Execution times are shown per utterance and record the number of seconds of CPU time used on a Sun SparcStation 490. Absolute values of execution time are not important here as the algorithm is not optimized for speed. The important point to note from these results is the relative increase in speed which is possible by increasing the acceptance threshold. Indeed, execution times can be lowered by around 40% without severe performance deterioration.

For the rejection threshold (Figure 8.6 (bottom)), improvement is no less dramatic — notice how recognition scores are still below 20% error rate even for a near-50% decrease in execution time.

In Figure 8.7, a comparison of execution times for the branch and bound and dynamic programming algorithms is shown. Improvement in recognition performance
Figure 8.6: The effect of varying the acceptance (good path) threshold on recognition performance and execution time is quite dramatic. Execution times can be reduced significantly without the severe degradation of performance that might be expected. Effective use of the rejection (bad-path) threshold can also produce significant improvements in execution speed without damaging the recognition performance except for values of rejection threshold below approx. 8.
comes somewhat at the expense of execution speed. However, through judicious choice of the acceptance and rejection thresholds, it is possible to increase the speed of execution with only a marginal worsening of the recognition error.

![Graph showing recognition error versus execution time for training and test data with Branch and Bound algorithm compared to Dynamic Programming algorithm.](image)

**Figure 8.7:** It is possible to increase the speed of execution of the branch and bound template matching scheme through judicious choice of the rejection and acceptance thresholds (the acceptance threshold only is employed in this graph). Although the branch and bound algorithm is not optimized for speed, it can be seen that execution time can be increased by around 40% before the recognition performance falls below that of the DP algorithm (all execution times given in seconds of CPU time on a Sun SparcStation 490).

**Question:** What effect do path weights have upon recognition performance?

In Section 6.5.2 it was suggested that unnecessary searching may occur when successive frames have similar values. This is because the matching criterion will generate similar scores in the search tree encouraging more tree exploration than is necessary. To avoid this, the idea of a path weight was suggested which adds a small value to each matching score, $M[k,p]$, evaluated by extending successor nodes (equation 6.13),

$$M[k,p] = M[k,p] + w(I_i).$$

Path weights can be arranged so as to encourage small or large skips in the input speech signal by setting small weights closer to or further from the current frame respectively. A particular set of weights can be called a profile. In Figure 8.8, a number...
Figure 8.8: The number of frames skipped at each stage can be artificially influenced by using path weights. The results of some experiments on letters "b", "d", and "g" using different weight profiles are shown here suggesting that a degree of path-weighting can yield improved classification performance.
of profiles have been tested for a range of weight values. Although the effects of path-weighting are not dramatic, marginal improvement in recognition error can be seen to be possible.

8.6 Creation of Templates

Question: When creating the templates, at what point should the inverse covariance reconstruction (ICR) technique be introduced? Is there a desirable range of values for the ICR eigenvalue threshold?

It is recalled that the creation of the special model matrix, $X$, is hampered by rank-deficiency in the full covariances $\Sigma$ created during the template inferencing process (Section 7.3). To overcome this problem, a novel technique was developed on page 113 which reconstructed the rank-deficient covariance matrices from the eigenvectors of the original covariance matrix. The amount of reconstruction which takes place is governed by the eigenvalues of $\Sigma$ — small eigenvalues encourage more reconstruction from eigenvectors. The point at which the reconstruction technique is introduced is controlled by an eigenvalue threshold, $t_v$. The relationship between $t_v$ and the recognition error is shown in Figures 8.9 and 8.10 for the three experiments, "bdg", "ptv" and "aeiou".

![Figure 8.9: The effect of varying the eigenvalue threshold for a "ptv" spoken letter recognition task.](image-url)
Figure 8.10: The effect of varying the eigenvalue threshold at which the inverse covariance matrix reconstruction (ICR) technique is introduced can be seen for a speaker-independent "bdg" (top) and "aeiou" (bottom) recognition tasks. Clearly the best eigenvalue threshold lies within a small range — 25-45 for "bdg" and 13-30 for "aeiou". For large threshold values, performance decreases.
Ideally, $t_v$ should be set for each utterance class individually. However, in these experiments, the same value for $t_v$ was used for all utterances within the group (see "Further Research" on page 146).

It is clear from these results that the correct value for $t_v$ lies within a particular range which is, unfortunately, different for each sound group. When $t_v$ is too small, the covariance matrices still contain too many linearly-dependent rows and columns. On the other hand, a large value of $t_v$ deletes too many rows and columns from $\Sigma$ and it no longer represents the data accurately.

**Question:** How does the recognition performance vary with iteration?

Creation of class template models is conducted by iteratively normalizing the length of the training database with the current template and then averaging the resultant utterances (Section 7.4.2). Intuitively, one might expect this process to generate better templates with each iteration with convergence to a stable model. In fact, the process quickly produces good template models in the first few iterations. Furthermore, recognition performance reduces if training continues for too long (Figure 8.11). If the ICR technique were optimized for each class of utterance, then the training characteristics might exhibit better convergence.

![Figure 8.11: To create good class models, it seems sensible to iterate the inferencing process many times over the training database. Greater improvement in the performance of the templates occurs in the first few iterations.](image)
Question: How does the size of the training database affect the performance of the templates?

In Figure 8.12, the effect of training database size on recognition error is shown. As one might expect, performance is improved when a larger number of training tokens are used. Owing to the limited number of utterances in the BT database, it is not possible to predict how performance improves with very large numbers of training tokens although the trend in Figure 8.12 would suggest that larger training sets may be highly desirable.

![Figure 8.12: The performance of speech models depends upon the number of training utterances used to infer the speech models.](image)

8.7 Summary

In this chapter, some speech recognition experiments are described to test the new speech template matching scheme and provide some results which validate the method and suggest further courses of research. The main results are as follows:

- Matching error correlations exist between frames in a speech signal. These correlations vary for the different types of speech;

- Through experiments on a selection of spoken letters, the incorporation of these correlations into a template matching scheme has been shown to significantly
improve the recognition performance;

- Tests on various spoken letters have indicated that the improvement in classification performance is most pronounced on consonant sounds, such as the initial portions of the letters "b", "d", "g", "p", "t" and "v". Vowel sounds ("a", "e", "i", "o" and "u") have shown just marginal improvement. From observations of the utterances at the parametric level, the dynamic range of MFCC value within vowels is smaller than consonants and, generally, they vary more slowly. This may suggest that inter-frame correlations help to discriminate between sounds with a higher transient nature;

- By using the control strategies described in Section 6.5, it is possible to alter the speed of the branch and bound algorithm without severely affecting the classification performance.

References


Chapter 9

Conclusions

9.1 Achievements

Currently popular methods in speech template matching are dynamic programming-based time-alignment (DTW) approaches, stochastic solutions (such as hidden Markov models (HMMs)) and connectionist networks (such as multi-layer Perceptrons and self-organizing networks). Connectionist networks can be shown to be good static pattern classifiers (see Appendix A) and, because fewer assumptions are made regarding the nature of the data, we assume that these networks make use of a variety of discriminatory information present in the speech data. However, for removing local and global timing differences between the input speech and a reference template, DTW and HMM recognizers are currently the most directly applicable approaches.

Neither DTW nor HMMs utilize long-range frame-to-frame statistical dependencies explicitly, although attempts at incorporating contextual information are made by training recognition units in context (supervised training [3]).

In this thesis, a matching criterion and a companion search algorithm, have been developed which take the DTW idea and extend it to use long-range statistical dependencies between frames in both the time-alignment and template matching operations. The new algorithm has been shown to significantly increase the discriminatory power of the DTW method. The main achievements reported in this thesis can be summarized as follows.

- A recursive matching criterion has been developed which incorporates frame-to-frame statistical dependencies and simultaneously performs speaker compensation by removing the mean frequency level of the utterance as recognition progresses. The matching criterion is developed from classical Bayesian decision theory and assumes normally distributed speech models. The resulting Mahalanobis distance-based criterion function effectively exploits the correlations be-
between frames along the entire length of the reference template. Furthermore, we show that the criterion function monotonically increases with the template frame index.

- Speaker variation often manifests itself as shifts in the mean level of the frames parameters. To remove this undesirable effect, the new matching scheme incorporates a mean level compensation scheme which estimates the optimal correction factor to be applied at each point in the time-alignment process.

- A recursive form of the new matching criterion has been developed which allows the calculation of the function value as and when new speech frames become available. The recursive calculation is computationally quite inexpensive requiring $L(k+4) + 1$ multiplications, $3L + 1$ divisions, $L(k+1)$ subtractions and $L(k+4) + 2$ additions where $L$ is the number of frame parameters and $k$ is the template index.

- Template and covariance matrix storage is, fortunately, kept to a minimum through an elegant feature of the matching criterion. As only one column of the inverse covariance matrix is required per stage, storage of the covariance information, for each speech parameter, is limited to just $\frac{N(N-1)}{2}$ locations per class, where $N$ is the template length.

- A search strategy, based upon the branch and bound algorithm, has been developed which successfully allows the minimization of the criterion function. As the criterion function value is monotonically increasing with the depth of the search tree, the highly efficient branch and bound algorithm guarantees to find the optimal warping transformation between the input and the reference model. The traditional dynamic programming algorithm is not suitable for the optimization task as the criterion function is not separable.

- A number of strategies have been suggested which enable flexible control of the search to be achieved including the control of the speed/recognition performance characteristics. The control is divided into the common global and local constraint groups. The global constraint governs the generation of a pre-processed pattern from the raw input speech which caters for the fastest and the slowest speakers within a specified dynamic range.
Local constraints provide

1. a restriction on the maximum and minimum numbers of possible frame-pair matches to be performed at each stage,

2. thresholds which prevent unnecessary calculations in the presence of very good or very bad matches and

3. a "desirable path-weighting" facility.

• An iterative method for creating template models was introduced which includes a novel technique for reconstructing rank-deficient covariance matrices. The strategy produces the models by iteratively length-normalizing (warping) each of the training utterances to a "current-best" set of reference models and then averaging these normalized waveforms for each class. A problem with the creation of rank-deficient covariance matrices is overcome by an effective technique which generates a replacement model matrix from the eigenvectors of the original covariance matrix.

• Correlations between frames in the speech waveform were shown to exist through observations of the class covariance matrices. Both vowel ("aeiou") and consonant ("bdgptv") sounds exhibited strong, and broadly similar, correlations.

• The new matching criterion was shown to achieve significantly better classification performance when compared with a conventional dynamic programming approach tested on spoken letter ("bdg" and "ptv") speaker independent recognition tasks. Performance for vowels ("aeiou") was only marginally improved which is believed to be mainly due to the steady state nature of these sounds.

• Significant improvements in execution speed were shown to be possible by judicious choice of the local constraint threshold value without severe loss of recognition performance.

• The central factor in the improvement of speech models appears to be the control of the inverse covariance reconstruction technique effected at the moment by setting an eigenvalue threshold.

• Results on the effect of training database size suggests that further improvement is almost certainly possible by obtaining more training data.
9.2 Further Work

The studies reported in this thesis merely scratch the surface of the number of advantages of utilizing inter-frame correlations in acoustic-level automatic speech recognition. The reported improvements in classification performance have not been achieved easily. For example, real hope of being able to use inter-frame dependencies to improve classification performance came only after the problems of rank-deficiency in the class covariance matrices were solved. We have found also that the recognition of certain types of speech improves more than for others. In short, there are a number of avenues of investigation to be undertaken which explore the benefits and use of the new matching scheme and which develop better speech model inferencing strategies.

Quantitative Analysis of Consonant/Vowel Utterance Differences: We have suggested that the difference between the improvement in recognition performance for consonants and vowels might be due to the more steady-state, lower dynamic range nature of vowel waveforms (Figures 8.4 and 8.5). To pin down the reasons for these differences, it may be useful to test more utterances in order to devise a quantitative measure for the activity of each utterance. A plot of this activity against the corresponding recognition score may indicate a correlation.

Comparison with a Hidden Markov Model Approach.

Alternative Model Generation Methods: At present, speech models for use in the new technique are generated by an iterative averaging technique. This approach, although capable of generating adequate models, is likely to be affected by poorly endpointed or badly spoken utterances which may be present in the training database. The effect of such outliers could be removed by developing alternative model inferencing methods based on, for example, cluster analysis techniques [1], and enhanced further by a suitable fine-tuning technique to delicately modify the decision boundary (a technique similar to the LVQ2 algorithm by Kohonen [2]).

Basic Recognition Unit: Currently, the basic recognition unit used is designed by hand by observing characteristic features in the MFCC waveforms. An investigation of other recognition units should be undertaken or, if a non-linguistic recognition unit is to be employed, a more robust segmentation procedure should be developed which could be based on waveform energy or derivative terms.
Extension to Continuous Speech Recognition (CSR): At the moment, recognition of single speech units is undertaken in order to validate the new method. The long-term aim should be to develop a full continuous speech recognizer based on the new matching criterion. To accomplish this it is hoped that segmentation of the input speech into segments can be performed simultaneously with recognition, as is often the case in conventional DTW methods applied to CSR, with end-points being allowed to move freely along the input utterance.

Investigation of the Effect of the Seed Template on Model Performance: The shape of the initial seed template may be important. The generation of good seed templates in HMM-based recognizers, for example [3], is considered to be worthwhile. In contrast, many neural-net classifiers are trained with templates comprising random numbers. The benefits of generating better seed templates may just be confined to a reduction in training time.

Multimodal Classes: Allied to the investigation of alternative inferencing strategies comes the need to test the recognition performance of multimodal speech classes — i.e. the use of more than one model per class. Multimodal classes allow the modeling of features such as non-Gaussian utterance groups and accents. The $k$-means training algorithm may be suitable for developing a multimodal inferencing strategy.

Automatic Eigenvalue Thresholding on Single Classes in the ICR Technique: At present, inverse covariance matrix reconstruction (ICR) is controlled by a threshold, $t_v$, against which the eigenvalues of the original covariance sub-matrices are compared. The level of this threshold, although not critical, is important if the maximum performance is to be obtained from the new matching criterion. Furthermore, the best range of values for $t_v$ seems to vary for each utterance group tested ("bdg", "ptv" and "aeiou"). This would suggest that there is a unique range of values for $t_v$ each speech class.

Instead of performing a series of experiments on each speech group for a range of values of $t_v$, it would be desirable to develop an automatic technique, possibly iterative, which could be applied to each class of utterance in isolation (word, phoneme, etc.) in order to find an optimal, or near optimal, value for $t_v$.

Increased Training Set Size: In Section 8.6, results of varying the number of training vectors on recognition performance were presented (Figure 8.12). These results showed a steady improvement in performance with increased training set size and did not
appear to be flattening out. If the size of training database could be increased beyond
the 208 training utterances per class currently used, then recognition scores could be
obtained which approach those reported by the large speech recognizers such as SPHINX,
TANGORA etc., which employ very complex and comprehensive training procedures
(and achieve less than 5% error in general).

Single Model Matrix per Sound Group: We recall from the covariance images on
page 130 that similar utterances (e.g. vowels, consonants) exhibit similar inter-frame
correlation patterns. So, if model storage becomes a problem for some applications, one
solution might be to use a reduced number of models matrices, \( X \). Instead of assigning
each recognition unit its own model matrix, an averaged matrix could be generated
which covers a range of similar sounds.

Robustness in Noise: The BT database used for the experiments in this thesis con-
tained pre-processed data. A rather desirable investigation would involve obtaining
some raw time-domain data, adding various levels of noise and attempting to recognize
the noisy data with the new matching scheme. Intuitively, one might expect a matching
criterion which is influenced by the global variance along the length of the utterance
might be more robust in noise than one which makes decisions on a frame-by-frame
basis.

References

Englewood Cliffs, NJ.)

networks: benchmarking studies”, Proc. IEEE Int. Conf. on Neural Networks ICNN-88, pp. 61–
68.

zine, July, pp. 26–41.
This glossary contains some common terms encountered in phonetics, phonology and acoustic signal processing. If a term is explained in more detail elsewhere in the thesis, or in the literature, a corresponding reference will be given. A word appearing in *italics* will be found as a separate entry in the glossary.

**ACCENT** difference in *phonetic* quality (or pronunciation) between speakers; often regional. There is usually little problem understanding the meaning (c.f. *dialect*).

**AFFRICATES** see *plosives*.

**ALLOPHONE** (see Section 2.4.2) phones which are realisations of the same *phoneme* usually enclosed in square brackets [ ].

**CO-ARTICULATION** (see Section 2.3.1) the effect of neighbouring sounds on the position of the tongue and other articulators.

**CONSONANT** (see Section 2.4.2) set of *phonemes* produced (usually) by a substantial restriction of air flow through the mouth (c.f. *vowel*). Also see *plosives*.

**DIALECT** (see Section 2.3.2) use of alternative words and changes in grammar and pronunciation often used only by members of a certain region. Meaning sometimes difficult to understand (c.f. *accent*).

**DIPTHONGS** (see Section 2.4.2) *vowel phoneme* formed by making a transition from one *vowel* quality to another.

**EUCLIDEAN DISTANCE** a mathematical measure defining the distance between two points in (multi-dimensional) space (see Equation (4.5)).

**FRICATIVES** (subdivided into voiced and unvoiced) - sustainable *consonants* excited primarily by air turbulence.

**FORMANT** peak in the envelope of the power density spectrum (PDS) of a voiced speech sound (see Section 3.3 and the PDS on page 34). Formants, of which there are usually between three and five recognisable, are often used to characterise voiced utterances.

**FRAME** portion of speech (say 10-20ms long) over which the signal is said to be stationary. This time-domain speech segment is transformed into the frequency domain and represented parametrically (see Chapter 3).

**HOMOPHONES** words which have different spellings and meanings but sound alike (e.g. "rays" and "raise").

**I.P.A.** International Phonetic Alphabet - a consistent set of phonetic symbols used in languages.

**MFCC** Mel-frequency cepstral coefficient. A parametric representation of speech based on a frequency- and amplitude-warped DFT magnitude spectrum (see Section 3.4.1).

**NASALS** *phonemes* generated by allowing air to circulate in the nasal cavity.
PRONUNCIATION (see Section 2.4.2) particular manifestation of a phoneme according to the way it was produced (usually due to the position of the phoneme within the word).

PHONEME (see Section 2.4.2) smallest unit in speech where substitution of one for another might make a distinction in the meaning. Usually written /t/.

PHONETICS the generation and classification of speech sounds.

PHONOLOGY study of the function of speech sounds in languages.

PLOSIVES (sometimes called stop consonants and subdivided into voiced, unvoiced and affricative) - phonemes produced by a complete blockage of air flow for a few tens of seconds.

PROSODICS (see Section 2.3.3) combination of pitch, duration (timing) and intensity providing additional information about what is said, e.g. mood, word emphasis.

SEMANTIC AMBIGUITY (see Section 2.3.10) the situation where a syntactically correct sentence can be interpreted in more than one way, for example, “the council decided to subsidise the port”.

SPECTROGRAM a graph of spectral energies with time along the x-axis and frequency along the y-axis. Energies are depicted either by the density of the plot (see Figure 2.1) or by peaks on a three-dimensional perspective view.

SYNTACTIC AMBIGUITY (see Section 2.3.10) usually a confusion as to which object relates to which in a sentence. For example, “the man sat on the chair with the wooden leg”.

STOP CONSONANTS see plosives.

TEMPLATE reference speech utterance model against which unknown input speech is compared.

VOWEL (subdivided into front, middle and back depending upon the position of the “hump” in the tongue) - set of phonemes produced by a relatively unconstricted flow of air through the mouth (c.f. consonant).
Appendix A

A Study of Connectionist & Established Classifiers for Speech Recognition

A.1 Summary

Five pattern classifiers have been studied and applied to the problem of recognising the spoken words "YES" and "NO". Data representation was a 15-dimensional feature vector of linear predictive coefficients. We compared two connectionist, or neural net, methods; the multi-layer perceptron (MLP) and Kohonen's self-organising feature maps, and three established classifiers; the k-nearest neighbour rule, the MULTIEDIT/CONDENSING algorithm and the Gaussian classifier. Extensive experiments were performed to analyse the effect of various classifier learning schemes and parameters, alternative distance measures and feature selection. Results conclude that the MLP is the marginally superior classifier in terms of percentage correct classification of the test data.
A.2 Introduction

Connectionist, or neural network, approaches have been applied widely and successfully to the problem of automatic speech recognition over the last couple of years. This resurgence of interest has established connectionist methods as a very important class of fixed-dimension pattern classifiers.

Not surprisingly, a number of studies comparing the relative performance of connectionist and established classifiers have been performed [5, 2]. Such studies have shown that connectionist methods can match conventional pattern recognition techniques in terms of error rate. Our aims were to extend these studies by including more classifiers in the comparison, and to experiment with aspects of classifier design such as feature selection, distance measures and learning strategy.

In this study, we compare two connectionist and three established classification methods on a two-class speech recognition problem. The connectionist classifiers are Kohonen's self-organising feature maps (SOFMs) [7] and the multi-layer perceptron trained with the back-propagation algorithm [13]. Of the many established and well-documented classical methods we chose two nearest-neighbour based non-parametric classifiers; the $k$-nearest neighbour ($k$-NN) rule and the multiedit algorithm with condensing [4], and one parametric approach; the Gaussian classifier. Each classifier was applied to the two-class speech recognition problem of correctly classifying the words "YES" and "NO". Distinct sets of speech patterns were used for training and testing each classifier.

Firstly in this paper, we present the theory for each classifier in Sections A.3 to A.7. Secondly, we describe the distance measures used and our method of feature selection. Section A.10 reports the results of the study for each classifier in turn. Finally, we discuss the relative merits of each method based on the criteria of error rate, speed of learning and classification and algorithm complexity.

A.3 The Multi-layer Perceptron

The multi-layer perceptron (MLP) is probably the most popular neural network classifier in use today. Its widespread application is due to the discovery of the backpropagation algorithm by Rumelhart, et al [13].

The MLP consists of a number of interconnected layers, each layer referred to as either input, output or hidden. For $D$-dimensional training patterns, the input layer will contain $D$ continuous-valued nodes to which each training vector is presented. Commonly, one or two hidden layers are present. Usually, no more than two hidden layers
are used since it is possible to generate any arbitrarily complex discriminant surfaces with this arrangement alone (Lippman [10]). For an \( M \)-class problem, the output layer will contain either \( M \) or \( \log_2(M) \) nodes and it is the activity of these nodes that indicates to which class a given pattern has been assigned. For \( M \) output nodes, just one node will be active. Full interconnection between adjacent layers exists transmitting the node activity from input to output. Associated with each interconnection is a weight. It is the adaptation of the values of these weights by error back-propagation which constitutes learning in an MLP. A typical structure for the MLP is shown in Figure A.1.

MLP learning is performed on an \( L \) layer, feed-forward network with \( N^\ell \) nodes in each layer, \( \ell = 1, \ldots, L \) (input layer, \( \ell = 1 \), output, layer \( \ell = L \)) using a set of \( N_k \) training vectors in the following manner. Assuming weights are initialised to small, random values, present the \( k \)-th training vector, \( k = 1, \ldots, N_A \), to the input layer, \( \ell = 1 \), of the MLP. Compute the value \( x_j^{(\ell+1)} \) of the \( j \)-th node, \( j = 1, \ldots, N^{\ell+1} \), of the \( (\ell + 1) \)-st layer by finding the weighted sum over all nodes, \( x_i^\ell \), \( i = 1, \ldots, N^\ell \) in the \( \ell \)-th layer as in equation (A.1).

\[
x_j^{(\ell+1)} = f \left( \sum_{i=1}^{N^\ell} w_{ij}^\ell x_i^\ell - \theta_j^\ell \right)
\]  

(A.1)

where \( w_{ij}^\ell \) is the weight from node \( i \) to node \( j \), \( \theta_j^\ell \) is a threshold associated with node
$j$ and the function, $f(\cdot)$, is the sigmoid non-linearity function

$$f(\alpha) = \frac{1}{1 + e^{-\alpha}}. \quad (A.2)$$

This calculation is repeated for all nodes in all subsequent layers until the value of the nodes in the output layer $L$, $x^L_j$, $j = 1, \ldots, N^L$ are known. The actual output node values are compared with the $k$-th pattern desired outputs, $s_{jk}$, $j = 1, \ldots, N^L$, which will be set to values 0.9–1.0 for nodes corresponding to correct classes, and values 0.0–0.1 or (−1.0)–(−0.9) for all other nodes. This comparison yields an error term which, for the output layer $L$, is defined as

$$\delta^L_j = x^L_j(1 - x^L_j)(s_{jk} - x^L_j) \quad (A.3)$$

and for other layers, $\ell = 1, \ldots, (L - 1)$ is

$$\delta^\ell_j = x^\ell_j(1 - x^\ell_j) \sum_{n=1}^{N^{\ell+1}} \delta_n^\ell w_{jn}^\ell. \quad (A.4)$$

Error terms are used to adapt weights as

$$w_{ij}^{(\ell-1)}(k + 1) = w_{ij}^{(\ell-1)}(k) + \eta \delta^\ell_j x_{i}^{\ell-1} + \Delta [w_{ij}^{(\ell-1)}(k) - w_{ij}^{(\ell-1)}(k - 1)]. \quad (A.5)$$

The weight for the $(k+1)$-st pattern will, therefore, be defined in terms of some fraction $\eta$ of the current weight, sometimes called the learning rate. Often, as in equation (A.5), weight changes are smoothed by incorporating another term which is a function of the $(k - 1)$-st weight scaled by a momentum factor, $\Delta$. Threshold values, $\theta^\ell_j$, are updated in a similar way to weights (see the note below).

The learning algorithm is a gradient method aiming to minimise the squared error $E$ between the actual MLP outputs and the desired values over all patterns in the training set where

$$E = \sum_{k=1}^{N_k} \sum_{j=1}^{N^L} (s_{jk} - x^L_j)^2. \quad (A.6)$$

When classifying unknown patterns with the MLP, one computes the activity of each output node (using equation (A.1) repeatedly) on presentation of a test pattern to the input. A high value (0.5–1.0) on only one output node (in the $M$ node case) indicates that the MLP assigns the current pattern to the class associated with the active node.

The MLP in this study had fifteen input nodes and two output nodes. The number of hidden layers was varied between one and two and we experimented with the number of nodes in each layer also. Weight updating was performed after each
pattern was presented but we could easily have chosen to update after all patterns were presented.

A Note on Programming and Updating Threshold Values, $\theta_j^t$ To ease the task of incorporating the threshold values, $\theta_j^t$, into the MLP program, it is common to represent $\theta_j^t$ as an additional weight coupled with an additional node, $N^\ell + 1$, with a constant output value of $-1$. Thus, the input to a node in the layer above will be $-1 \cdot \theta_j^t$, or $-\theta_j^t$ which is correct. Updating $\theta_j^t$ is then a simple matter since it is dealt with in exactly the same manner as the other weights in the associated layer.

The Back-Propagation Algorithm

**Step 1.** Read User Defined Variables

Read quantities $L$, the number of layers in the Perceptron; $N^\ell$, the number of nodes in layer $\ell$, $\ell = 2, \ldots, (L - 1)$; $\eta$, the gain term for adjusting weights; $m$, the momentum term and $t$, the acceptance threshold for comparing with the cost function defined in step 22.

**Step 2.** Read Training Set and Add Extra Dimension

Read $N$ dimensional training vectors and append a dimension $x_i^1$, $i = N + 1$ set to $-1$ to pre-multiply the node offsets. Weights from this extra node to the nodes in the layers above are called offsets or thresholds and represent the quantity $\theta_j^t$ in the sigmoid function given in Step 10.

**Step 3.** Set Layer Index to Input Layer, $\ell = 1$

**Step 4.** Initialise Weights and Threshold

Initialise weights, $w_{ij}^\ell$, ($i \leq N^\ell$), and threshold, $\theta_j^t$, ($i = N^\ell + 1$) to small random values where $i = 1, \ldots, N^\ell + 1$ and $j = 1, \ldots, N^\ell + 1$.

**Step 5.** Until $\ell = L - 1$, $\ell = \ell + 1$ and Go To Step 4

**Step 6.** Set Pattern Counter, $k = 1$

Prepare to read first pattern from the training set of $N_k$ patterns.
Step 7. Present $k^{th}$ Pattern
Present the $k^{th}$ training pattern to input layer $x^1_i$ and set vector, $S_k = (s_{1k}, \ldots, s_{N_k})$.

Step 8. Set Current Layer Index, $\ell = 1$

Step 9. Set Index of Node in Layer Above Current One, $j = 1$

Step 10. Compute Node Value
Compute the value of node $j$ in the layer above the current one using the expression

$$x^\ell_{j+1} = f \left( \sum_{i=1}^{N^{\ell+1}} w^\ell_{ij} x^\ell_i \right)$$

where the function, $f(\cdot)$, is the sigmoid non-linearity function

$$f(\alpha) = \frac{1}{1 + e^{-\alpha}}$$

and $\theta^\ell_j = w^\ell_{ij}, i = N^\ell + 1$.

This is equivalent to evaluating the expression

$$x^\ell_{j+1} = f \left( \sum_{i=1}^{N^\ell} w^\ell_{ij} x^\ell_i - \theta^\ell_j \right).$$

Step 11. Until $j = N^{\ell+1}, j = j + 1$ and Go To Step 10

Step 12. Until $\ell = L - 1, \ell = \ell + 1$ and Go To Step 9

Step 13. Set Layer Index $\ell = L$ (the Output Layer)

Step 14. Set Node Index $j = 1$

Step 15. Compute Error Term
If $\ell = L$ then compute error term between output layer and desired output:

$$\delta^L_j = x^L_j (1 - x^L_j)(s_{jk} - x^L_j).$$
If \( \ell \neq L \) then compute error term between consecutive layers:

\[
\delta_j^{\ell} = x_j^{\ell}(1 - x_j^{\ell}) \sum_{n=1}^{N^{\ell+1}} \delta_n^{\ell+1} w_{jn}^{\ell}
\]

where \( n \) indexes error terms in the layer above the current one.

**Step 16.** Set Node Index, \( i = 1 \)

**Step 17.** Compute New Weight

Modify the weight value between the \( j \)th node in the current layer and the \( i \)th node in the layer below (nearer the input) using the error term scaled by the gain term, \( \eta \).

Thus, weights for the \( k+1 \)st pattern will be

\[
w_{ij}^{\ell-1}(k + 1) = w_{ij}^{\ell-1}(k) + \eta \delta_j^{\ell-1} x_i^{\ell-1} + m(w_{ij}^{\ell-1}(k) - w_{ij}^{\ell-1}(k - 1))
\]

where the last term is a momentum term term to aid convergence and smooth the weight changes.

**Step 18.** Until \( i = (N^{\ell-1} + 1), i = i + 1 \) and Go To Step 17. Includes Offset

**Step 19.** Until \( j = N^{\ell}, j = j + 1 \) and Go To Step 15

**Step 20.** Until \( \ell = 2, \ell = \ell - 1 \) and Go To Step 14

**Step 21.** Until \( k = N_k, k = k + 1 \) and Present Next Pattern, Go To Step 7

**Step 22.** Check Classification Error

Check the classification error by presenting the training set to the multi-layer Perceptron once more. This time, compute the cost function

\[
E = \sum_{k=1}^{N_k} \sum_{j=1}^{N^L} ||S_{jk} - x_j^{\ell}||^2
\]

where \( k \) indexes all patterns in the training set and \( j \) indexes all nodes in the output layer. Compare this value with the acceptance threshold, \( t \).
158 Appendix A

Step 23. If $E < t$ then Stop, else Go To Step 6

A.4 Kohonen's Self-Organising Feature Maps

Kohonen's self-organising feature maps have generated a lot of interest over the last few years, particular for speech processing. They have been used successfully in pattern recognition applications such as phonetic typewriters (Kohonen [8]).

Self-organising feature maps, or just feature maps, are a non-supervised method of pattern classification in which a pre-determined number, $N_m$, of $D$-dimensional reference vectors, $m_i$, $i = 1, \ldots, N_m$ are represented in a low-dimensional, regular structure (usually two-dimensional, hence the term feature map). Reference vectors are moved, or updated, using $N_k$ training vectors according to the learning strategy below. The resulting map tends to be organised such that vectors from the same class occupy adjacent map locations.

Learning commences by setting the reference vectors to random positions near the origin in $D$-dimensional feature space. For the first pattern, $k = 1$, the Euclidean distance to all reference points is computed. The point with the shortest distance, point $c$, becomes the centre of a neighbourhood on the feature map. This neighbourhood of $N_c$ points is large initially and decreases in size proportional to some inverse function of time $t$. Figure A.2 shows a typical feature map with members of the neighbourhood in grey.

Members of the neighbourhood are moved towards the current training point in the following manner:

$$m_i(t + 1) = \begin{cases} m_i(t) + \alpha(j)[x(t) - m_i(t)] & \text{for } i \in N_c \\ m_i(t) & \text{otherwise} \end{cases} \quad (A.7)$$

where $\alpha(t)$ is a gain function which decreases with time. Learning continues for the remaining training patterns for a number of iterations of the data. This vector updating procedure just described is depicted graphically in Figure A.2.

In this study, three learning phases and three gain functions were investigated:

Phase 1 Non-supervised learning using the method described above uses three types of gain function; linear, exponential and centre weighted, the latter being a function of both time and map radius. Experiments using centre weighted adaptation by
Figure A.2: Kohonen's self-organising feature map showing reference vectors in the measurement space and on the feature map. Map learning is performed by moving reference points towards the current training sample with a constantly decreasing gain and neighbourhood size.
Brauer and Knagenhjelm [1] indicated that it might be possible to achieve faster convergence with this function than with the others.

Phase 2 Non-supervised linear adaptation of point c only.

Phase 3 Supervised LVQ2 algorithm. The LVQ2 algorithm has been recently developed (Kohonen [9]) which aims to move reference points in feature space which are close to the decision boundary and which are likely to cause misclassification. Major successes with LVQ2 have been reported by McDermott and Katagiri [12] suggesting performances exceeding that of the MLP.

In addition, extensive experiments were performed to analyse the effect of feature map size, maximum neighbourhood size and feature selection on performance.

Kohonen's SOFM Learning Algorithm

**Step 1.** Set Initial Size of Neighbourhood

Initialize the size of feature-map neighbourhood \( N_c(0) \).

**Step 2.** Initialize Output Point Positions

Initialize the positions of \( M \) output points, \( m_i(0) \), \( i = 1, \ldots, M \), in \( D \)-dimensional pattern space to small random values.

**Step 3.** Present New Input

Present \( j \)-th input to the network, \( x(j) \), \( j = 0, \ldots, N_j - 1 \), where \( N_j \) is the number of patterns used in the learning process.

**Step 4.** Compute Distance to All Output Points

Compute squared Euclidean distances \( d_i \) between current input point \( x(j) \) and each of the output points:

\[
d_i = [x(j) - m_i(j)]^T [x(j) - m_i(j)]
\]

**Step 5.** Select Output Point with Minimum Distance

Select the output point with the shortest distance \( d_i \) and let this point be the centre of the neighbourhood \( N_c \).
Step 6. Update Position of Output Points

Update the position of each output point in the following manner:

\[ m_i(t+1) = \begin{cases} 
  m_i(t) + \alpha(j)[x(t) - m_i(t)] & \text{for } i \in N_c \\
  m_i(t) & \text{otherwise}
\end{cases} \]

The term \( \alpha(j) \) is a gain term \( (0 < \alpha(j) < 1) \) that decreases with time and 1 is a unit vector.

Step 7. Reduce Neighbourhood Radius

The radius of the new neighbourhood will follow some inverse function of the input pattern number, \( j \).

Step 8. Increment \( j \) and goto Step 3

A.5 The Multiedit/Condensing Algorithm

The multiedit/condensing algorithm is a nearest-neighbour type classifier which overcomes the prohibitively large computation problem usually associated with this type of classifier. It is a method which has been demonstrated to be successful in the design of pattern recognition systems such as optical character reading machines (Voisin and Devijver, [15]).

The multiedit algorithm performs repeated editing, or discarding, of \( N_k \) training samples. Edited points tend not to belong to their own Bayes acceptance region and, therefore, would be likely to cause misclassification. Once edited, the resulting clusters are homogeneous and lie either side of the optimal Bayes decision boundary. Theoretical and experimental evidence [4] show that multiediting converges asymptotically to the Bayes decision rule with the number of iterations. The proportion of samples which are edited is, however, usually small and can be shown to be bounded from above by \( 2E_1 \), where \( E_1 \) is the nearest neighbour rule error rate [4].
Step 1. Diffusion
Make a random partition of the training data into $N$ subsets, $S_i$, $i = 1, \ldots, N$ where $N \geq 3$.

Step 2. Classification
Classify the samples in $S_i$ using the nearest neighbour rule with $S_{(i=1) \mod N}$ as a training set.

Step 3. Editing
Discard all samples that were misclassified at Step 2.

Step 4. Confusion
Pool all the remaining data to constitute a new set.

Step 5. Termination
If the last $I$ iterations produced no editing, then exit with the final set, else goto Step 1.

Further, and more dramatic editing, is performed by the condensing algorithm applied to the edited training data set. The success of the condensing algorithm is based on the valid assumption that the class boundary so formed by the edited samples is defined by a relatively small number of samples in the outer regions of the class clusters. Samples deeply embedded within clusters may be discarded with little effect on error rate. Classification of unknown patterns is performed with the nearest neighbour rule using members of the condensed set as reference vectors.

Step 1. Initialisation
Set up bins called STORE and GRABBAG. The first edited set sample is placed in STORE, the remaining samples into GRABBAG. Let the number of samples in GRABBAG at any time be $n_g$. 

The Multiedit Algorithm

The Condensing Algorithm
Step 2. Classification

Use the nearest neighbour rule with the current contents of STORE to classify the i-th sample from GRABBAG. If classified correctly, the sample is returned to GRASSAC, otherwise it is placed in STORE. Repeat this operation for i = 1, ..., n_g.

Step 3. Termination

If one complete pass is made through Step 2 with no transfer from GRABBAG to STORE or the GRABBAG is exhausted then terminate with the edited set in STORE, else go to Step 2.

It is known (Devijver [3]) that the basic condensing algorithm is not optimal in it's selection of reference vectors from the edited training set. Recently, we have developed a scheme for ordering the edited data which has been found to improve performance in some cases. The results presented in Section A.10.5) confirm that the ordering technique did indeed improve the classification performance for this problem.

A.6 Gaussian Classifier

In the study, one parametric classifier was implemented; the Gaussian classifier. The estimated parameters of mean, \( \mu_i \) and covariance, \( \Sigma_i \) for the i-th class, \( \omega_i \) are found from the samples in the training set from equations (A.8) and (A.9).

\[
\mu_i = \frac{1}{N_i} \sum_{j=1}^{N_i} x_j \quad x_j \in \omega_i \tag{A.8}
\]

\[
\Sigma_i = \frac{1}{N_i} \sum_{j=1}^{N_i} (x_j - \mu_i)(x_j - \mu_i)^T \tag{A.9}
\]

where \( N_i \) denotes the number of training samples for the i-th class.

In our study, we chose to use just one Gaussian model per class, but it is possible to use more than one. The class conditional densities are of the form:

\[
p(x \mid \omega_i) = \frac{1}{\sqrt{(2\pi)^D | \Sigma_i |}} e^{-\frac{1}{2} (x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i)} \tag{A.10}
\]
where $| \Sigma_i |$ is the determinant of $\Sigma_i$, $\Sigma_i^{-1}$ is the inverse of $\Sigma_i$ and $(x - \mu_i)^T$ denotes the mathematical transpose of $(x - \mu_i)$.

### A.7 $k$-Nearest Neighbours Classifier

The $k$-nearest neighbour ($k$-NN) classifier is a well established non-parametric classifier which approximates the optimal Bayes decision boundary [4]. Consider a set of $n$ pairs of training samples $(x_i, \theta_i)$, $i = 1, \ldots, n$, where $\theta_i$ is the true class label for sample $x_i$ from any one of $M$ classes $\omega_c$, $c = 1, \ldots, M$. For an unknown pattern, $(x', \theta')$, the distance metric $\delta$ is computed between all training samples and the unknown pattern. Often, as in this study, the Euclidean metric is chosen. The nearest neighbour to $x'$, $x_{NN}$, is

$$ x_{NN} = x_j \text{ if } \delta(x_j, x') = \min_{i=1, \ldots, n} \delta(x_i, x'). \tag{A.11} $$

Within the set of $k$ nearest neighbours to $x'$, there will in general be a mixture of class labels. The number of samples from class $c$, $c = 1, \ldots, M$ in the $k$ nearest neighbours is designated $\ell_c$. The class label for the unknown sample is then estimated thus:

$$ \theta' = \omega_j \text{ if } \ell_j = \max_{i=1, \ldots, M} \ell_i. \tag{A.12} $$

### A.8 Alternative Distance Measures

In this study, it was our intention to experiment with currently successful enhancements to some of the classifiers. One of the most interesting advances in nearest neighbour techniques has been the work of Short and Fukunaga on optimal distance measures (Short and Fukunaga, [14]). They formulate a distance measure which minimises locally the mean squared error between the large-sample and finite-sample probabilities of misclassification of the nearest neighbour rule.

In the two-class case, this distance measure can be readily computed. Firstly, the $m$ nearest neighbours $x_i$, $i = 1, \ldots, m$ to the current test sample $x'$ are found. If there exists a mixture of class labels within this group of $m$ samples then the means of each class, $M_1$ and $M_2$, are computed. The $k$ nearest neighbours to sample $x'$ taken from the $m$ closest samples are found using the projected distance measure $d_2$, equation (A.14). The performances of the multiedit/condensing algorithm and $k$-NN classifier were assessed using the metric $d_2$ and the standard Euclidean metric, $d_1$, in equation (A.13).
\[ d_1 = (x' - x_i)^T(x' - x_i) \quad (A.13) \]

\[ d_2 = |(M_1 - M_2)^T(x' - x_i)| \quad (A.14) \]

For \( m \) nearest neighbours, only \( m \) extra distances need to be computed and hence the percentage increase in computation time is just \( m/N_k \), where \( N_k \) is the number of training samples. This marginal increase in learning time is not a problem as classification speed using the 1-NN rule is not affected. Short and Fukunaga have shown theoretically and experimentally [14] that the new measure is superior in performance to the Euclidean distance metric \( d_1 \) thus motivating its use in this study.

### A.9 Feature Selection

It is sometimes the case, especially in high dimensional feature spaces, that selective components in the feature vectors either contain redundant information, or make a detrimental contribution to classification performance. Experiments were performed in this study to test the effect of feature selection as a natural part of system design. The time required to perform exhaustive selection of features for this problem is prohibitively large. As a first approximation to the effect of feature selection, it was decided to take a more simple, if sub-optimal, approach.

Starting with the full feature vectors, each feature component was removed in turn and the subsequent performance assessed. With the best result, a further component was removed, in turn, from those remaining, assessing performance once more. This process was continued to completion where just one component remained. An additional set of experiments using a similar, except bottom-up, empty-to-full component method, helped to quickly identify a set of features which represented the problem well.

### A.10 Experimental Results

#### A.10.1 Training and Test Data

The data used to train and test each classifier was a set of 1418 pattern vectors of the utterances "YES" and "NO" spoken over the public switched telephone network. Each fifteen dimensional vector contained five segments of three features derived by low order linear prediction analysis. From this set, 798 samples were used for training and 620 different samples for testing the classifiers. All data was supplied by British Telecom as part of the CONNEX project.
A multi-layer perceptron with a maximum of two hidden layers was trained on the "YES/NO" training data described above. The number of nodes in each hidden layer was varied in the ranges 2-15 in hidden layer 1, and 2-10 in hidden layer 2. For this two-class speech recognition problem, two output nodes were used allowing independent activity. Training proceeded with the correct class output node set to a value of 1.0 and the incorrect class node to 0.0. For most experiments, 4500 iterations of the complete training pattern set were made during learning.

Learning in multi-layer perceptrons with the basic back-propagation algorithm is highly computer intensive. The smallest net, with just two nodes in one hidden layer, took six hours of central processor time on a MicroVAX II and the largest net, with 25 nodes in two hidden layers, took 36 hours.

Various values of learning rate, $\eta$, and momentum, $\Delta$, were used in a number of experiments. Large values of $\eta$ ($>0.5$) caused instability in the learning phase, the best choice for $\eta$ being in the range 0.01-0.1, and for $\Delta$ in the range 0.1-0.9. Figure A.3 shows an example of a smooth learning characteristic for a small net of one hidden layer with three nodes and values for $\eta$ and $\Delta$ of 0.01 and 0.9 respectively. Figure A.4 shows the corresponding squared output error, $E$.

During classification, the activity of each output node was examined and com-
Figure A.4: Multi-layer perceptron learning characteristics showing how the squared output error, $E$, varies with iteration for test and training data.

<table>
<thead>
<tr>
<th>Output node threshold values</th>
<th>% Correct classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>threshold $T_H$</td>
<td>threshold $T_L$</td>
</tr>
<tr>
<td>0.7</td>
<td>0.3</td>
</tr>
<tr>
<td>0.6</td>
<td>0.4</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table A.1: Effect of output node threshold values on performance for an MLP

pared against two thresholds: $T_H$, a “high” threshold, and $T_L$, a “low” threshold. Let the value of the node corresponding to class $\omega_i$ be $x_{Li}$, $i = 1, 2$. Correct classification of a particular test sample from class $\omega_i$, was achieved if $x_{Li} > T_H$ and $x_{Lj} < T_L$, $j \neq i$. A number of threshold pair values, $(T_H, T_L)$, were used in classification experiments. Results for an MLP with two hidden layers are shown in Table A.1.

The overall behaviour of the MLP is very much dependent upon the particular structure used. To assess the variation of performance for a number of structures, several experiments varying the number of hidden layers and hidden units were performed. Single hidden layer nets achieved superior stability during learning and reached peak performance with a minimal number of hidden units. Indeed, the best score, one of 95.3%, was achieved with an MLP comprising one hidden layer of five nodes. Complete results of the MLP experiments are presented in Table A.2.
### Table A.2: Effect of changing the number of hidden layer nodes on MLP performance

<table>
<thead>
<tr>
<th>Number of hidden layer nodes</th>
<th>% correct classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer 1</td>
<td>Layer 2</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
</tr>
</tbody>
</table>

A.10.3 Kohonen's Self-Organising Feature Maps

A number of self-organising feature maps were trained on the “YES/NO” data and the set of reference vectors generated were used to classify the test data with the nearest neighbour rule. Three phases of learning were performed in turn (see Section A.4) and the peak results for phase 1 are shown in Table A.3 for both linear and exponential gain functions. In general, learning behaviour was not stable for phase 1 learning alone. The addition of phase 2 adaptation, however, greatly improved stability. A selection of the best learning curves for phase 1 and 2 adaptation is shown in Figure A.5 for three map sizes. A 7x7 map can be seen to achieve the best performance using linear phase 1 and 2 gain functions.

Experiments were performed to analyse the effect of map size and initial neighbourhood size on performance. Larger maps produced higher classification scores (see Table A.3) although this was not found to be a general rule. The results also indicate that, in some cases, a smaller initial neighbourhood size can improve performance.

Phase 1 adaption of the reference vectors was performed with three gain functions; linear, exponential and centre weighted. A detailed comparison of linear and exponential adaptation is provided in Table A.3 but it can be seen that neither gain function achieves consistently best results for all map sizes. However, linear gain achieved the peak score for Kohonen feature maps of 93.1% for a 7x7 map. Results for the centre weighted gain function are shown in Table A.4

Feature maps have a higher learning rate than MLPs. From (Table A.4), it can be seen that feature maps can be trained in around 20 iterations of the training data.

Phase 3 learning, using the LVQ2 algorithm, proved to be of little value, not being
Table A.3: Effect of maximum neighbourhood size on feature map performance for linear and exponential gain functions applied to the test data.

<table>
<thead>
<tr>
<th>Map size</th>
<th>max. neigh'd</th>
<th>Best %correct</th>
<th>Linear</th>
<th>Expo.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4x4</td>
<td>4</td>
<td>89.8</td>
<td>90.0</td>
<td></td>
</tr>
<tr>
<td>4x4</td>
<td>3</td>
<td>90.6</td>
<td>90.3</td>
<td></td>
</tr>
<tr>
<td>5x5</td>
<td>5</td>
<td>91.6</td>
<td>91.5</td>
<td></td>
</tr>
<tr>
<td>5x5</td>
<td>4</td>
<td>91.5</td>
<td>92.4</td>
<td></td>
</tr>
<tr>
<td>5x5</td>
<td>3</td>
<td>91.5</td>
<td>91.1</td>
<td></td>
</tr>
<tr>
<td>6x6</td>
<td>6</td>
<td>91.6</td>
<td>90.8</td>
<td></td>
</tr>
<tr>
<td>6x6</td>
<td>5</td>
<td>89.8</td>
<td>90.5</td>
<td></td>
</tr>
<tr>
<td>6x6</td>
<td>4</td>
<td>91.3</td>
<td>90.0</td>
<td></td>
</tr>
<tr>
<td>7x7</td>
<td>7</td>
<td>90.5</td>
<td>90.2</td>
<td></td>
</tr>
<tr>
<td>7x7</td>
<td>6</td>
<td>93.1</td>
<td>91.1</td>
<td></td>
</tr>
<tr>
<td>7x7</td>
<td>5</td>
<td>91.5</td>
<td>91.9</td>
<td></td>
</tr>
<tr>
<td>7x7</td>
<td>4</td>
<td>92.3</td>
<td>91.8</td>
<td></td>
</tr>
</tbody>
</table>

Figure A.5: A comparison of learning characteristics in Kohonen feature maps for phase 1 and 2 adaptation of the LVQ2 algorithm for three sizes of feature map.

Table A.4: Peak recognition performances and learning rates for linear, exponential and centre gain functions in Kohonen's learning algorithm on the test data.

<table>
<thead>
<tr>
<th>Gain function</th>
<th>%Correct</th>
<th>Map size</th>
<th>Number of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear</td>
<td>93.1</td>
<td>7x7</td>
<td>21</td>
</tr>
<tr>
<td>expo.</td>
<td>92.4</td>
<td>5x5</td>
<td>17</td>
</tr>
<tr>
<td>centre</td>
<td>92.9</td>
<td>7x7</td>
<td>25</td>
</tr>
</tbody>
</table>
able to increase the performance of the previously trained net for any map size. Feature selection was performed on the training data during learning using the technique outlined in Section A.9. A small improvement in performance, to 93.9%, was achieved by removing five coefficients (2, 4, 9, 12 and 14) from each feature vector.

### A.10.4 k-NN Classifier

The k-NN rule algorithm was applied to the speech data to generate a benchmark figure of performance for the group of NN classifiers under study. Both Euclidean (d1) and Short/Fukunaga (d2) distance measures (see Section A.8) were used for this classifier. Figure A.6 shows the performance of the k-NN rule for $1 \leq k \leq 33$ using both metrics. A complete table of results for both metrics, and for feature selection, is given in Table A.5 suggesting that metric $d2$ can achieve a significant improvement over measure $d1$ and that both metrics can be enhanced by careful choice of features.

### A.10.5 The Multiedit/Condensing Algorithm

The multiedit algorithm was applied to the speech training data editing around 200 of the 798 original samples. Condensing edited a more dramatic number of samples leaving just 25–35 reference vectors. Employing an ordering scheme to the edited data before condensing (see Section A.5) produced a significant improvement in classification
<table>
<thead>
<tr>
<th>Distance measure</th>
<th>Feature selection</th>
<th>Best % correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>No</td>
<td>93.2</td>
</tr>
<tr>
<td>$d_1$</td>
<td>Yes</td>
<td>94.5</td>
</tr>
<tr>
<td>$d_2$</td>
<td>No</td>
<td>94.7</td>
</tr>
<tr>
<td>$d_2$</td>
<td>Yes</td>
<td>94.8</td>
</tr>
</tbody>
</table>

Table A.5: Classification performance using the k-NN rule on the test data for the Euclidean and Short/Fukunaga distance measures and feature selection

<table>
<thead>
<tr>
<th>Distance measure</th>
<th>Ordered set</th>
<th>Feature selected</th>
<th>%Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>No</td>
<td>No</td>
<td>86.9</td>
</tr>
<tr>
<td>$d_1$</td>
<td>No</td>
<td>Yes</td>
<td>93.1</td>
</tr>
<tr>
<td>$d_1$</td>
<td>Yes</td>
<td>No</td>
<td>91.3</td>
</tr>
<tr>
<td>$d_2$</td>
<td>No</td>
<td>No</td>
<td>90.2</td>
</tr>
<tr>
<td>$d_2$</td>
<td>No</td>
<td>Yes</td>
<td>92.1</td>
</tr>
</tbody>
</table>

Table A.6: Results obtained using the multiedit/condensing algorithm on the test data for two distance measures showing the effects of feature selection and the data ordering method

scores. However, to date, only the $d_1$ measure has been used and using full-feature samples. The alternative distance measure $d_2$ also improved performance but not as much as expected from the results obtained with the k-NN rule. A summary of the best results for all tests performed on the multiedit/condensing algorithm are given in Table A.6. It can be seen that the best score achieved using this classifier was obtained with the Euclidean measure on a feature selected training set. Note should be made, however, that no unique feature set was used for the feature selection experiment with metric $d_2$ unlike experiments using $d_1$. Future tests may prove metric $d_2$ to be superior with reduced dimension feature vectors as well.

A.10.6 Gaussian Classifier

The final recognition method implemented in this study was the parametric Gaussian classifier. The mean and covariance matrices for both classes ("YES" and "NO") were estimated using the 798 training samples. In this study, only one set of parameters were estimated per class, but future work may extend this to use mixtures of intra-class cluster parameters. A typical clustering technique might be the k-means algorithm. Feature selection proved useful showing a significant improvement in performance. A complete set of recognition scores are shown in Table A.7 for training and test data with and without feature selection.
A.11 Conclusions

Five pattern classifiers have been studied and applied to the problem of recognising the spoken words "YES" and "NO". Two methods were from the class of classifiers commonly known as connectionist or neural net methods; the multi-layer perceptron and Kohonen's self-organising feature maps. The others were established methods of pattern classification, namely; the k-NN rule, the multiedit/condensing algorithm and the Gaussian classifier. In each case, training was performed on 798 fifteen-dimensional feature vectors and tested using 620 different samples. A summary of the best performance for each method, including some enhancements, is shown in Table A.8.

In terms of classification error, the multi-layer perceptron is superior, achieving 95.3% on the test set. The feature map classifier and the multiedit/condensing algorithms achieved similar performances, a result which was reported in earlier work (Lucas and Kittler [11]).

Learning speed varied considerably. MLP learning was the most computationally intensive technique (see Section A.10.2) with the multiedit/condensing algorithm next, using around 45 minutes of CPU time. Kohonen maps have the advantage of being trained rapidly (in tens of minutes) in comparison to the MLP.

Classification speeds for a trained classifier are an important factor for real-time (speech frame rate) recognition systems. Feature maps and the multiedit/condensing algorithm use the NN rule for classification. Both methods generate similar numbers of
reference vectors (around 25-35) and hence have similar classification speeds. Therefore, these classifiers could be used in real-life recognition systems. Peak performance for the MLP classifier was achieved with just three hidden units in one hidden layer (see Table A.2). Such a network could easily be applied to real-time classification problems. Of course, the k-NN classifier would not, in general, be considered as a feasible classification method in real-time as it is dependent upon the number of training samples. It is included in this study to set a standard for NN performance. Classification using the Gaussian model is not dependent upon the size of a training set, as with the k-NN rule. However, the computation of equation (A.10) may be prohibitive for large covariance matrices unless a function look-up table was used.

The multiedit/condensing, feature map and k-NN algorithms perform simple, repetitive calculations requiring the computation of the Euclidean distance measure. In this sense, they have a low complexity and would be easy to implement in hardware. Classification using the MLP requires the calculation (or look-up) of the sigmoid function (equation (A.2)). However, the MLP is still a simple classifier which could be used in real-time.

Acknowledgement

The author is grateful to British Telecom for supplying the data used in this study.

References


Appendix B

Notation

The following list constitutes a comprehensive definition of the mathematical notation applied in this thesis.

Note: At some points in the text, the analysis is carried out on a single parameter of the \( L \) parameters which make up a complete speech frame. In these cases, the term frame no longer refers to an \( L \)-dimensional vector of speech parameters acquired over a short period in time but, instead, identifies a single parameter from the frame — effectively a one-dimensional sample of the speech signal.

\[ g(i, j) \] cumulative template matching error between \( i \)-th input speech frame and \( j \)-th template frame.

\[ y_r \] general \( r \)-th class input speech vector (class subscript sometimes omitted).

\[ y_r(t) \] \( t \)-th training utterance \( t = 1, \ldots, \Omega_r \).

\[ y_r(t) \] normalised version of \( y_r(t) \).

\( \Omega_r \) number of training utterances in class \( \omega_r \).

\[ y_j[k] \] vector of the first \( k \) frames of the \( j \)-th parameter (of \( L \) parameters) of the input speech (\( j \) superscript sometimes omitted).

\[ y_{i,k}^j \] \( k \)-th element of the \( j \)-th parameter of the \( i \)-th class vector \( y_i \) (class subscript \( i \) and parameter subscript \( j \) sometimes omitted).

\( \mu_r \) general \( r \)-th class speech template vector (class subscript sometimes omitted).
$\mu^j[k]$ ............................ vector of the first $k$ frames of the $j$-th parameter (of $L$ parameters) of the template ($j$ superscript sometimes omitted).

$\mu^j_k$ ............................ $k$-th element of the $j$-th parameter of the $i$-th class mean (or template) vector, $\mu_i$ (class subscript $i$ and parameter subscript $j$ sometimes omitted).

$\Sigma^j_i$ ............................ $i$-th class covariance matrix for the $j$-th speech parameter (class subscript sometimes omitted).

$\Sigma^j_i[k]$ ............................ $k \times k$ $i$-th class covariance matrix for the $j$-th speech parameter (class subscript sometimes omitted).

$\Sigma^j_i[0,k]$ ............................ $k \times k$ top left-hand corner sub-matrix of $\Sigma^j_i$ at the $i$-th iteration of the speech model inferencing process.

$X^j_i$ ............................... $i$-th class (sometimes omitted) special matrix for the $j$-th speech parameter. The special matrix is a triangular matrix formed from the last columns of the inverses of all the $(k \times k)$, $k = 1, \ldots, N$ upper-left-hand corner submatrices of the $i$-th class covariance matrix.

$\sigma^j_k$ ............................ $k$-th column of the inverse covariance matrix $(\Sigma^j_i)^{-1}$ and the special inverse covariance matrix $X^j_i$ for the $j$-th parameter of the template.

$\sigma^j_{kk}$ ............................ element of $(\Sigma^j_i)^{-1}$ at location $k, k$.

$\tilde{\sigma}^j_k$ ............................ first $(k - 1)$ elements of $\sigma^j_k$.

$\omega_i$ ............................... $i$-th class — group of points belonging to the same class and thus having the same label.

$p(y|\omega_i)$ ............................ conditional class probability density function of class $\omega_i$.

$P(\omega_i)$ ............................ a priori probability of the occurrence of class $\omega_i$.

$J_i(y)$ ............................... $i$-th class criterion function taking vector $y$ as an argument.

$W(Z)$ ............................... time axis warping transformation.

$d(i, j)$ ............................... Euclidean distance measure (see $g(i, j)$).

$M[k]$ ............................... matching criterion value up to the $k$-th frame of the template.

$M[k, p]$ ............................... matching criterion value up to the $k$-th frame of the template when paired with the $p$-th frame of the input signal ($p = q_i(k)$).
Notation

\( p \) ................................... equivalent to \( q_{k+1}^{k+1} \) the input speech frame index.

\( M_r(y) \) ................................... \( r \)-th class Mahalanobis distance measure taking vector \( y \) as an argument.

\( S_k \) ................................... constant level subtracted from the input speech signal up to the \( k \)-th frame.

\( e[k] \) ................................... \( k \)-dimensional unit vector.

\( \gamma^j[k] \) ................................... mean level of the input speech signal up to the \( k \)-th frame.

\( \alpha^j[k] \) ................................... mean level of the template up to the \( k \)-th frame.

\( M_B \) ................................... current best match criterion function value.

\([k, p]\) ................................... node in the branch and bound search tree locating the matching criterion value up to the \( k \)-th template frame and the \( p \)-th input speech frame \( (p = q_{k+1}^{k+1}) \).

\( I_i \) ................................... frame index increment giving the relative location of a frame with respect to the current frame, \( p \). There are \( \ell \) frame increments, \( I_i, i = 1, \ldots, \ell \) and \( I_i \in (1, \ell) \).

\( s(i) \) ................................... number of unpruned branches at the \( i \)-th frame of the template (\( i \)-th level of the branch and bound tree), \( s(i) \in (0, \ell) \).

\( q_{s(k)}^{k+1} \) ................................... index to a frame in the input speech which is currently identified by the \( s(k) \)-th branch at the \( (k + 1) \)-st level in the branch and bound tree.

\( Q_k \) ................................... sequence of successor nodes to the \( (k + 1) \)-st level of the branch and bound tree from the \( k \)-th level.

\( \varepsilon_k/\zeta_k \) ................................... matching criterion correction factor at the \( k \)-th frame of the template defined in terms of \( \eta^j_k, \rho^j_k \) and \( \sigma^j_k \).

\( \eta^j_k \) ................................... simplifying notation defined in terms of \( y^j_k, \mu^j_k \) and \( \sigma^j_k \) for the \( j \)-th parameter of the \( k \)-th frame of the template.

\( \rho^j_k \) ................................... simplifying notation defined in terms of \( \rho^j_k \) for the \( j \)-th parameter of the \( k \)-th frame of the template.

\( Z \) ................................... raw unprocessed input speech utterance.

\( N_Z \) ................................... dimensionality of the raw input speech signal, \( Z \).

\( Z' \) ................................... pre-processed input speech utterance.

\( N_{Z'} \) ................................... dimensionality of the pre-processed input speech signal, \( Z' \).
$Z_p$ .................................. $p$-th element of vector $Z$.

$N$ .................................. dimensionality of the template $\mu$.

$N_{z\text{min}}$ .................................. minimum number of speech frames expected (fastest speaker).

$N_{z\text{max}}$ .................................. maximum number of speech frames expected (slowest speaker).

$R_R$ .................................. frame repeat rate — a ratio of template length to $N_{z\text{min}}$.

$R_s$ .................................. speech record length factor — a ratio of $N_{z\text{max}}$ to template length.

$\ell$ .................................. number of possible frames (successor pairs) from the input speech signal to be paired up with a single frame from the template at each stage $k$ of the branch and bound search tree.

$\ell'(I_{i-1})$ .................................. successor node excursion restriction defining the maximum number of successor nodes which can be expanded to the $i$-th level of the branch and bound search tree. It is set dynamically depending upon the number of frames skipped at the previous $(i-1)$-st stage, $I_{i-1}$.

$P(i, n)$ .................................. probability that word $i$ is uttered at time $n$.

$Q(i|j)$ .................................. transition probability that word $i$ is uttered at time $n+1$ given that word $j$ was uttered at time $n$.

$C(i)$ .................................. correctness measure quantifying the probability that a sentence (string of known words) is uttered.

$T_r(k)$ .................................. "bad-path" rejection threshold which is a function of the tree depth, $k$.

$T_a$ .................................. "good-path" acceptance threshold.