KATHOLIEKE UNIVERSITEIT LEUVEN
FAKULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
AFDELING E.S.A.T.
Kardinaal Mercierlaan 94, B-3001 Leuven (Heverlee)

CONNECTIONIST VECTOR QUANTIZATION IN AUTOMATIC SPEECH RECOGNITION

Promotoren:

Prof. Dr. ir. D. Van Compernolle
Prof. Dr. ir. P. Wambacq

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Weiye MA

Januari 1999

KATHOLIEKE UNIVERSITEIT LEUVEN
FAKULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
AFDELING E.S.A.T.
Kardinaal Mercierlaan 94, B-3001 Leuven (Heverlee)

CONNECTIONIST VECTOR QUANTIZATION IN AUTOMATIC SPEECH RECOGNITION

Jury:
Prof. E. Aernoudt, voorzitter
Prof. D. Van Compernolle, promotor
Prof. P. Wambacq, promotor
Prof. J. Vandewalle
Prof. D. Bollé (Fac. Wetensch.)
Prof. W. Verhelst (VUB)

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Weiye MA

U.D.C. 621.3*127

Januari 1999

D/1998/7515/63

ISBN 90-5682-161-X

For My Parents

Acknowledgments

All through the years of this long project, many people have provided me with help and support. I would like to acknowledge them with great appreciation.

I am heavily indebted to my promoter, Professor Dirk Van Compernolle, for his guidance, support and continuous stimulation throughout my work. He has taught me many things, both about speech recognition and the science and art of doing research. He has also always been ready to help in difficult moments and to point me in the right direction. Completing a dissertation is a trying challenge in itself; working full time while undertaking that challenge would have rendered the task impossible were it not for Professor Compernolle's patience and constant support. Without his help and encouragement I would not have been able to finish this dissertation.

I am extremely grateful to Professor Patrick Wambacq for accepting to become my promoter. His help and support came at a crucial moment in the project, and I want to make a special note of that. I also want to thank him for his insightful feedback on the text of this dissertation, for his recommendations on improving the work, and for helping me with the final arrangements of the defense.

I would like to thank Professor Joos Vandewalle for promoting research and discussions on neural networks in the SISTA group. I learned a lot from my interaction with him and the SISTA group. His feedback on the text has been especially valuable and has critically enhanced the quality of this work.

I wish to express my sincere gratitude to my previous promoter, Professor André Oosterlinck, who first gave me the opportunity to study at ESAT and provided me with both a scholarship and a research subject.

Many thanks go to my colleagues in ESAT, Kris Demuynck, Gert Sablon, Frank Schoeters, Xiaohai Shen, Johan Smolders, Stefaan Van Gerven, Wei Xu, Fei Xie, Chen Yi, Daobin Zhang, and especially Philippe Le Cerf and Marc Van Diest, for sharing their thoughts and ideas with me.

Tom Claes and Jacques Duchateau helped me with the red tape and the paperwork while I was in the US. Kris Hermus provided me with design information for the cover page. I want to thank them sincerely for their crucial help.

My gratitude goes to our secretaries, Annitta DeMessemaeker and Patricia Waege, for their help. During the last phases, Annitta was immensely helpful in administrative matters and always came through for me whenever I needed her. I am greatly indebted to her.

I want to thank my supervisor in Unisys, Dr. Bill Scholz, for the infinite support he has given me and for letting me have a flexible work schedule whenever possible. His infectious and unflappable enthusiasm has many times given me the needed boost of energy to push on with the work. I can safely say that it would have been impossible to complete this work without his support and understanding.

I would also like to thank my friends and colleagues in Unisys who together form a remarkably pleasant bunch of people to have around: Reggie Blue, DongDong Chen, Deborah Dahl, Ray Diedrichs, Cindy Demoss, David Ferro, Jim Irwin, Daythall Kendall, Marcia Linebarger, Li Li, Lew Norton, Bahrati Palle, John Romania, Bill Rose, Ed Thompson, Dennis Wadsworth, Joe Walsh, Joe Yaworski, John Yuchimiuk, and Carrie Zerbe. Special thanks to Marryanne Slatowski and Jumei Wang for their advice and for keeping my spirits up.

I want to thank my good friend and colleague, Ahmed Bouzid, who patiently reviewed my text and corrected my English, and who made sure that I kept to my schedule. His valuable suggestions and constant encouragement took me a long way towards completing this work.

Very special thanks go to my friends Kang Yang and Professor Gene Hall. They gave me a lot of encouragement and help and I will always remember them for their kindness and positive attitude.

I affectionately thank my mother for her patience and her unconditional love, support, and encouragement during these long years away from home. My sister and my brother likewise deserve a great deal of gratitude for their support and for taking care of the family while I was away.

I dearly remember my late father who originally suggested my area of study when I entered college in China. I am comforted in the thought that he is proud that his daughter is now completing her PhD.

Pennsylvania, USA
December 1998

Abstract

In this thesis, we successfully apply connectionist approaches, particularly the Multi-Layer Perceptron (MLP), to tasks of speech recognition. We present in detail the Back Propagation theory and its implementation issues, including a modified weight adaptation algorithm. We provide a weight updating strategy to speed up the convergence during network training. The training data is balanced phonetically such that the network treats all phonemes equally. We introduce a random database generator to obtain a robust MLP network. We introduce the fuzzy MLP into speech recognition and use the overlapped Hamming window as the fuzzy membership function for the MLP output.

We design and implement the Multi-Layer Perceptron to be used as a labeler for the Hidden Markov Model (HMM) system, which combines the good short-time classification properties of MLPs with the good integration and overall recognition capabilities of discrete HMMs. The standard vector quantization has been replaced by an MLP labeler giving phone-like labels in an MLP/HMM hybrid system. Compared with using MLPs as probability generators for HMMs, our system is more flexible in system design because it can use the word models instead of phonetic models. Moreover, as it does not need to be trained to reach a global minimum, the network can have fewer hidden units and therefore can be trained faster. Also, we do not need to retrain our MLPs with segmentations generated by a Viterbi alignment. Compared to Euclidean labeling, our method has the advantages of needing fewer HMM parameters per state and of obtaining higher recognition accuracy.

We use histograms to illustrate the MLP output value for each phonetic class. From those MLP output histograms, we observe that the winner-

take-all MLPs ignore the relativity of different phonetic classes. We extend our base-line winner-take-all method to several Top-N methods. A series of MLP/HMM hybrid models are discussed to fully use the MLP output information and to improve the speech recognition performance. Those investigated models are: MLP multi-dimensional labeling, MLP multi-labeling, MLP fuzzy-labeling, multi-MLP multi-labeling and multi-MLP fuzzy labeling.

Glossary

List of Abbreviations

A/D	Analog/Digital
ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
CSR	Continuous Speech Recognition
DFT	Discrete Fourier Transform
BP	Back-Propagation
FVQ	Fuzzy Vector Quantization
HMM	Hidden Markov Model
IWR	Isolated Word Recognition
KWS	Keyword Spotting
LPC	Linear Predictive Coding
LVQ	Leaning Vector Quantization
MLP	Multi-Layer Perceptron
PLP	Perceptual Linear Prediction
TDNN	Time Delay Neural Network
VQ	Vector Quantization

List of Symbols

α	BP momentum parameter or pre-emphasis factor
η	BP learning rate
γ	Normalization factor

$\delta_t(j)$	The highest probability along a single Viterbi path ending at state j, at time t
$\phi_t(j)$	Viterbi path array
$\Phi_{xx}[k]$	FFT power spectrum at the k-th FFT frequency band
C_k	Class corresponding to the k-th MLP output or the k-th VQ cell
D	VQ distortion
$d(\mathbf{x}, \mathbf{z})$	VQ distortion between input spectrum $\mathbf{x}$ and the codeword $\mathbf{z}$
$D_i ... D_N$	TDNN time delay
E	Mean square error
E_p	MLP error for input pattern p
$E[z]$	Linear prediction error in the z-domain
F	FVQ degree of fuzziness
$F()$	Semi-linear function (sigmoid function)
F_s	Sampling frequency
G	Vocal tract excitation gain
$H[z]$	Vocal tract spectrum in the z-domain
L	Codebook length
M	Number of samples of frame shift
N	Window length
P	Linear prediction order
$P(O)$	HMM sequential output probability
$P(\mathbf{x}\|j)$	HMM a priori probability
$R(j)$	j-th autocorrelation coefficient
S_l	Output spectrum for the lth auditory spectrum
$X[z]$	Speech signal in the z-domain
a_{ij}	HMM transition probability from state i to state j
a_i	i-th linear prediction coefficient
b	Bark-scaled Frequency
$b_j(k)$	HMM observation probability for label k at state j
b_j	MLP bias on unit j
c_i	i-th cepstral coefficient
Δc_i	i-th first derivative cepstral coefficient
$\Delta^2 c_i$	i-th second derivative cepstral coefficient
d_{pj}	MLP target value on unit j for the input pattern p
$d_{norm}(j)$	j-th normalized distance
$d(\boldsymbol{x}, \boldsymbol{z}_j)$	Vector quantization error between vector $\boldsymbol{x}$ and the codeword $\boldsymbol{z}_j$
$e[n]$	Linear prediction error signal

f	Frequency
m	Mel-scaled Frequency or discrete time index
m_{ij}	Fuzzy VQ observation component
n	Discrete time index
n_i	Number of units in the MLP input layer
n_h	Number of units in the MLP hidden layer (only one hidden layer)
n_o	Number of units in the MLP output layer
q_i	i-th state in an HMM
$s[n]$	Sampled speech signal
$\hat{s}[n]$	Predicted speech signal
t	Time
w_{ij}	Connectionist weight between unit i in the lower layer to unit j in the upper layer
$\Delta_p w_{ij}$	Weight change for the input pattern p
$w_{q_t}(x)$	HMM observation probability for output $\mathbf{x}$ at state q_t
$w[n]$	Hamming window coefficient
$x[n]$	Windowed speech signal
x_{pj}	MLP input to unit j for the input pattern p
y_{pj}	MLP output value on unit j for the input pattern p
$\mathbf{z}_i$	i-th codeword vector
z_{pj}	MLP total input to unit j for the input pattern p

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Speech is the most natural form of human communication. What clinches the case in favor of speech recognition is the experimental evidence established by Ochsman and Cahpanis [61] in 1974, that voice communication is critical to the single- and multi-modality communication links. A pair of people communicated with each other via ten alternative communication channels, including voice, typewriting, handwriting, close circuit video and visual contact. They found that the communication centered around voice channel was fast and liberating, so that the user could move around and be involved in other tasks.

Artificial neural networks and computerized speech processing are two technologies which are still in their developmental stages. Research in speech processing is focused primarily on the man-machine interface. One of the goals of artificial neural networks (ANN) is to process information in a manner similar to that of biological neural systems. The common denominator between these two disciplines is the auditory processing mechanism we use to understand speech.

The speech recognition process is a fascinating phenomenon which involves numerous processing stages between signal detection and language understanding. It is an important scientific problem because in the future it may help us understand how humans think and form conceptual relationships. For instance, when listening to different languages, at what level of neural processing do multi-lingual individuals understand what is being

said? The answer is not known, although it has been shown that the auditory pathway must be trained to recognize phonemes which are unique to some languages. At a higher level, people often think to themselves in grammatically structured sentences. Does this "thinking" take place in an area of the brain which is stimulated by the auditory pathway? Unfortunately, the mechanisms by which humans understand speech are poorly understood. Much more research needs to be performed before many of these questions can be answered.

Automatic Speech Recognition (ASR) is an appropriate problem to investigate with ANNs. Two characteristics of ANNs must be noted. First, ANNs can be used in each part of traditional approaches, in which the recognition problem is often divided into signal processing, feature extraction, pattern matching and language modeling. This modularity is ideal for implementing and evaluating ANNs , especially using ANNs to solve subtasks of the ASR problems and integrating these ANNs into conventional recognition systems. In this manner, ANN and conventional speech recognition techniques can be examined within the same framework to determine which approach is best suited for specific subtasks. Second, since ANNs are the simulation of human neural system, they go one step ahead of the conventional mathematical models in understanding human speech production and perception.

1.2 Automatic Speech Recognition

The goal of Automatic Speech Recognition is to develop techniques and systems that enable computers to accept speech input. A typical digitized time domain speech signal is illustrated in Figure 1.1. The speech recognition problem may be interpreted as a speech-to-text conversion problem. Users want their voices, speech signals such as shown in Figure 1.1 to be transcribed into text by a computer. From a commercial point of view, speech recognition is a technology with a potentially large market. Since the early 80's, compact implementations of accurate, real-time speech recognizers have found widespread use in many applications, including voice-activated transcription, simplified man-machine communication, aids for hearing impaired individuals and the physically disabled, telephone assistance, and other man-machine interface tasks.

As one goes from problem solving tasks such as puzzles and chess to

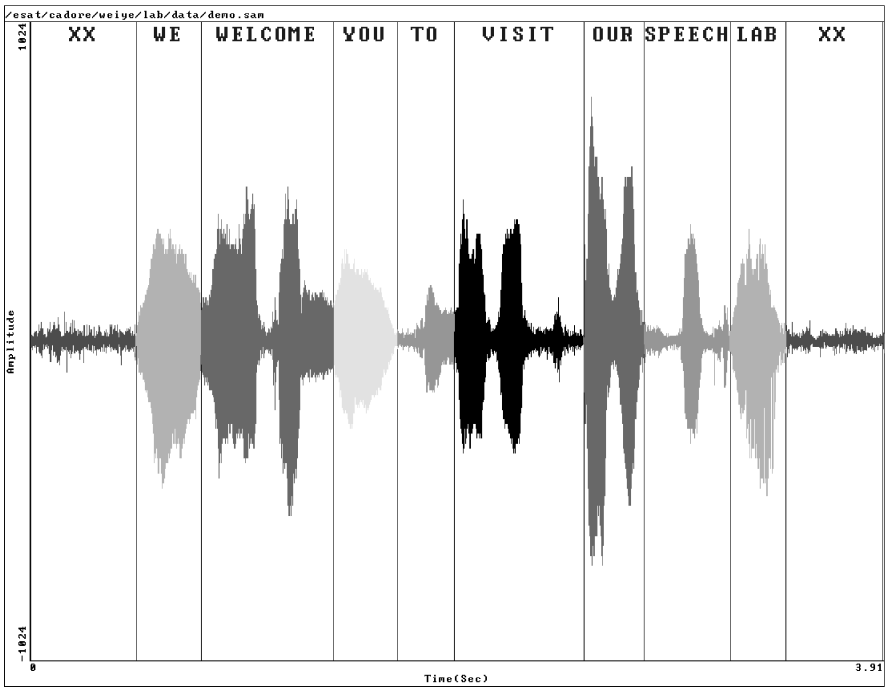

Figure 1.1: A time domain speech signal

perceptual tasks such as speech, the problem characteristics change dramatically: knowledge-poor to knowledge-rich, low data rates to high data rates, slow response time (minutes to hours) to virtually instantaneous response time. Taken together, these characteristics increase the computational complexity of the problem by several orders of magnitude. Further, speech provides a challenging task domain which embodies many of the requirements of intelligent behavior: operating in real time, exploiting vast amounts of knowledge, using symbols and abstractions, communicating in natural language and learning from the environment.

Unfortunately, current speech recognizers perform poorly on speaker independent continuous speech recognition tasks that people perform without apparent difficulty. Although children learn to understand speech with little explicit supervision and adults take speech recognition ability for granted, speech recognition has been proven a difficult task to duplicate with machines. This is due to the variability and overlap of information in the

acoustic signal, the need for high computation rates (a human-like system must match inputs to 50,000 words in real time), the multiplicity of analyses that must be performed (phonetic, phonemic, syntactic, semantic, and pragmatic), and the lack of any comprehensive theory of speech recognition.

The dominant technological approaches for speech recognition systems are based on pattern matching of statistical representations of the acoustic speech signal, such as Hidden Markov Model (HMM) whole word and subword (e.g., phoneme) models. However, although significant progress has been made in the field of ASR these last years, the typical vocabulary size is still very limited and the performance of the resulting systems is still not comparable to that achieved by human beings.

1.2.1 The Challenge of Speech Recognition

Humans are able to understand speech so easily that we often fail to appreciate the difficulty that this task poses for machines. Here are some dimensions in which machine performance faces the challenge:

- *Speaker dependence or independence*

  The operating mode of speech recognition systems can be divided into three categories: speaker-dependent, speaker-independent and multi-speaker, each involving different training paradigms. A speaker-dependent system uses speech from the target speaker to learn its model parameters. This strategy leads to good accuracy, but requires a new training session that has to be done for every new speaker and a large memory that has to be used to store specific models for every user. On the other hand, a speaker-independent system is trained once, and must model a variety of speaker voices. Due to their increased variability, speaker-independent systems are typically less accurate than speaker dependent systems. Multi-speaker systems work well for certain groups of speakers for which the system has been trained. Gender and regional accent dependent systems fall into this category. Multi-speaker and speaker-independent systems can also be constructed to adapt to the current user.

- *Spontaneous speech*

  For a system to be useful in a real life application it has to accept spontaneous speech. Most current ASR systems, however deal with read

speech only. The practical difference is a host of disfluencies that people produce, for instance, filled pauses ("um" or "er") or false starts. Additionally, in natural speech, talkers will almost certainly use some words that are outside of the recognizer lexicon. Spontaneous speech is often ungrammatical and ill-structured.

- *Vocabulary size*

  The vocabulary size varies inversely with the system accuracy and efficiency — more words introduce more confusion and require more time to process. For large vocabularies, it is also not generally feasible to work with the whole word models, since this would require not only a prohibitive amount of training data but a large amount of parameters, particularly for a speaker-independent system. Instead, one must turn to smaller sub-word units (phoneme, syllables), which may be more ambiguous and harder to detect and to recognize. In order to realize a large vocabulary system, it is essential to look for a compact presentation such as sub-word units.

- *Task and language constraints*

  In most cases, the task of continuous speech recognition is simplified by restricting the possible utterances. This is usually done by using syntactic and semantic information to reduce the complexity of the task and the ambiguity between words. However, this is still a very active research area since it is not known how to properly interface general grammars and natural speech constraints with acoustic recognizers. The use of semantic information is still an open issue. Since the degree to which these non-acoustic knowledge sources limit the possible utterances can vary, vocabulary size is not a good measure of a speech recognition task's difficulty. The constraining power of a language model is usually measured by its complexity–roughly the geometric mean of the number of words that can occur at any decision point. High complexity generally implies a high level of difficulty for a task, as many word candidates must be examined by the acoustic recognizer.

- *Acoustic ambiguity, confusability*

  While some recognizers may achieve respectable performance over relatively unambiguous words (e.g., 10 digits), such systems may not

necessarily deliver acceptable recognition rate to some words, e.g., the words for the E-set alphabetic letters, B,D,E,P,T,C,Z,V,G. A confusable vocabulary requires detailed high-performance acoustic pattern analysis.

- *Adverse conditions*

  Several adverse conditions that can alter the performance of ASR systems have been identified:

  - environmental noise, i.e., stationary or non-stationary additive noise (e.g., factory floor, car, cockpit, door slams);
  - distorted acoustics and speech correlated noise (e.g., reverberant room acoustics, non-linear distortions);
  - different microphones (e.g., telephone receiver, headset microphones, table microphones) and different filter characteristics, which usually lead to convolutional noise;
  - limited frequency bandwidth (e.g., telephone channels where the transmitted frequencies are limited between approximately 330 Hz and 3,300 Hz);
  - altered speaking manner, (e.g., Lombard effect, differing speaking rate, speaker stress, breath and lip noise, pitch, uncooperative talker, etc.);
  - a combination of the previous issues (which is, unfortunately, the most frequent case).

- *Isolated, connected, and continuous speech*

  Isolated Word Recognition (IWR), involves the recognition of single words. The speaker may pronounce these words in isolation, or in an utterance consisting of several discrete words separated by distinct pauses. Connected speech recognition, in which each word is clearly articulated and there are no pauses between words in the utterance, is a more difficult problem. Because there are no discernible pauses, and coarticulation effects may be present, difficulties arise when attempting to determine word boundaries. Continuous Speech Recognition (CSR) is the most difficult ASR problem. There are no pauses between words and the words in the utterance are not always articulated clearly. The demand for CSR systems range from

	Isolated word		Connected speech		Continuous speech	
Speaker dependent	small vocab.	1	small vocab.	4	small vocab.	5
	large vocab.	4	large vocab.	5	large vocab.	6
Multi-speaker	small vocab.	2	small vocab.	4	small vocab.	6
	large vocab.	4	large vocab.	5	large vocab.	7
Speaker independent	digits	3	digits	4	digits	5
	large vocab.	5	large vocab.	8	large vocab.	10

Table 1.1: The complexity of the ASR problem expressed as a function of operating mode and speaking rate

small vocabulary recognition systems (< 100 words), to moderately sized (around 1000 words) domain-specific vocabularies, to large-vocabulary (> 5000 words) systems targeted for the office environment. In many large-vocabulary systems, continuous speech recognition implies the recognition of natural speech, where the speaker is not constrained by vocabulary size, speaking rate, or grammatical usage. Another problem that is neither isolated word recognition nor connected speech recognition (but which is as difficult as CSR) is often referred to as Keyword Spotting (KWS). In this case, one wants to detect "keywords" from unconstrained speech, while ignoring all other words or non-speech sounds. This is a kind of generalization of an isolated word recognition system in which the user is not constrained to pronounce words in isolation. The challenging problem of rejecting utterances with no keywords is the focus in a KWS system.

Table 1.1 illustrates a typical partitioning of the ASR problem with respect to its modes of operation and speaking rate (which may be accommodated). Problem difficulty is assessed on a scale of 1 to 10, with 10 being the most difficult, and a 5 representing the current state-of-the-art in commercially available systems. Each of these dimensions of difficulties embodies some aspect of speech variability, which is the central problem of speech recognition.

1.2.2 The State of the Art

Speech Recognition Technology

The current best performing speech recognition algorithms use Hidden Markov Model (HMM) techniques in the commercial market. The HMM approach provides a framework which includes an efficient decoding algorithm for use in recognition (the Viterbi algorithm) and an automatic supervised training algorithm (the forward-backward algorithm). HMM, however, has its own performance limitations. These include poor low-level and high-level modeling. Poor low-level acoustic-phonetic modeling leads to confusions among acoustically similar words while poor high-level speech understanding or semantic modeling restricts applications to simple situations where finite state or probabilistic grammars are acceptable. In addition, the first-order Markov assumption makes it difficult to model coarticulation directly and HMM training algorithms can not currently learn the topological structure of word and sub-word models. Finally, HMM theory does not specify the structure of implementation hardware. It is likely that the high computation and memory demand of the current HMM algorithms may require parallel hardware design to produce a compact, large-vocabulary, continuous-speech recognizer.

Table 1.1 also provides the information on the current state-of-the-art of speech recognition technology. The first ASR system which appeared in the commercial marketplace perform very constrained speech recognition tasks and filled several "niches" in terms of user needs. These systems gradually progressed so that large vocabulary "multi-speaker" discrete utterance recognition, speaker-independent continuous digit recognition, and "moderate-sized" vocabulary speaker-dependent connected speech recognition and large vocabulary speaker-independent continuous speech recognition have become available.

Recognition Accuracy

Speech recognizers naturally have a wide range of accuracies. Accuracy for a difficult 997-word speaker-independent high-quality continuous speech task using a strong language model (an average of only 20 different words possible after any other word) can be as high as 96% (Lee and Hon 1989 [44]). This word accuracy score translates to an unacceptable sentence accuracy of roughly 50%. In addition, the word accuracy of this high-

performance recognizer is typically below 70% correct when tested with no grammar model. Results such as these have illustrated the poor low-level acoustic phonetic matching provided by the current speech recognizers. These recognizers depend heavily on constraining grammars to achieve good performance. There is a very high accuracy demand for digit recognition. An accuracy of 99.9% has been reached for recognition of high-quality, microphone, read, legal credit-card numbers. For low-quality speech such as telephone speech, some commercial products have obtained an accuracy of 99.5% (Ma 1997 [51]). In spite of the promising progress over the past few years, speech recognition by machines still has a long way to go to reach a generally applicable real system.

Real World Speech Application

Most speech recognition applications are constructed to perform data entry, command-and-control, information access and dictation. The voice input devices, microphone and telephone, divide the speech applications into two directions: PC multimedia and computer telephony. In the same level application, speech recognition is more difficult when using telephone than when using microphone. In a large vocabulary dictation system, a user has to train the system first in order to reach a high recognition accuracy. However, this method is not applicable to telephone applications because of speaker-independent requirement.

Computer Telephony The ability to use speech recognition over the telephone has been an industry goal for forty years, but it was not viable until the mid-80's. Commercial success did not come until the beginning of the 90's. Speech recognizers are required to handle telephone quality speech in real time. Some vendors such as *Lernout & Hauspie Speech Products*, *Nuance Communication* and *Purespeech* have exported their recognizer to Dialogic Antares DSP cards in order to gain speed and to handle multiple calling instances. A long-established interaction paradigm, the prompt-and-response pattern in IVR (Interactive Voice Response) systems, has been imported to speech recognition. The use of grammars limits the active vocabulary and improves recognition performance. Well-crafted dialog sequences can significantly increase overall recognition performance, as they steer the speaker to utter words within the recognizer's vocabulary. The following guidelines are used in the design of a dialog.

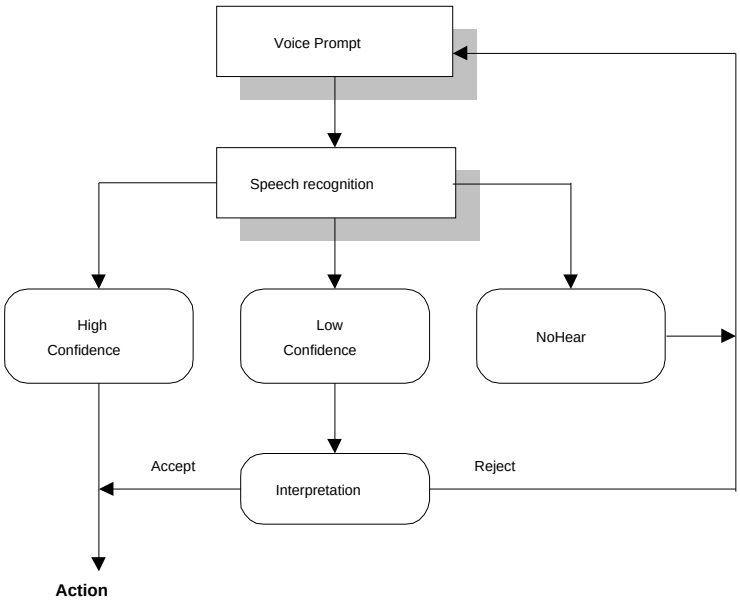

Figure 1.2: Dialog design with error-recovery

- Use simple and less ambiguous grammars for the recognizer.

- Prompt and convey the user to speak words or sentences within the grammar.

- Provide an elegant error recovery.

Figure 1.2 illustrates the appropriate error handling scheme, in which the confidence level for the recognition result is the basis to control the dialog flow. Herb Clark says that "speaking and listening are two parts of a collective activity"[18]. A major design challenge in creating speech applications, therefore, is to simulate the role of speaker/listener convincingly enough to produce successful communication with the human collaborator.

1.3 Neural Networks in Speech Recognition

1.3.1 Artificial Neural Networks

Artificial neural networks consist of multiple processing units or nodes, which are analogous to nerve cells in the brain that receive and transmit electrochemical signals. Each neuron is connected to many others through "synapses" of various strengths, like the connections between neurons in the brain. Unlike conventional computers and expert systems, which rely on a single central processing unit and very meticulously programmed instructions, neural networks rely on many nodes and synapses acting in concert and a very simple "instruction set" for each node. By changing the strength of synapses, neural networks can be trained by examples to perform desired input-output transformations.

The complex operation of neural networks resulting from abundant feedback loops, which together with non-linearities of the processing elements and adaptive changes of their parameters, can define very complicated dynamic behavior.

Hence, the central and crucial problem in neural network is the development of training and learning strategies and algorithms, i.e., methods of programming a neural network adaptively, so that it performs the desired interaction with a changing or unknown and fuzzy environment such as speech processing and recognition.

1.3.2 Benefits of Neural Networks

The performance of current speech recognizers is far below that of humans. Neural networks offer the potential for providing massive parallelism, adaptation and new algorithmic approaches to speech recognition problems. Neural networks for speech recognition have been explored as part of the recent resurgence of interest in this area. Research has focused on evaluating new neural network pattern classification and training algorithms using real speech data, and on determining whether parallel neural network architectures can be designed to perform the computation required by important speech recognition algorithms. The neural network has been given serious consideration for speech recognition for several reasons. They include the following:

- ANNs can readily implement a massive degree of parallel computation. Because neural networks are highly parallel structures of simple, identical, computational elements, it should be clear that they are prime candidates for massively parallel (analog or digital) computation.

- They intrinsically possess a great deal of robustness or fault tolerance. Since the information embedded in the neural network is propagated to every computational element within the network, this structure is inherently among the least sensitive of networks to noise or defects within the structure.

- The connection weights of the network need not be constrained to be fixed; they can be adapted in real time to improve performance. This is the basis of the concept of adaptive learning, which is inherent in the neural network structure.

- Because of the non-linearity within each computational element, a sufficiently large neural network can approximate any nonlinearity or nonlinear dynamic system. Hence neural networks provide a convenient way of implementing nonlinear transformations between arbitrary inputs and outputs and are often more efficient than alternative physical implementations of nonlinearity. Nonlinearity is a highly important property of speech signals.

1.3.3 Using Neural Networks for Speech Processing and Recognition

Speech recognition, like most difficult problems, can be divided into a number of challenging sub-problems. Examples of these problems include a proper spectral and temporal representation of speech, the definition of the fundamental units of speech, the degree of sensitivity of speech units to context, and the syntax and semantics of language. Problems such as ambient noise rejection and sound location are also important factors in our perception of speech.

The only system that completely and effectively solves all of these problems is the human auditory system, which implements a biological neural network. The auditory system is a highly structured apparatus characterized by rich anatomical and physiological diversity. Instead of a single

continuous network, the auditory system consists of a complex linkage of specialized sub-networks, each of which adds, processes, or extracts information for speech recognition.

While ANNs have already proved useful in recognizing isolated speech units by using some of the techniques described below, they have only recently begun to make serious inroads into large vocabulary ASR systems. Indeed, nearly all ANN systems for ASR are just static pattern classifiers – given labeled and segmented training data, a network can be trained to recognize isolated speech segments. However, the most general form of ASR should accept continuous speech as input. Any such ASR system, whether it uses ANNs or not, must perform a dynamic recognition process, in which the input speech is segmented (perhaps implicitly) as well as classified, so that the output is a succession of words which explain the acoustical input. Furthermore, linguistic constraints (both syntactic and semantic) are a component of most ASR systems. Because of these considerations, as well as the inherent difficulty of robust ASR, applying ANN methods to ASR is a challenging research area.

Many new neural network models have been proposed for recognizing temporal pattern sequences. Some are based on physiological data and attempt to model the behavior of biological networks (Dehaene et al. 1987 [20], Cohen et al. 1987 [19]), while others attempt to extend existing auto-associative networks to temporal problems (Amit 1988 [1], Buhmann and Schilten 1988 [14], Kleinfield 1986 [35]). However, new learning algorithms and network architectures will be required to provide real-time response and automatic learning of internal word and phrase models required for high-performance continuous speech recognition. This is still an important open problem in the field of neural speech recognition. The current research focuses on the following field:

- New physiological-based front ends;
- Neural network classifiers for static speech input patterns;
- Neural networks designed specifically to classify temporal pattern sequences;
- Hybrid systems that integrate neural network and conventional speech recognition approaches;
- Neural network architectures that implement conventional algorithms;

- VLSI hardware neural networks that implement both neural network and conventional algorithms.

1.4 The Goal of This Dissertation and the Main Results of the Work

The main contributions of our work to neural networks in speech recognition are:

- We propose a weight updating strategy to speed up the convergence during network training. The training data is balanced phonetically such that the network treats all phonemes equally. We introduce a random database generator to reach a robust training. From the network point of view, we find optimal speech features and remove the redundant information for speech recognition.

- We design a Multi-Layer Perceptron (MLP) network. As it has much less hidden nodes and has no restrictions to converge to a global minimum, the network is more stable and takes less time to train, compared with using MLPs as probability generators for HMMs. During training, we use newly random generated data for each iteration and make connectionnist weight adaptation on the average of the accumulated weight changes. We use less training data to obtain a robust network.

- We give the MLP a fuzzy interpretation and utilize the overlapped Hamming window as the fuzzy membership function for the MLP output.

- We propose using MLP as a labeler for HMMs (Hidden Markov Model). Instead of using the conventional Vector Quantization(VQ), which is obtained by using K-means clustering, we implement an MLP/HMM system using MLP-VQ. As MLP has a free input pattern selection and a unique non-linear mapping capability, it overcomes the disadvantage of conventional linear VQ and the input pattern restrictions in HMMs. Standard vector quantization has been replaced by an MLP labeler giving phone like labels. Compared to Euclidean labeling, our method has the advantages of needing fewer HMM parameters per state and obtaining a higher recognition accuracy.

- Several improvements of the above baseline MLP labeling are investigated. We introduce Top-N VQ and fuzzy VQ methods to improve the baseline winner-take-all method. Our purpose is to decode all the information on each node and each connectionist weight, especially the output nodes. In Top-N VQ, we consider the Top-N labels instead of the Top-one label as the output vector. We expand our approach to multi-code book HMMs. In MLP fuzzy VQ, we interpret each MLP output value as the closeness measurement to each output label. We modify the semi-continuous HMM and make an MLP fuzzy VQ HMM.

- We investigate the MLP labeler systematically and also present several other hybrid MLP/HMM models which include the MLP multi-labeling, the multi-MLP multi-labeling and the multi-MLP fuzzy-labeling.

1.5 Outline of the Dissertation

This dissertation consists of two main parts. In the first part we introduce the conceptual and theoretical background of a wide range of techniques investigated in this work, including speech signal processing, speech recognition and artificial neural networks. In the course of the introduction we attempt to address some of the major limitations associated with the conventional approach to speech recognition, and we discuss the possibility and development of incorporating artificial neural network to overcome these limitations. In the second part, we focus on the implementation of the speech recognition system using neural networks.

Presented in chapter 2 is an introduction to the fundamentals of the conventional approach to speech recognition. We start with the physical speech production mechanism and proceed to describe the information encoded in the acoustic signal. A general speech recognition model provides the framework for our discussion on speech recognition algorithms. Within this framework, several speech processing and speech recognition algorithms are presented, most of which have been implemented during the work of this dissertation.

In chapter 3, we introduce the mathematical model of the Multi-Layer Perceptron(MLP). We give a detailed description about MLP neural networks from network structure to network training. We present the ap-

plications using MLPs: MLP phonetic frame recognition and Time Delay Neural Network (TDNN) phoneme recognition. We give a fuzzy interpretation to the MLP neural network based on the fuzzy set theory with which we analyze the MLP training process and its output.

Based on our interdisciplinary study of the subjects presented in the previous chapters, we proceed to present our concrete work in the development of automatic speech recognition tailored to Vector Quantization(VQ). In chapter 4 we describe in detail various methods we have investigated or used in accordance with the conventional paradigm of vector quantization. We present a series of hybrid MLP/HMM models. These models are MLP-labeling, MLP fuzzy-labeling, MLP multi-dimensional labeling, MLP multi-labeling, multi-MLP multi-labeling and multi-MLP fuzzy-labeling.

We present our experimental results in chapter 5. The proposed data weighing, random data selection, MLP training, MLP-VQ, MLP fuzzy VQ and MLP-HMM hybrid system are discussed here. We first build our training and test speech database from the existing audio database, construct our own MLP, then feed those data to the MLP to train and test our MLP/HMM system. Finally, we evaluate our experimental results, comparing them with different techniques.

Chapter 6 concludes the dissertation and presents a perspective for future developments in the research of artificial neural networks and automatic speech recognition.

Chapter 2

Speech Processing and Speech Recognition

2.1 Introduction

This chapter describes the speech recognition problem, which is essentially the focus of the entire text. In addition, later chapters introduce ANNs, which are based upon conventional recognition techniques described in this chapter. We begin by reviewing the physical speech production mechanism and proceed to describe the information encoded in the acoustic signal. A general speech recognition model provides the framework for our discussion of speech recognition algorithms. Within this framework, several speech processing and speech recognition algorithms are presented, most of which were implemented during the work of this dissertation.

2.2 Human Speech Production and Perception

2.2.1 Human Speech Production and Perception

Figure 2.1 illustrates the process of speech production and perception in human beings. The production process begins when a speaker formulates a message in the brain. The next step in the process is the conversion of the message into a language code. Once the language code is chosen, the speaker executes a series of neuromuscular commands to cause the vocal cord to vibrate when appropriate, and to shape the vocal tract such that

the proper sequence of speech sounds is created and spoken by the speaker, thereby producing an acoustic signal as the final output.

When the speech signal is generated and propagated to the listener, the speech perception process begins. First, the listener processes the acoustic signal along the basilar membrane in the inner era, which provides a spectrum analysis of the incoming signal. A neural transduction process converts the spectral signal at the output of the basilar membrane into activity signals on the auditory nerve, corresponding roughly to a feature extraction process. In a manner that is not well understood, the neural activity along the auditory nerve is converted into a language code at the higher centers of processing within the brain, and finally message comprehension (understanding of meaning) is achieved.

Considering all of the variables in the physiological structure of the vocal tract, and the discernible fact that virtually all individuals sound different, it is remarkable that humans can understand one another. This leads one to believe that there is a significant amount of phonetic information encoded in the acoustic speech signal which is not affected by speaker-dependent characteristics. In addition to the specific sounds generated, this information includes the syntax, or rules, one uses to construct the language.

To understand how particular sounds can be classified, and how to decode the speech signal for automatic speech recognition, it is important to understand how the acoustic signal is produced first. Figure 2.2 is a simplified model of the speech production system. This model shows that the vocal tract is excited by unvoiced components of the *breath noise* and/or an *impulse train* from the glottis. The periodicity of the glottal pauses is called the fundamental frequency, or pitch, of the speech signal. A glottal pulse is generated from the accumulation and sudden release of pressure behind the glottis. The other excitation source, breath noise (which is for all practical purposes white noise), can be obtained simply by exhaling.

2.2.2 Information in the Speech Signal

Formants

Figure 2.3 illustrates the 8kHz sampled time-domain speech signal from Dutch digit *three* and its corresponding frequency domain representation-spectrum. The voiced speech in the time-domain signal is characterized by high-energy periodic peaks which correspond to the glottal pulse. The

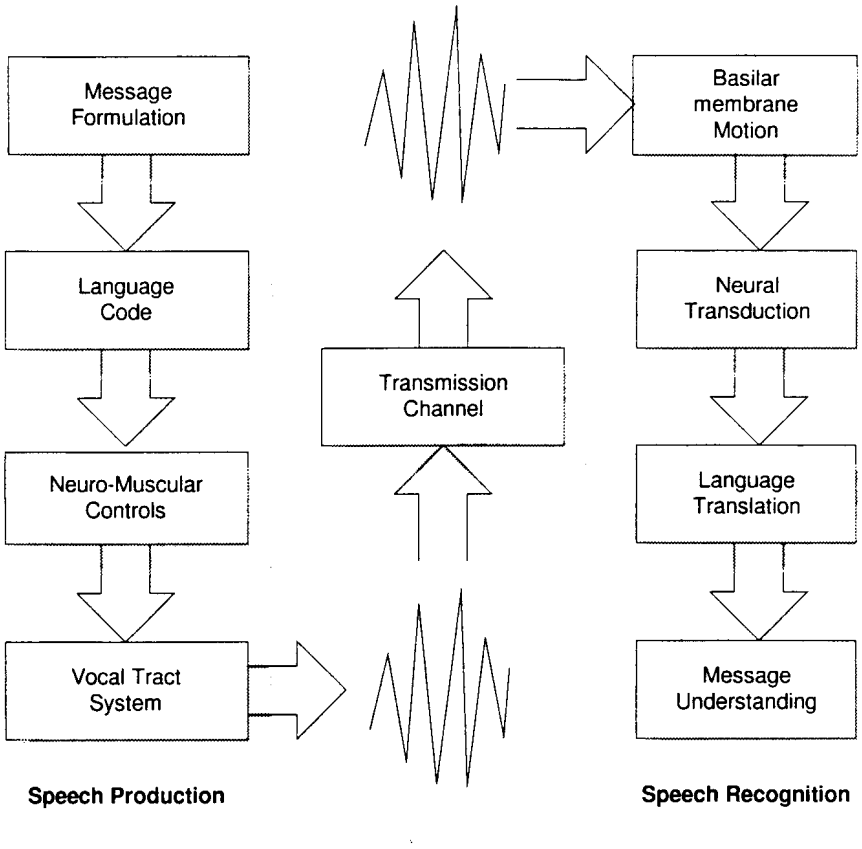

Figure 2.1: Speech production and perception process

periodicity of these peaks is the fundamental frequency. The spectrum is a two-dimensional display of the energy of the frequency components of a signal over time. The dark bands in the spectrum correspond to frequencies with high energy, which are resonant frequencies because they are indicative of the location of the poles in the transfer function. Those resonant frequencies of the vocal tract transfer function are called *formants*.

There are typically about three resonances of significance below 3500 Hz. The first formant, *f1*, is the lowest resonant frequency. The lowest two formants (and sometimes the lowest three formants) are usually sufficient to identify specific phonemes, while the location of the higher formants is

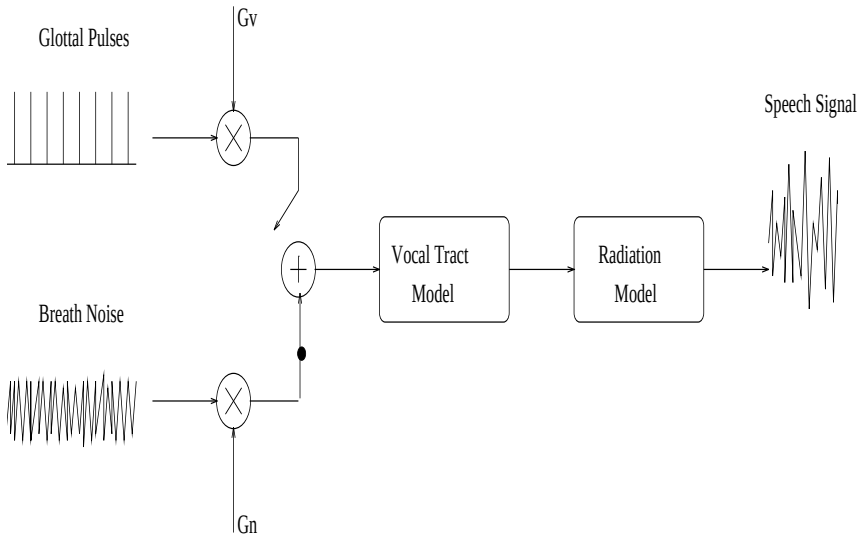

Figure 2.2: A module of the speech production system.

generally speaker dependent. The major problem, however, is the difficulty of reliably estimating the formants for low-level voiced sounds, and the difficulty of defining the formants for unvoiced or silence regions. Notice the movement and energy variation of these poles in the spectrum over time. From the time and frequency domain information typical of Figure 2.3, auditory processing mechanisms are apparently able to identify the acoustic cues in the signal and extract the necessary information for speech understanding.

Phonemes

Phonemes are the fundamental units used to pronounce a word. For this reason, some ASR systems perform recognition at the phoneme level. However, the actual pronunciation of a phoneme can vary appreciably between speakers, given both physical effects and effects due to regional accents. Phonemes are constructed from permutations of voice, tongue, mouth, jaw and lip positions. A large number of phonemes can be created from these permutations, and many of those occur in other languages.

American English phonemes are organized into four major groups: vowels, consonants, diphthongs and semi-vowels. These groups are further

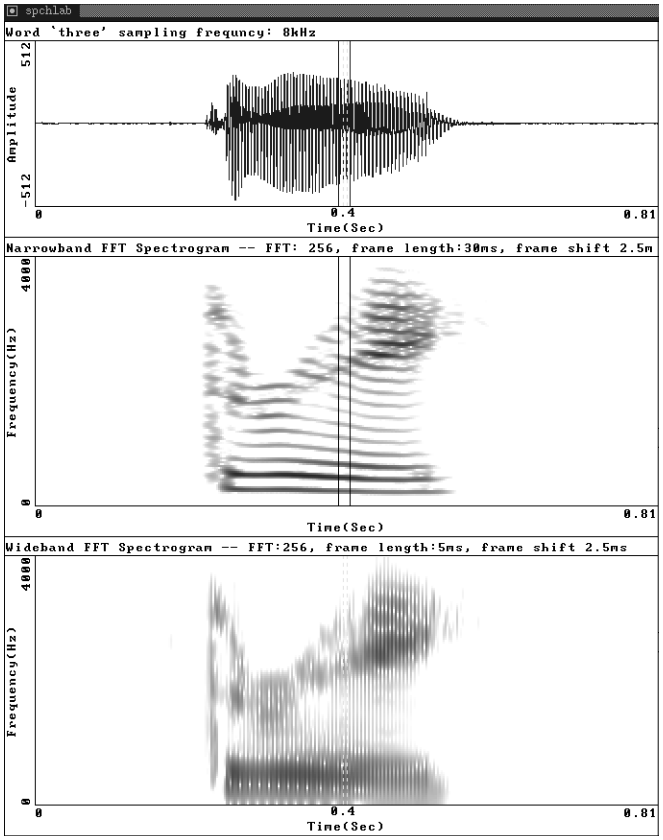

Figure 2.3: Time-domain representation of the speech signal (top), its narrow-band (middle) and its wide-band spectrograms (bottom).

subdivided into type and place of articulation. The vowels, semi-vowels, nasals, fricatives and whispers are composed of *steady-state* sound. The other diphthong, stops and affricates are sounds created during changing vocal tract configurations.

The ultimate goal of speech recognition is to uniquely and automatically provide a segmentation and labeling of speech into constituent sounds or sound groups such as syllables, words, then sentences. The highest energy in the signal usually indicates a vowel because the speech is voiced and the mouth is open. As the mouth is closed, the signal is increasingly attenuated. This exercise provides general information concerning the differences

between phonetic classes. However, as one might expect, it is very difficult to distinguish among class members. One of the reasons is coarticulation.

Coarticulation

Speech production involves a sequence of articulator gestures timed so that certain key aspects of vocal tract shape occur in an order corresponding to the intended phoneme sequence. Gestures for successive phonemes overlap in time so that the vocal tract shapes during a phoneme are highly dependent on the phoneme's context.

There are two phenomena that coarticulation involves. One is the articulation of one phoneme while the vocal tract is to configure itself for the articulation of the next phoneme. When this transition occurs in continuously spoken speech, the transeme effect can resemble a diphthong. Another phenomenon invokes coarticulation, which occurs when the phoneme at the end of one word and at the beginning of the next, is shared. These phenomena are just a few of the problems associated with recognizing continuously spoken speech.

While coarticulation causes significant problems for automatic speech recognition, its effects aid speech perception. Most phonemes can be identified by portions of the speech signal from the middle (i.e., steady-state portions) of their phonemes.

2.3 A General Speech Recognition System

The study of speech recognition is based on three principles. First, the information in the speech signal can be accurately represented by the short-term amplitude spectrum of the speech waveform. This allows us to extract features based on the short-term amplitude spectrum from speech and to confidently use these features as the basis for pattern matching. Second, the contents of the speech signal can be expressed in written form. Furthermore, the meaning of speech can be written as a sequence of a few dozen symbols selected from the characters in an alphabet or from the phonetic symbols in a lexicon. This principle gives us confidence that the meaning of an utterance is preserved as we transcribe a sequence of acoustic features to a sequence of phonetic symbols. Third, speech recognition is a cognitive process. In human speech understanding, it is impossible to

separate perception of sounds from the grammatical, semantic, and pragmatic structures of language. Because spoken language is meaningful, both semantic and pragmatic information can be valuable guides to speech recognition when acoustic information alone is ambiguous. Unfortunately, current speech recognizers, in their simplicity, have taken little advantage of semantic and grammatic structures except in small, constrained tasks, such as the grammar-based speech recognition.

A block diagram of a general speech recognition system is shown in Figure 2.4. A speech signal is input to the recognition system and a classification decision is obtained from the system output. Four major operations are required.

- a signal processing module for obtaining a representation of the speech signal,
- a feature extraction module for identifying the key components of this representation and eliminating redundant information,
- time alignment and pattern matching modules for performing phoneme and word detection, and
- language processing module for selecting a final word string.

This structure of the system makes ANNs applicable to ASR, since one can select an appropriate ANN for a specific problem and insert it into a conventional recognition system. It also makes ANNs comparable with conventional recognition systems as we can freely select a convenient signal representation (for signal processing), feature presentation (for feature extraction), or symbolic representation (for language processing). A description of each part in the structure is given below.

2.4 Signal Processing and Feature Extraction

The purpose of signal processing is to derive a set of parameters to represent speech signals in a form which is convenient for subsequent processing and to process the sampled speech signal and produce a representation which is independent of amplitude variations, speaker stress, and noise which is introduced from the transmission medium. Both time domain and frequency domain approaches can be used. Time domain approaches, such as parameters of energy and zero crossing rate, directly dealing with the waveform

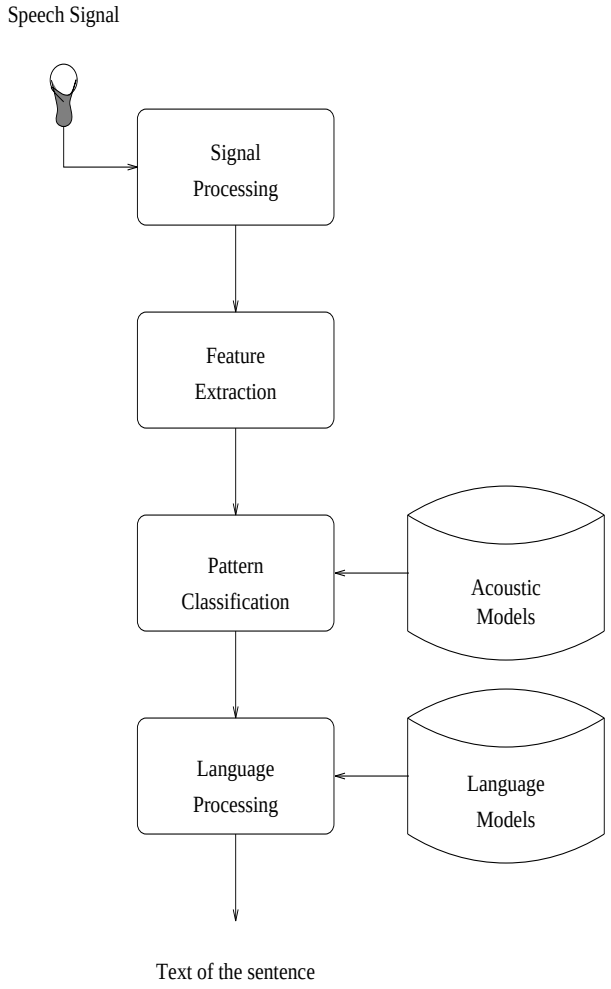

Figure 2.4: A general speech recognition system

of the speech signal, are usually simple to implement. Frequency domain approaches involve some form of spectral analysis that are not directly evident in the time domain. The latter approaches are more widely used in speech recognition.

2.4.1 Recording, First Order Pre-emphasis, Windowing and Buffering

Recording

Encoding speech from sound is by no means a simple task. First, the spoken word must be sampled, digitized, and filtered to remove the background noise added by the speaker's environment and the listening apparatus. The quality of speech varies widely, from a conversation in a relatively quiet, controlled condition such as an office, to a telephone conversation in a train station. Digital anti-aliasing filtering is used to emphasize frequencies that most likely contain key speech energy, and to compensate for nonlinearities in the recording process. As the telephone channel bandwidth is limited to 300-3000 Hz, the most popular sampling frequency is 8kHz in digital telephony.

Pre-emphasis

The digitized speech signal, $s(n)$, is put through a first-order FIR filter. It is common to apply pre-emphasis to speech signals before any further analysis. This operation compensates for the slope in the natural speech spectrum of about -6 dB/Oct [48], as hearing is more sensitive above the 1 kHz region of the spectrum. In the time domain, it is presented as:

$$s^{'}(n) = s(n) - \alpha s(n-1) \quad 0.95 \leq \alpha \leq 1.0 \tag{2.1}$$

α is always close to 1.0, typically 0.95. In the frequency domain, it looks like:

$$H(z) = 1 - \alpha z^{-1} \tag{2.2}$$

Buffering

In this step the preemphasized speech signal, $s^{'}(n)$, is blocked into frame buffers of N samples with an adjacent frame separated by M samples. Speech is typically analyzed in overlapping short frames of about 30 msec

long with a 10 msec frame shift. If the sampling frequency is 8 kHz, then $N = 240$ and $M = 80$. Let $x_i(m)$ represent the i-th short-time signal buffer, then:

$$x_i(m) = s^{'}(m + iM) \quad m = 0, \ldots, N-1 \tag{2.3}$$

Windowing

The next step in the process is to window each individual frame so as to minimize the signal discontinuities at the beginning and end of each frame. If we define the window as $w(m)$, then the result of windowing is the signal $x_i^{'}(m)$:

$$x_i^{'}(m) = x_i(m)w(m) \quad m = 0, \ldots, N-1 \tag{2.4}$$

A widely used window in speech recognition is the Hamming window, which has the form:

$$w[m] = 0.54 - 0.46\cos\left(\frac{2\pi m}{N-1}\right) \quad m = 0, \ldots, N-1 \tag{2.5}$$

In energy and spectral estimation, it is often required to have spectra independent on the window length N and the window type. We calculate the normalization factor γ as:

$$\gamma = \sum_{m=1}^{N} w^2[m] \tag{2.6}$$

Thus,

$$x_i^{'}(m) = \frac{x_i(m)w(m)}{\gamma} \quad m = 0, \ldots, N-1 \tag{2.7}$$

2.4.2 Spectral Analysis

Mathematical techniques such as Fourier transforms and linear prediction on coefficients are used to quantify the power and the fundamental frequency of the samples, which are then concatenated to a single parameter vector for each frame. A speech recognition system can improve its speed and accuracy by restricting its analysis to those combinations of frequencies which are perceptually meaningful to the human auditory system (ear and

brain). The following spectral analysis algorithms have been implemented and used in our speech recognition systems.

Auditory Spectral Analysis

The auditory filter bank is one of the most fundamental concepts in speech processing (O'Shaughnessy [63]). An auditory filter bank can be regarded as a crude model of the initial stage of transduction in the human auditory system based upon the theory of the critical bandwidth and logarithmic scale in frequency. A perceptual measure, called the *Bark* scale or critical-band rate, relates acoustical frequency to perceptual frequency resolution. A mapping of acoustic frequency f to a perceptual frequency scale b is defined as follows (O'Shaughnessy [63]):

$$b = 13atan(0.76f/1000) + 3.5atan\left((f/7500)^2\right) \tag{2.8}$$

A more popular approximation to this type of mapping in speech recognition is known as the *Mel* scale (O'Shaughnessy [63]):

$$m = 7arcsinh\left(\frac{f}{650}\right) \tag{2.9}$$

m is the *Mel* scale perceptual frequency. Both *Mel* scale and *Bark* scale are displayed in Figure 2.5. They are approximated as a linear scale from 0 to 1 kHz, and a logarithmic scale beyond 1 kHz. The Mapping from the perceptual frequency to the real frequency is presented as:

$$f = 650\sinh\left(\frac{m}{7}\right) \tag{2.10}$$

$$f \approx 1315.8\tan\left(\frac{b}{13}\right) \quad b < 5.5 \tag{2.11}$$

$$= 1000.0 \times 10^{\frac{b-8.7}{14.2}} \quad b > 5.5 \tag{2.12}$$

The main application of auditory frequency scales is in the design of filter banks. In speech recognition applications, the very low frequency range ($< 200Hz$) is beyond the capacity of the A/D converter and the data collected in this frequency range are extremely noisy. The critical bands with center frequencies below 250 Hz (bands 1-2) are discarded. In order to cover the telephone bandwidth, it can be seen that 15 channels are required here.

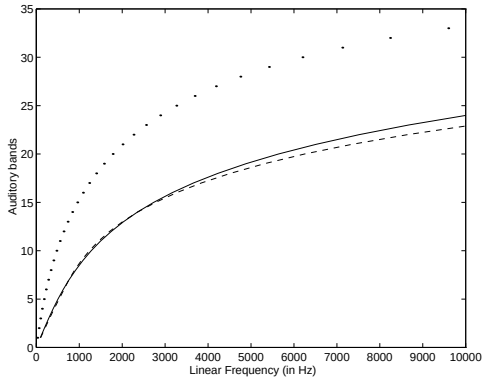

Figure 2.5: An example of auditory frequency Scales. The solid line is for *Mel*-scale and the dashed line is for *Bark* scale

Fourier Transform Filter Bank

We have previously discussed the advantages of using non-uniformly spaced frequency samples. One of the easiest and more efficient ways to compute a non-uniformly spaced filter bank model of the signal is to simply perform a Fourier transform on the signal, and sample the transform output at the desired frequencies.

We compute the auditory spectrum with our FFT (Fast Fourier Transform) (Oppenheim [62], Van Compernolle [87]). The auditory filter bank is implemented by the following:

$$S_l = 10 \log_{10} \left(\sum_{k=k_{min}(l)}^{k_{max}(l)} \Phi_{xx}[k] \right) \tag{2.13}$$

$$k_{min}(l) = \frac{f_l + f_{l-1}}{2 * F_s} \tag{2.14}$$

$$k_{max}(l) = \frac{f_l + f_{l+1}}{2 * F_s} \tag{2.15}$$

Where S_l is the output spectrum for the lth auditory spectral band, $\Phi_{xx}[k]$ is the FFT power spectrum of the kth FFT frequency band, f_l is the frequency for the lth auditory band and F_s is the sampling frequency.

LPC Analysis

Linear Predictive Coding (LPC) is a very important spectral estimation technique because it can provide an estimate of the poles (hence the formants) of the vocal tract transfer function. The LPC algorithm is a P^{th} order linear predictor which attempts to predict the value of any point in a time-various linear system based on the values of the previous P samples [48]. The *all-pole* representation of the vocal tract transfer function, $H(z)$, can be represented by the following equation:

$$H(z) = \frac{G}{A(z)} = \frac{G}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_P z^{-P}} \tag{2.16}$$

The values $a(i)$ are called the prediction coefficients while G represents the amplitude or gain associated with the vocal tract excitation. The poles of the transfer function in Equation 2.16 are determined by the roots of the polynomial in the denominator. Because the LPC model is an *all-pole model*, it can capture the resonant frequencies, or formants, but not the zeros, which are important for nasalized sounds. In addition, LPC does not adequately estimate signals which have no poles, such as some unvoiced speech and noise.

For the speech signal $s(n)$ produced by a linear system, the predicted speech sample $\hat{s}(n)$ is a function of $a(i)$ and prior speech samples according to:

$$\hat{s}(n) = \sum_{i=1}^{P} a(j)s(n-i) \tag{2.17}$$

LPC analysis involves solving for the $a(i)$ terms according to a least error criterion. If the error is defined as:

$$e(n) = s(n) - \hat{s}(n) \tag{2.18}$$

$$= s(n) - \sum_{i=1}^{P} a(i)s(n-i) \tag{2.19}$$

then taking the derivative of the square error with respect to the coefficients $a(j)$ and setting it equal to zero gives:

$$\frac{\partial}{\partial a(j)}(s(n) - \sum_{i=1}^{P} a(i)s(n-i))^2 = 0 \tag{2.20}$$

Thus,

$$s(n)s(n-j) = \sum_{i=1}^{P} a(i)s(n-i)s(n-j) \quad for j = 1, \ldots, P \tag{2.21}$$

There are two principal methods for solving Equation 2.21 for the prediction coefficients $a(i)$. The first is an autocorrelation method, which multiplies the speech signal by a Hamming window or similar time window, assuming that the speech signal is stationary within, and zero outside, the analysis window. The autocorrelation solution to Equation 2.21 can be expressed as

$$R(j) = \sum_{i=1}^{P} a(i)R(|i-j|)) \quad j = 1, \ldots, P \tag{2.22}$$

where $R(j)$ is an even function ($R(j) = R(-j)$) and is computed from:

$$R(j) = \frac{1}{\gamma} \sum_{m=0}^{N-1-j} s(m)s(m+j) \quad j = 1, \ldots, P \tag{2.23}$$

Where γ is a normalization factor as we mentioned before. Once the autocorrelation terms $R(j)$ have been calculated, a recursive algorithm, named Levinson-Durbin Algorithm [48], is used to determine the values of $a(i)$.

Given that the vocal tract does not produce a *purely* linear speech signal, the solution for $a(j)$ is optimal, but not exact. In fact, the most difficult part of the speech signal to predict is the glottal pulse because it contains a large amount of energy which *instantaneously* appears in the signal. In addition, a nonlinearity is introduced into $R(j)$ due to the discontinuities imposed by the use of a window function which forces the values outside the analysis window to zero. An alternative method for determining the LPC coefficients, called the covariance method, is a direct Cholesky decomposition solution of the following equation

$$R(j) = a(i)R(|i-j|) \tag{2.24}$$

This equation can be expressed in matrix form. Unlike the autocorrelation method, it does not use a window to force the samples outside the analysis interval to zero. Thus, the limits on the computation of $R(j)$ extend from $-P \leq n \leq N-1-P$. A more detailed discussion can be found in [65] and [48]. The principal disadvantage with this approach is that if a nonlinearity

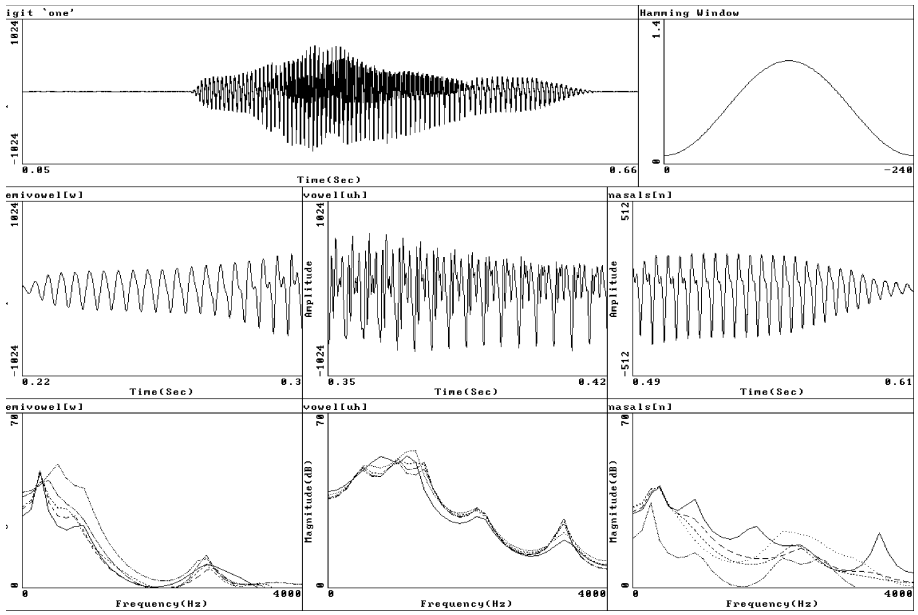

Figure 2.6: (a) Time-domain signal for digit *one*. (b) A hamming window. (c) (d) (e) Phonetic segments of speech corresponding to phonemes */w/*, */uh/* and */n/*. (f) (g) (h) LPC spectra up to the Nyquist frequency corresponding to some segments in each phonetic segment.

such as a glottal pulse occurs between samples $s(n-P)$ and $s(n)$, then the matrix array may not be invertible and a suitable solution for $a(i)$ will not be obtained. Figure 2.6 shows the speech feature analysis step by step. The speech signal is digitized at 8 kHz, framed by a Hamming window, then processed by LPC or FFT.

Cepstral Analysis

Almost all speech recognizers use cepstral analysis techniques because it operates in a domain in which the excitation function and vocal tract filter function are separable. This indicates that the characteristics of vocal tract and excitation are well represented separately in the cepstral coefficients.

There are two types of cepstral approaches: FFT cepstrum and LPC cepstrum. In the FFT cepstral analysis, the real cepstrum $c(n)$ is defined as the inverse FFT transform of the logarithm of the speech magnitude

spectrum.

$$c(n) = FFT^{-1}(10log_{10}|\Phi_{xx}[n]|) \quad 1 \leq n \leq N \tag{2.25}$$

Where $c(n)$ is the cepstral coefficient, $\Phi_{xx}[n]$ is the FFT power coefficient. To investigate properties of the LPC cepstrum, we take a z-transform in Equation 2.16. The excitation $E(z)$ and vocal tract filter $H(z)$, and a speech spectrum $X(z)$, are linearly separated by a complex logarithm operation applied to Equation 2.19. Then

$$logX(z) = logH(z) + logE(z) \tag{2.26}$$

The LPC cepstral coefficients $c(n)$ are defined as the inverse z-transform of the above log-spectrum $logX(z)$. The cepstral coefficients, $c(n)$, of the spectra obtained from LPC analysis can be computed recursively from the LPC coefficients, $a(i)$ [65][48],

$$c(i) = -a(n) - \sum_{i=1}^{n-1} \frac{n-i}{n} a(i)c(n-i) \quad n \geq 1 \tag{2.27}$$

where $a(i) = 0$ when $i > n$ (n is the order of LPC analysis). A distinctive advantage of cepstral analysis is that correlation between coefficients is extremely small so that simplified modeling assumptions can be applied.

Other Speech Features and Analysis Methods

FFT spectra and LPC coefficients are the fundamental speech features. Other features used in speech recognition, such as cepstrum, are calculated and derived from those fundamentals. Some parameters such as energy parameters can be obtained during the speech analysis.

Derivatives The time derivatives on frames reveal the sequential trend of a speech signal. Those parameters are essential for speech recognition. Most of times the first derivatives and the second derivatives of the feature parameters have been combined with the original features. Delta-cepstra and delta-delta cepstra are calculated in the following Equations:

$$\Delta c_i(t) = \sum_{j=-k}^{j=k} j c_i(t+j) \tag{2.28}$$

$$\Delta c_i^2(t) = \Delta c_i(t+1) - \Delta c_i(t-1) \tag{2.29}$$

Spectral Smoothing For speaker independent speech recognition and in noisy environments, spectral smoothing is commonly used to reduce the speaker dependent information and noise. Several efforts have been made in this area. Perceptual Linear Prediction (PLP) (Hermansky [26]) analysis takes into account some specific differences between the classical spectral analysis and the auditory system. It focuses on a different equal-loudness matching than pre-emphasis and on a distinction between intensity and loudness. PLP analysis filters out a lot of speaker dependent details. Spectral subtraction (Van Compernolle [86]) and Cepstral-Mean (Clues [17]) have been used to get rid of additive noise and Rasta PLP is used to reduce the convolutional noise in the telephone channel (Hermansky [27]).

2.4.3 Vector Quantization

Vector Quantization(VQ) is used to compare the representation obtained in the current analysis frame to a lookup table, called a codebook. The function of the codebook is to determine the closest match in the codebook. The index to the codebook entry, rather than the initial presentation, is then used to simplify subsequent processing phases. We will give a more detailed description of vector quantization in chapter 4.

2.5 Time Alignment and Pattern Matching

The above two steps convert the speech samples into observation vectors representing events in a probability space. The next step is to perform a statistical analysis on the vectors to determine if they might be part of a spoken word or phrase or whether they are merely noise. Speech sounds such as the */a:/* sound in *march* exhibit several resonances in the spectrum that typically extend for 120 ms. Transitional sounds, such as the *b* in *boy* exist for a brief interval of approximately 20 ms. Encoding the temporal behavior of these sounds is a challenging task in statistical modeling. We address two popular techniques: Dynamic Time Warping (DTW) and Hidden Markov Modeling (HMM).

2.5.1 Dynamic Time Warping

Dynamic Time Warping (DTW) uses a distortion metric, for example, Euclidean distance, to compare feature vectors from the stored reference tem-

plate and the input utterance, and then determines the overall score of the alignment. The underlying assumption of DTW is that the global variations in speaking rate for a speaker uttering the same word on different occasions can be handled by linear time normalization.

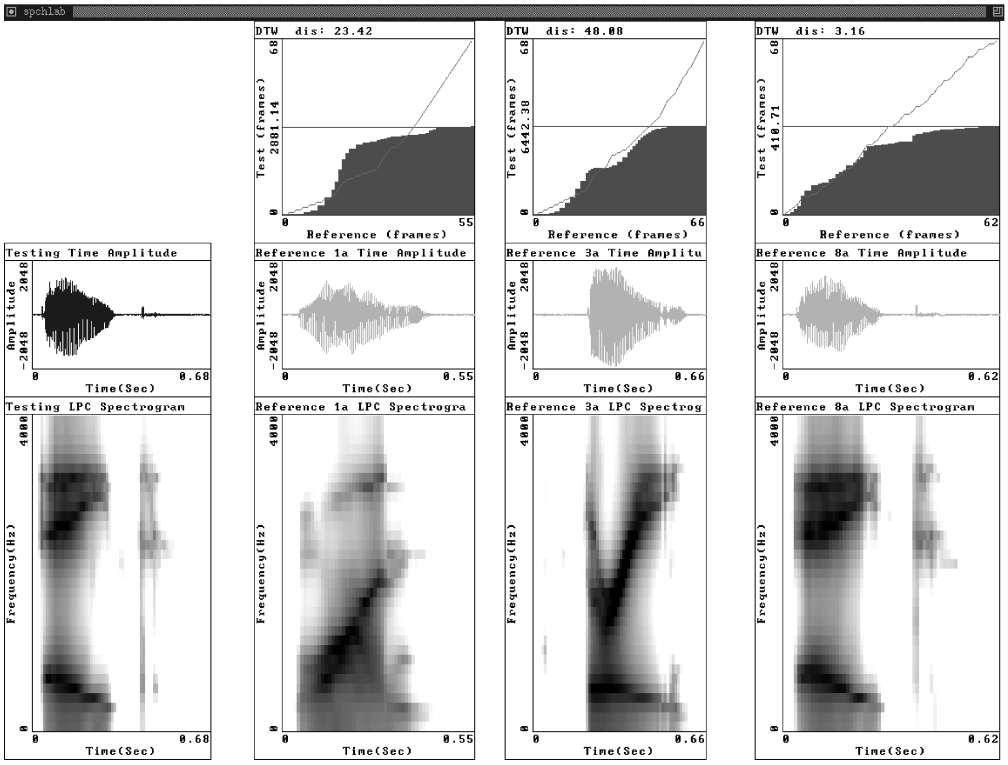

Figure 2.7: An illustration of a DTW with optimal paths, accumulated scores for a test utterance of Dutch digit *eight* and three reference templates of Dutch digits: *one*, *three* and *eight*.

Figure 2.7 is an example in which the test utterance: Dutch digit *eight* on the y-axis is compared to three templates on the x-axis: Dutch digits *one*, *three* and *eight*, from left to right. The cumulative scores are shown, along with the optimal path for each reference. The cumulative scores are calculated by cumulating the Euclidean distances between the input frames and its reference frames. Because the length of the reference template is fixed, and warping only occurs along the axis of the utterance, the average

score permits candidate words to be compared. The reference with the lowest average score indicates the utterance recognized, the one on the left in this example.

DTW begins and ends at/near the endpoints of each utterance. A begin/end point detection is crucial for the DTW performance. For connected or continuous speech recognition systems, a new path is started at each begin/end-point time interval.

Although the DTW algorithm adequately accounts for warping in the time domain, it has a number of defects. One of them is heavy computational load. Multiple word templates are commonly used in DTW systems. The templates are usually selected using a clustering algorithm, such as VQ. The computational complexity of a DTW solution may be acceptable in speaker-dependent ASR, but may not be practical for speaker-independent systems. That is especially true for large-vocabulary systems.

2.5.2 Hidden Markov Models

The dominant technique today for modeling the time course of a speech signal is Hidden Markov Modeling. The HMM was introduced in a landmark paper by Baum [6], where the model was proposed as a statistical method for estimating the probabilistic function of a *Markov* chain. Essentially, HMMs are a method for modeling a system with discrete, time dependent behavior characterized by common, short-time processes and transitions between them. An HMM can be considered as a finite state machine where the transitions between the states are dependent upon the occurrence of some symbols. Associated with each state transition is an output probability distribution which describes the probability with which a symbol will occur during the transition, and a transition probability indicating the likelihood of this transition.

First-order, left to right HMMs are commonly used in ASR products. There are two assumptions behind them. The first is the Markov assumption, i.e., at each observation time t, a new state is entered based on the transition probability, which only depends on the previous state. The transition may allow the process to remain in the previous state. The second assumption is the output-independence assumption, i.e., the output probability depends only on the state at that time regardless of when and how the state is entered. These two assumptions make calculation very efficient. A straightforward left-to-right model is shown in Figure 2.8, where

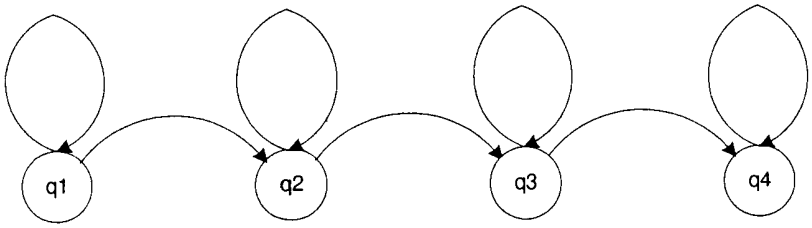

Figure 2.8: An illustration of a left-to-right hidden Markov model. This architecture is well suited to speech application because its inherent sequential structure models the temporal flow of speech.

q_i $(1 \leq i \leq 4)$ is the HMM state.

The key parameters to be determined in an HMM-based ASR system are the number of states per unit, and the state transition and observation probabilities. The unit can be a word (the word model) or a phoneme (the phoneme model). Below, we will examine the HMM word model. The results of the word model will apply directly to the phoneme model. Large amounts of training data are needed to obtain robust estimates of these probabilities so that the HMM algorithm is more powerful than the DTW algorithm, which uses a finite number of templates. An HMM-based ASR will generally have a number of HMMs. For instance 10 digit HMM word models correspond to 10 digits.

Training HMMs

The selection of the optimal number of states which properly describe the observed sequence of events for a given word is a somewhat empirical process. For discrete words, one might select a number of states which roughly correspond to the number of phonemes in the word to be modeled, with additional states corresponding to beginning an ending silences. An example structure of an HMM for Dutch digit *een* is shown in Figure 2.9. HMM training for isolated words can be implemented directly using an iterative procedure, known as the *forward* and *backward* algorithm or *Baum-Welch*

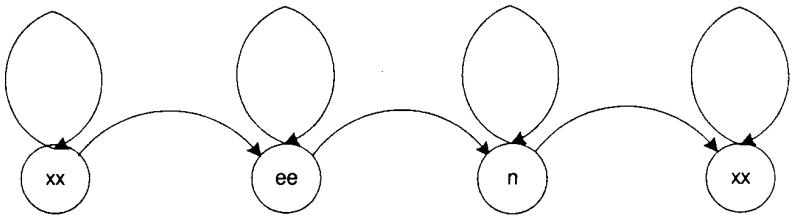

Figure 2.9: The word model for Dutch digit *een*.

algorithm (Baum[6]), which is a computationally efficient method for determining the model parameters. This iterative procedure uses the forward probability and backward probability to update the observation probability and the transition probability. An alternative is to use the *Viterbi* algorithm, which offers a recursive optimal solution to estimate the state sequence. The *Viterbi* alignment is essentially a dynamic programming procedure, like the one used in the DTW algorithm except that the probability between the test and reference model is computed in the HMM rather than the distance measure between speech frames in the DTW system. The detailed implementation is described in Van Compernolle [86]. Training can accomplish speaker adaptation for existing models and cross-validation for discriminative training.

Testing HMMs

During recognition, the input symbols generate a particular sequence of states which are visited by the HMM in producing the observation sequence. The state sequence essentially represents the segmentation of the word modeled by the HMM. However, to ensure that the optimal state sequence with the highest *a posteriori* probability is selected, the *Viterbi* algorithm is employed.

A trellis is used to demonstrate the two dimensional representation between states observations. From the trellis we can observe probability computations in HMMs efficiently. Figure 2.10 (b) illustrates the computation

of Viterbi probability calculation through a trellis using the HMM in Figure 2.10 (a) and the observation sequence AAB. To simplify our calculation, we assume the transition probabilities are equal, i.e. they are all equal to 1/2. We can omit these terms. Each cell indicates the cumulative probability at a particular state and time. The computation begins by assigning 1.0 to the initial state and 0.0 to all other states at time $t = 0$. The cells are computed time-synchronously from left to right at $t > 0$. Each column of states for time t is completely computed before going to time $t+1$, the next column. The last cell in the final column contains the probability of generating the observation sequence. The solid arrowed lines indicate Viterbi calculation paths and the dash arrowed lines backtrack the Viterbi path.

2.6 Natural Language Processing

The final stage of the recognition process consists of a Natural Language Processing(NLP) module which attempts to resolve the possible word selections using language specific *constraints* or *knowledge*. In ASR applications the input to the NLP is often an N-Best list of potential words to be evaluated. For small vocabulary recognizers, a set of phonetically dissimilar words can be selected to reduce the need for a language processing module. However, large vocabulary systems have many phonetically close words e.g., *vat* and *fat*, which become more acoustically similar when spoken in context: *his vat*, *the fat*. In this example, coarticulation of the unvoiced */s/* influences the */v/* to become unvoiced, and similarly, the voiced */e/* influences the unvoiced */f/* to become voiced. Language processing is a crucial element in any text generating system with a large vocabulary. Lexical knowledge (i.e. vocabulary definition) is required as is the syntax and semantics of the language. Syntactic models attempt to capture the structure of the language by using language construct rules and statistical observations of word occurrences within the language. Semantic models attempt to represent the meaning or understand the relationships which the language encodes. Semantic analysis gives answers to the questions such as who, what, where, and when, etc. Many Internet search engines utilize semantic analysis to understand what the user is searching. In current speech recognition, semantic information is often embedded within the structure of a syntactic model. The primary objectives of syntax are to

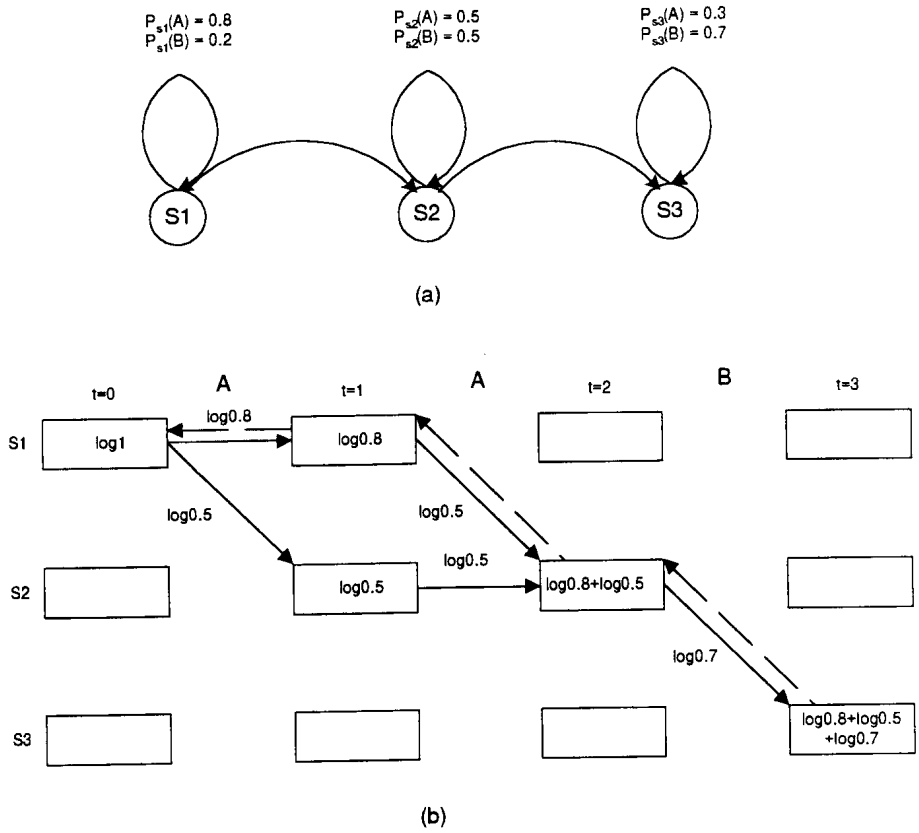

Figure 2.10: (a) An HMM with two states and two output symbols, A and B. (b) The Viterbi computation using the HMM.

- Reduce perplexity and increase speed and accuracy. The recognition process involves searching the application vocabulary to locate the best match for the input utterance. The term perplexity is the total number of searching branches in the entire application. The recognizer does not have to search the entire vocabulary for each input utterance, but only a subset of it. An application consisting only of *yes* and *no*, for instance, has a perplexity of two. As the recognizer has fewer choices, the recognition accuracy is improved and the searching time is automatically reduced.

- Enhance vocabulary flexibility. The careful design of active vocabularies enhances accuracy when the words within each active vocabulary is acoustically distinct. Confusable words, such as *mine* and *nine*, and homo-phones such as *to*, *too*, and *two* can still be included in a single application, but the recognition errors they often produce are minimized by placing them in different active vocabulary sets.

Finite-State Grammars

A finite-state grammar reduces perplexity by delineating the words that are allowable at any point in the input. Some automatic speech recognizers are constrained by a finite-state grammar, in which the vocabulary is divided into subsets. The number of well-formed sentences in a human language is theoretically infinite. But the rules of sentence construction are finite. Finite-states grammars are generally represented using finite state networks. Like hidden Markov models, finite-state grammars consist of states linked by left-to-right, directional transitions, and recursive transitions as illustrated in Figure 2.11. Unlike hidden Markov models, finite-state grammars are not characterized by probabilities because of their deterministic nature. Generally, a grammar is defined and coded by the application developer. Some graphical application interfaces such as the Unisys Speech Assistant Tool offer a solution for developers [76].

Statistical Models

Instead of using the syntactic rules of the language, statistical models, such as bigrams and trigrams, use probabilities estimated over a large training sample of the language. For instance, a word-pair bigram for word transitions is expressed as

$$P(transition) = P(word_2|word_1) \tag{2.30}$$

where the transition occurs from $word_2$ to $word_1$. Given that bigrams are dependent on the previous word, trigrams are dependent on the previous two words, and N-grams on the previous N-1 words. The drawback with N-grams is that they must be trained with a large, statistically significant data set.

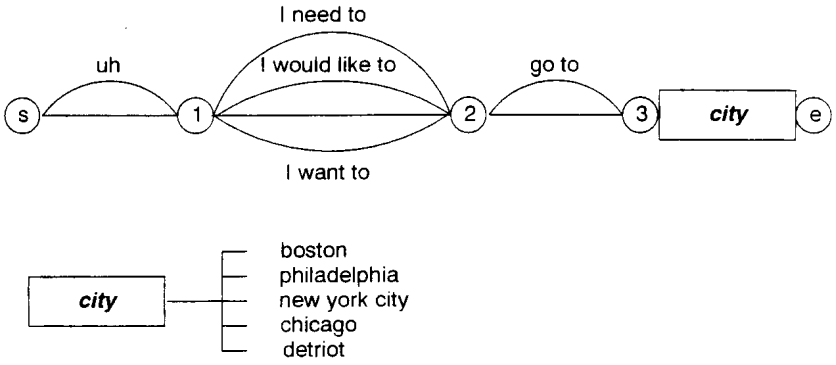

Figure 2.11: A finite-state grammar

Context-Free Grammar

Like finite-state grammars, context-free grammars are deterministic: they define allowable structures. Part of the flexibility of context-free grammars is that they can represent a variety of linguistic approaches [96]. The application of context-free grammar rules is commonly represented by a parser, as shown in Figure 2.12. For real-time recognition which takes place as the talker speaks, ASR parsing is performed bottom-up, in a left-to-right manner. Therefore, the potential words in the parser are combined in a hierarchy to form a phrase or sentence.

Of these three models, statistical N-grams are most often employed in ASR system. However, context-free grammar parsers and finite-state grammars are attractive because they can be constructed without the burden of gathering training data to present the relationships between different parts of speech in the language. Although parsers and grammars model sentence syntax and scale up well to large-vocabulary recognition tasks, they may be too rigid for conversational speech. On the other hand, statistical models are more suitable for conversational speech because they are not so restrictive as to permit only valid language constructs.

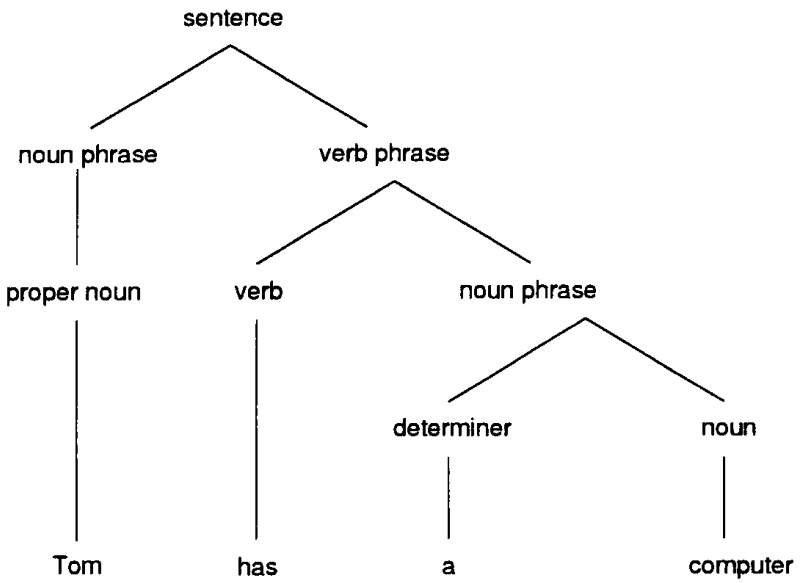

Figure 2.12: A parse tree for the sentence "Tom has a computer"

2.7 Summary

In this chapter we have addressed the fundamentals of speech processing and speech recognition. The general speech recognition model has been introduced and each of its components has been discussed. These components together serve as the framework for our future discussion. We have presented the algorithms which are used in our system implementation. We have focused on LPC and auditory analysis in signal processing and feature extraction. We have discussed DTW and HMM in pattern matching and time alignment. Additional discussion of vector quantization and HMM will be provided in chapter 4, where we integrate neural networks into the speech recognition system.

Chapter 3

Multi-Layer Perceptron for Speech Recognition

3.1 Introduction

In this chapter, we introduce the mathematical model of the Multi-Layer Perceptron(MLP). We give a detailed description about MLP neural networks from network structure to network training. We present the applications using MLPs: MLP phonetic frame recognition and Time Delay Neural Network (TDNN) phoneme recognition. We give a fuzzy interpretation to the MLP neural network based on the fuzzy set theory with which we analyze the MLP training process and its output.

3.1.1 Multilayer Perception

In connectionist models, knowledge or constraints are not encoded in individual units, rules or procedures, but distributed across many simple computing units. Uncertainty is modeled not as likelihoods or probability density functions of a single unit, but by the pattern of activities in many units. Therefore, knowledge is not programmed into any individual unit's function; rather, it lies in the connections and interactions between linked processing elements. The Multi-Layer Perceptron is one of the most popular connectionist models.

The Multi-Layer Perceptron architecture is a hierarchical design consisting of fully interconnected layers of computing units. It belongs to the

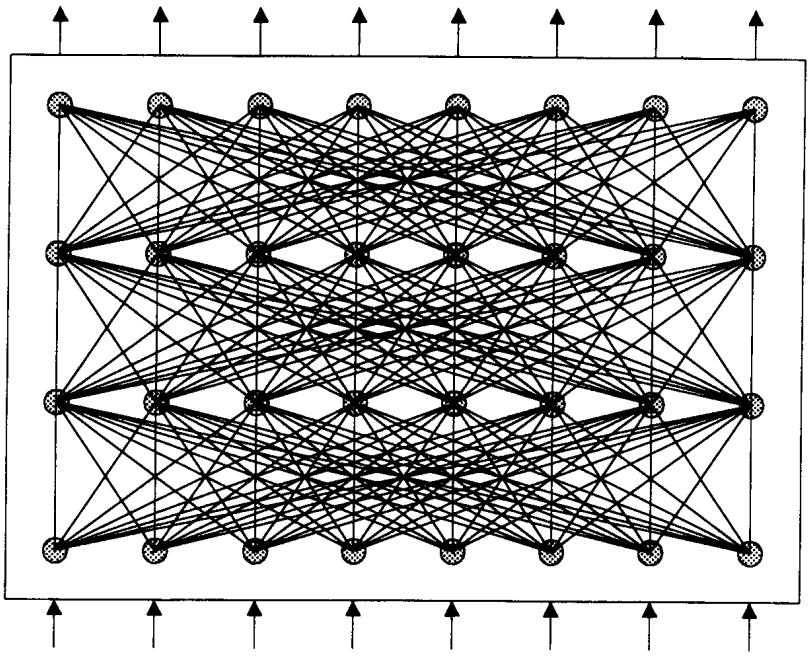

Figure 3.1: The structure of a Multi-Layer Perceptron

class of mapping neural network architectures. The structure is shown in Figure 3.1. A Multi-Layer Perceptron consists of an input layer, an output layer and at least one hidden layer which contains hidden units that are not directly connected to both the input and output units.

The Multi-Layer Perceptron overcomes limitations of single layer perceptrons, but were not used in the past because effective training algorithms were not available. This has changed with the development of a training algorithm – the Back Propagation algorithm which will be discussed below. Although it cannot be proven that this algorithm converges as a single layer perceptron (using LMS algorithm), it has been shown to be successful for many speech and image processing problems.

Considering its simple architecture, MLP offers a remarkably wide range of computational functions. Depending on the weights, the bias, and the input, a unit in a network can act as a simple linear summer, a boolean op-

erator, or a non-linear analog processing element. The processing capacities of multi-layer perceptrons stem from the non-linearity used within units. If units were linear elements, then a single layer network with appropriately chosen weights could exactly duplicate these calculations performed by multi-layer networks.

Lippman [46] has demonstrated that networks with two hidden layers are capable of performing any deterministic transformation requiring binary outputs and analog inputs. Hornik et al [28] have rigorously established that MLPs with two hidden layers are capable of approximating any measurable functions from one Euclidean space to another, to any desired degree of accuracy, provided sufficient hidden units are available. In this sense, the MLP is a class of universal approximators. Multi-Layer Perceptron classifiers have been applied to speech problems more often than any other neural network classifiers (Waibel et al [90], Bourlard et al [10], Ma and Van Compernolle [54], Robinson et al [71]).

3.2 Back Propagation Algorithm

The Back Propagation (BP) algorithm is powerful enough to construct appropriate internal representations. Without question, the Back Propagation algorithm is currently the most widely applied MLP training algorithm. This popularity primarily revolves around the ability of back propagation networks to learn complicated multi-dimensional mapping. The basic algorithm was first described by Werbos in his Ph.D. thesis in 1974 [91] and rediscovered and applied to neural computation by Rumelhart, Hinton and Williams in 1986 [73].

The Back Propagation algorithm, like other algorithms in neural network research, is a supervised learning procedure which involves the representation of a set of pairs of input and output patterns. The system first uses the input vector to produce its own output vector and then compares this with the desired output or the target vector. If there is no difference, then no learning takes place. Otherwise the weights are changed to reduce the difference. If the input units are directly connected to the output units, it is relatively easy to find learning rules that iteratively adjust the relative strengths of the connections, so as to progressively reduce the difference between the actual and desired output vector. The learning becomes more interesting but more difficult when we introduce hidden units whose ac-

tual or desired states are not specified by the task (input and output units can be). The learning procedure must decide under what circumstances the hidden units should be active in order to help achieving the desired input-output behavior.

The Back Propagation algorithm is a remarkably simple extension of Widrow and Hoff's delta rule [93], and is also called the "generalized delta rule". It uses a gradient search technique to minimize a cost function equal to the mean square difference between the desired and the actual network output. It requires a continuous differentiable non-linearity in the computational element. It consists of two passes of computation: the *feed forward pass* from bottom to top and the *feed backward pass* from top to bottom.

3.2.1 Feed Forward Process

In the forward pass, the synaptic weights remain unchanged throughout the network, and the function signals of the network are computed on a neuron-by-neuron basis. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying Equation 3.1 to Equation 3.3 to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined. A diagram of a unit is shown in Figure 3.2.

For the input pattern p, the total input z_{pj} to unit j is a linear function of the input x_{pi} of the units that are connected to j, the bias b_j, and the weights w_{ij}, i.e.

$$z_{pj} = \sum_i w_{ij} x_{pi} - b_j \tag{3.1}$$

The biases can be given by introducing an extra input to each unit which always has the value of 1. The weight on this extra input is equivalent to a threshold of the opposite sign. It can be treated just like the other weights, and the above formula becomes

$$z_{pj} = \sum_i w_{ij} x_{pi} \tag{3.2}$$

A unit has a real-valued output y_{pj}, which is a non-linear function of its total input

$$y_{pj} = F(z_{pj}) \tag{3.3}$$

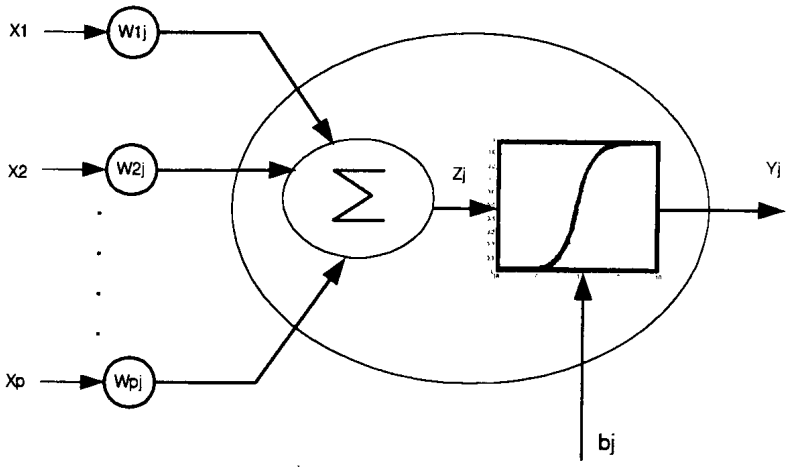

Figure 3.2: The block diagram of a unit

where F is a semi-linear function which is differentiable and non-decreasing. Here the activation function is the sigmoid function which is a continuous and nonlinear function shown in Figure 3.3.

$$F(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

The derivative of $F(x)$ is

$$F'(x) = F(x)(1 - F(x)) \tag{3.5}$$

It is not necessary to use exactly the function given in Equation 3.2, but the use of a linear function for combining the input to a unit before applying the non-linearity greatly simplifies the learning procedure.

The aim is to find a set of weights that ensures that for each input vector the output vector produced by the network is the same as (or sufficiently close to) the desired output vector. If there is a fixed, finite set of input-output cases, the total error in the performance of the network with a particular set of weights can be computed by comparing the actual and

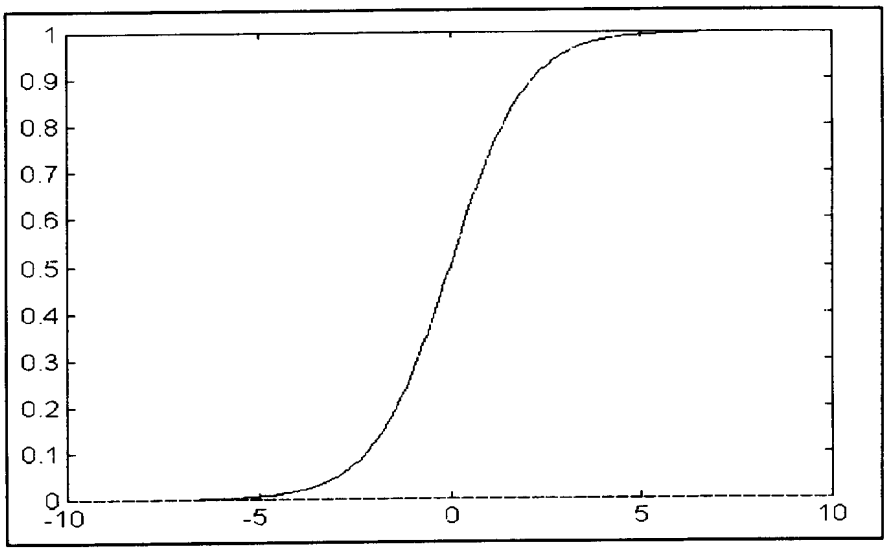

Figure 3.3: The sigmoid function

desired output vectors for every case. Let

$$E_p = \frac{1}{2}\sum_i (d_{pj} - y_{pj})^2 \tag{3.6}$$

be our measure of the error E_p on input/output pattern p where j is an index over the output units, y_{pj} and d_{pj} are the actual output value of an output unit and its desired output value respectively. Then

$$E = \sum_p E_p \tag{3.7}$$

where p is an index over all the cases (input-output pairs) and E is our overall measure of the error. To minimize E by gradient descent, it is necessary to compute the partial derivatives of E with respect to each weight in the network.

3.2.2 Standard Delta Learning Rule

The learning procedure involves the presentation of a set of pairs of input and output patterns. The system first uses the input vector to produce

its own output vector and then compares this with the desired output or the target vector. If there is no difference, no learning takes place. Otherwise, the weights are changed to reduce the difference. In the case without hidden units, the standard delta rule for changing weights following the presentation of each input/output pattern pair p is given by

$$\Delta_p w_{ij} = \eta(d_{pj} - y_{pj})x_{pi} \tag{3.8}$$

$$\Delta_p w_{ij} = \eta\delta_{pj}x_{pi} \tag{3.9}$$

where d_{pj} is the target output value for the jth component of the output pattern, y_{pj} is the jth element of the actual output value produced by the presentation of the input pattern p, x_{pi} is the value of the ith element of the input pattern, δ_{pj} is the error of jth element between the desired output and the actual output, $\Delta_p w_{ij}$ is the change to be made to the weight from the ith to the jth unit following presentation of pattern p, and η is the learning rate.

The standard delta rule implements a gradient descent in E when all units are linear. There are many ways of deriving this rule. For our purpose here, it is useful to see that for linear units it minimizes the squares of the differences between the actual and desired output values summed over the output units and all pairs of input/output vectors. One way to observe this is to show that the derivative of the error measure with respect to each weight is proportional to the weight change dictated by the delta rule, with a negative constant of proportionality.

$$\Delta_p w_{ij} = -\eta\frac{\partial E_p}{\partial w_{ij}} \tag{3.10}$$

This corresponds to performing a steepest descent on a surface in weight space. From this standard delta rule, the *generalized delta rule* can be derived.

3.2.3 Generalized Delta Rule and Backward Process

We apply the standard delta rule to the non-linear case. The problem of how to calculate the change of weights is equivalent to the problem of how to calculate the derivative $\partial E_p/\partial w_{ij}$. This is simply the sum of the partial derivatives for each of the input-output cases. For a given case, the partial

derivatives of the error with respect to each weight are computed in two passes. We have already described the forward pass in which the units in each layer have their output values determined by the inputs received from units in the lower layers using Equations 3.1 and 3.3. The backward pass which propagates derivatives from the top layer back to the bottom one is more complicated.

We use the chain rule to write the derivative as the product of two parts: the derivative of the error with respect to the output of the unit times the derivative of the output with respect to the weight:

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial z_{pj}} \frac{\partial z_{pj}}{\partial w_{ij}} \tag{3.11}$$

By Equation 3.2, we see that the second factor is

$$\frac{\partial z_{pj}}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_k w_{kj} x_{pk} = x_{pi} \tag{3.12}$$

Comparing with Equation 3.9, we define

$$\delta_{pj} = -\frac{\partial E_p}{\partial z_{pj}} \tag{3.13}$$

Equation 3.12 thus has the equivalent form

$$\frac{\partial E_p}{\partial w_{ij}} = -\delta_{pj} x_{pi} \tag{3.14}$$

This says that to implement gradient descent in E, we should make our weights change according to

$$\Delta_p w_{ij} = \eta \delta_{pj} x_{pi} \tag{3.15}$$

just as in the standard delta rule. The trick is to figure out what δ_{pj} should be for each unit j in the network. There is a simple recursive computation of these δ 's which can be implemented by propagating error signals backward through the network.

To compute Equation 3.13, we apply the chain rule to write this partial derivative as the product of two factors, one factor reflecting the change in error as a function of changes in the unit output and one reflecting the change in the output as a function of changes in the input. Thus, we have

$$\delta_{pj} = -\frac{\partial E_p}{\partial z_{pj}} = -\frac{\partial E_p}{\partial y_{pj}} \frac{\partial y_{pj}}{\partial z_{pj}} \tag{3.16}$$

Let us compute the second factor. By Equation 3.3 we observe that

$$\frac{\partial y_{pj}}{\partial z_{pj}} = F'(z_{pj}) \tag{3.17}$$

which is simply the derivative of the squashing function F, evaluated at the network input z_{pj} for the jth unit. To compute the first factor, we consider two cases. First, assume that unit j is a unit in the network output layer. In this case, it follows from the definition of the error function E_p that

$$\frac{\partial E_p}{\partial y_{pj}} = -(d_{pj} - y_{pj}) \tag{3.18}$$

which is the same result as we obtained with the standard delta rule. Substituting for the two factors in Equation 3.16, we get

$$\delta_{pj} = (d_{pj} - y_{pj})F'(z_{pj}) \tag{3.19}$$

for any unit j in the output layer. If unit j is not a unit in the output layer, we use the chain rule to write

$$\begin{aligned} \frac{\partial E_p}{\partial y_{pj}} &= \sum_k \frac{\partial E_p}{\partial z_{pk}} \frac{\partial z_{pk}}{\partial y_{pj}} \\ &= \sum_k \frac{\partial E_p}{\partial z_{pk}} \frac{\partial}{\partial y_{pj}} \sum_i w_{ik} y_{pi} \\ &= \sum_k \frac{\partial E_p}{\partial z_{pk}} w_{jk} \\ &= -\sum_k \delta_{pk} w_{jk} \end{aligned} \tag{3.20}$$

In this case, substituting for the two factors in Equation 3.16 yields

$$\delta_{pj} = F'(z_{pj}) \sum_k \delta_{pk} w_{jk} \tag{3.21}$$

whenever unit j is not a unit in the output layer. Equation 3.19 and Equation 3.21 give a recursive procedure for computing the δ's for all units in the networks, which are used then to compute the weight changes in the network according to Equation 3.15.

3.2.4 Back Propagation Algorithm

To summarize the above discussion, the six steps of the Back Propagation learning procedure are given below.

1. **Initialization of weights and offsets**

   Start with a reasonable network configuration and set all synaptic weights and unit offsets to small random values that are uniformly distributed.

2. **Input and output representation**

   The input vector and the specified desired outputs are presented. In speech recognition, we use MLP as a classifier, then all desired outputs are typically set to zero except the one corresponding to the class the input comes from. In this case the desired output is one. The input are samples from a training set which can be presented cyclically or randomly.

3. **Forward pass**

   During the forward pass, the input is presented and propagated forward through the network to compute the output value y_{pj} for each unit:

   $$y_{pj} = F(\sum_i w_{ij} x_{pi}) \tag{3.22}$$

   where x_{pi} is the output of unit i in the current layer, y_{pj} is the output of unit j in the layer above the current layer, and w_{ij} is the weight from unit i to unit j. $F()$ is the sigmoid function.

   This output is then compared with the target (the desired output [0 or 1]), resulting in an error signal δ_{pj} for each output unit. This is simply the difference between the actual and desired output values times the derivative of the sigmoid function.

   $$\delta_{pj} = y_{pj}(1 - y_{pj})(d_{pj} - y_{pj}) \tag{3.23}$$

4. **Backward pass**

   During the backward pass, the error signal is passed from the output layer to each unit of the input layer and the hidden layer:

   $$\delta_{pj} = x_{pj}(1 - x_{pj}) \sum_k \delta_{pk} w_{jk} \tag{3.24}$$

where j is the unit in the current layer and k is the unit in the layer above the current layer. This propagates errors back one layer each time; the same process can be repeated for every internal layer.

5. **Adaptation of weights and offsets**

   After δ is calculated, we can compute weight changes for all the connections that feed into the final layer:

$$\Delta w_{ij} = \eta \delta_{pj} x_{pi} \tag{3.25}$$

   where η is the learning rate $0 < \eta < 1$.

6. **Repetition by going to step 2**

   The learning procedure is repeated again and again until a stop criterion is satisfied. The stop criterion can be a preset threshold of the average error in the output layer or some other strategies we will discuss in section 3.3.

3.3 BP Implementation

So far we have obtained the theoretical foundation of the Back Propagation algorithm. As in the HMM, the same HMM theory has been utilized by almost all speech industries; however, the speech recognition performance is still different from one to another. When we have a sound algorithm, the most important thing is how to implement it and how to tune its parameters to maximize its performance. However the algorithm itself does not tell us how to do so at all. Here we focus on the practical implementation issues.

3.3.1 Network Initialization

Without any prior knowledge, the connection weights of the network are often randomly initialized according to the minimum entropy criterion (Rumelhart et al [73] [74] and Thimm et al [80]). Since the transition region of the sigmoid function is relatively narrow while the saturation regions are relatively wide, randomly initializing with very small values of random numbers and a uniform distribution between $[-0.3, +0.3]$ will decrease the possibility that the basic units operate in the saturation regions of the sigmoid function.

3.3.2 Momentum Term and Learning Rate

Our learning procedure requires only that the change in weight be proportional to the derivative $\partial E_p/\partial w_{ij}$. True gradient descent requires that infinitesimal steps be taken. The constant of proportionality η is the learning rate in our procedure. The larger this constant, the larger the changes in the weights. This easily leads to oscillation but offers the most rapid learning. One way to increase the learning rate without leading to oscillation is to modify the generalized delta learning rule to include a "momentum" term. This can be accomplished by the following equation:

$$\Delta w_{ij}(n) = \eta \delta_{pj} x_i + \alpha \Delta w_{ij}(n-1) \tag{3.26}$$

where the index n shows the presentation number, η is the learning rate, and α is a constant which determines the effect of past weight changes on the current direction of movement in weight space. This provides a kind of momentum in weight space that effectively filters out high-frequency variations of the error-surface in the weight space. This is useful in space containing long ravines that are characterized by sharp curvature across the ravine and a gently slopping floor. The sharp curvature tends to cause divergent oscillations across the ravine. To prevent these it is necessary to take very small steps, but this causes very slow progress along the ravine. The momentum filters out the high curvature and thus allows the effective weight steps to be bigger.

3.3.3 Weight Adaptation

There are two ways to change weights. One way is to change weights after each input-output case. This has the advantage that no separated memory is required for the derivatives, but it takes a long time to converge. It is suitable for a small training database and few patterns classified. An alternative scheme is to accumulate the partial derivative of the error E with respect to the weight w_{ij} over all the input-output cases and compute the average before changing the weights. Any conflicting and correlating patterns can be identified at this stage. However, if the number of training patterns is large, this updating scheme may not be feasible. To overcome this problem, we propose to divide the training data into subsets such that the distribution of each subset is similar to the distribution of all training patterns. In this case the error energy space of each subset can be

approximated to the global error energy space. Each subset is trained in the alternative cycle. The convergence speed is compromised. The main advantage of this strategy is that the created network has more flexibility and it may give better testing results.

3.3.4 Learning Time, Cross Validation and Stop Criterion

The essence of back-propagation learning is to encode an input-output relation, represented by synaptic weights. In practice, learning time is a big problem for the Back Propagation algorithm, especially with large training databases as in speech recognition. Training the connectionist models can be very time consuming with improper network control coefficients such as initial weights, learning rate η and momentum term α. It needs several adjustments to have the network go to the converging direction.

It is difficult to say when the network is ready as the convergence goes very slow after some iterations. Normally, the network is useable when the average error energy on each unit in the output layer is less than 0.1 empirically. However, this is not enough for speech recognition in which data for training and the data for testing are always quite different. The smaller the average error energy, the better classification for the training data, however, the performance for test data may decrease. When the network is over-trained with the speech training data, the network could learn some database-dependent information from the training data. This kind of information degrades the network classification ability for test data. The network performance vs. the number of iterations is shown in Figure 3.4.

A standard tool in statistics, known as *cross validation*, provides an appealing guiding principle [34]. The available training data set is randomly partitioned into a training set and a test set. The training set is further partitioned into two subsets: one subset used for MLP training and the other used for evaluation of the performance of the trained MLP model. When the performance goes down for the evaluation set, the training should be stopped. This method is also used to decide upon the MLP structure: the number of units in each layer.

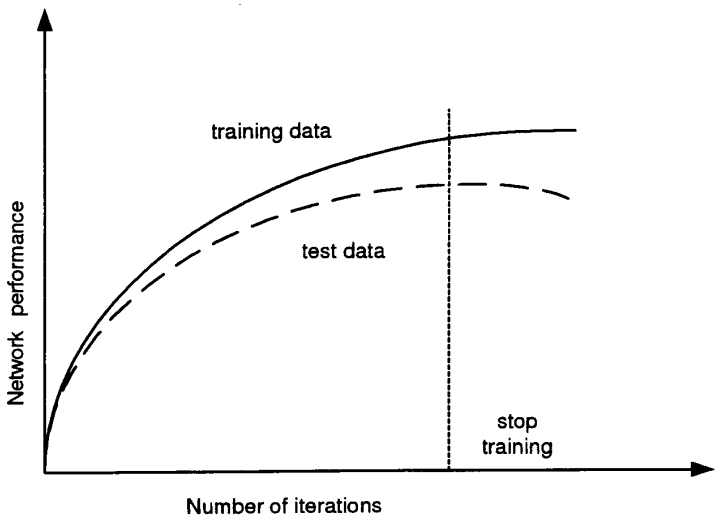

Figure 3.4: The MLP performance vs. the number of iterations

3.3.5 Modified Weight Adaptation and Weighted Training Data

In speech recognition, the training database is very large and might contain a lot of frames in which parameters have very little differences; for instance, speech parameters have little changes between two consecutive frames in the middle part of vowels (Bridle et al. [12] [11]). Here we use the cumulating weight change method to update the weights. The reason is that the same input patterns will cause the same corresponding weight changes in the neural network and the summation can be done after all input patterns forward pass through the network. It is not necessary to pass the same pattern a lot of times. The equivalent results can be obtained by multiplying the weight changes of each input pattern by a factor. The average of cumulating weight changes is given by

$$\Delta w_{ij}(n) = \frac{1}{N \sum_{p=1}^{N} \gamma_p} \sum_{p=1}^{N} \gamma_p \, \Delta w_{pij} + \alpha \Delta w_{ij}(n-1) \tag{3.27}$$

where γ_p is the weighting factor for the pattern p, N is the number of patterns. In this way, a small subset of the training database can be selected

to replace the large training database, and consequently, the training time will decrease.

Feature Emphasis

This method not only can speed up the training process but also emphasize the important input patterns. If the input pattern is important, a large weighting factor is assigned to it, otherwise a small factor is assigned. The network will be more robust to the important features. If the training patterns are contextually independent phoneme tokens, for instance, a large weighting factor γ_p should be applied to the frames in the middle part of the phoneme frame sequence and a small one to the boundary frames. The weighting function can simply be a Hamming window. Figure 3.5 illustrates the use of Hamming weighting functions on a phonetically segmented speech signal: Dutch digit `een`. Large weighting factors are assigned for nasal consonant `n`. As the vowel `ee` has more frames than the phoneme `n`, smaller values are applied to it. The noise is the least important, and the smallest values are used for those noise frames.

Training Data Balance

Moreover, the weighting method can also be used to balance the training database. In the training database, some patterns have much more samples than others. For instance, Dutch digit `een` contains phonemes `ee` and `n`. However, the vowel phoneme `ee` has more frames than the nasal phoneme `n`. We often expect the network to have the same classifying capability with respect to the different sizes of the training patterns. This can be done by assigning small weighting factors to the patterns with the large number of samples or *vice versa*, in order to avoid over-training the pattern with more input samples or under-training the pattern with fewer input samples caused by unbalanced training database. Further discussions on this issue will be presented in section 3.6.

3.4 Scaling the Network

Encouraged by the good performance and the desirable properties of the MLP model, we try to extend MLPs to large scale connectionist speech recognition systems. Some simple preliminary considerations, however,

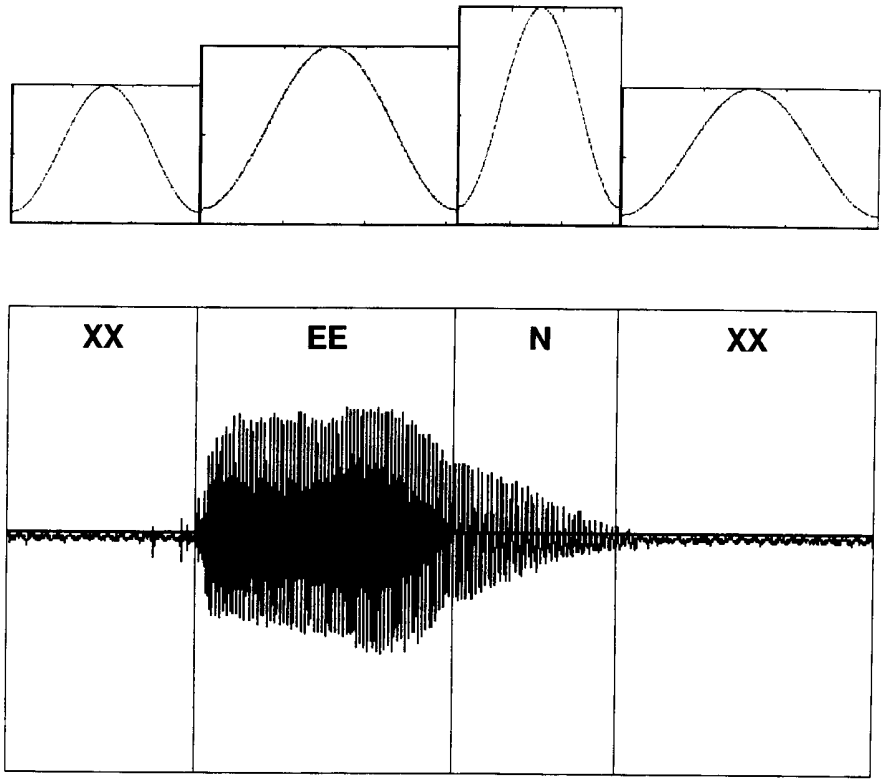

Figure 3.5: Hamming weighting functions on Dutch digit `een`

raise serious questions about the extendibility of connectionist design: Is it feasible to build and train ever larger neural networks within limited resources and time? Is it possible to add new knowledge to existing networks? With speech being one of the most complex and all-encompassing human cognitive abilities, questions of scaling must be addressed.

The structure of a scaled MLP is shown in Figure 3.6. The change of the network size only happens in the input layer. We build up a small network indicated by the dashed lines at the bottom input layer. We start training the small network based on the random initial connectionist weights. After a number of iterations, we add a frame on both sides of the input layer. Because of the contextual properties of speech, the new frames are not

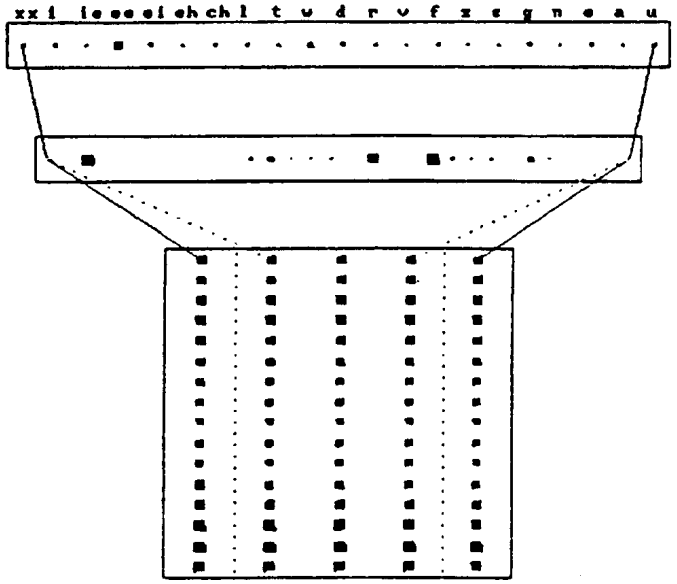

Figure 3.6: A scaled network superimposed on the original one

much different from their neighbor frames. From this we suspect that the weights of new frames might have the same values as their neighbor frames. The weights in upper layers might not be influenced by adding new frames in the input layer if we adjust the weights between the input layer and the first hidden layer in the network. For instance, this can be done by setting the weights in newly added frame i_{new} and its adjacent frame i in the enlarged network equal to half of the corresponding weight values in the frame i from the original network and keeping the weight values unchanged in the other frames of the input layer. That is

$$\hat{w}_{i_{new}j} = \hat{w}_{ij} = \frac{1}{2} w_{ij} \tag{3.28}$$

Where $\hat{w}_{i_{new}j}$ and $\hat{w}_{ij}$ are the connectionist weights in the new network, and w_{ij} is the weight in the original one.

The augmented network is trained by starting from the inherited weights from the original network. This process can be repeated several times. A large connectionist network may be established based on a small network

rather than random initial data. The training time may be saved in this way.

Our proposed strategy is based on three observations:

- Networks trained to perform a smaller task may not produce outputs that are useful for solving a more complex task, but the knowledge and internal abstractions developed in the process may indeed be valuable.
- Learning complex concepts in (developmental) stages based on previously learned knowledge is a plausible model of human learning and should be applied in connectionist systems.
- To increase competence, connectionist learning strategies should be built on existing distributed knowledge rather than trying to undo, ignore or re-learn such knowledge.

To avoid the convergence problem, the network size is increased before the network is trained to reach the global (or local) minimum. The difficulty for the network to converge increases as the size of the network grows. From what we have experienced, this method works only on a small scale (less than 10 frames in the input layer).

3.5 Time Delay Neural Networks for Phoneme Recognition

We have discussed the MLP and its BP algorithm. The next step is how to use them in speech recognition. Here we provide two basic schemes which have been investigated during this work: TDNN for phoneme recognition and MLP for frame recognition. The former will be described in this section. The latter will be addressed in the next section.

3.5.1 Structure of a TDNN

We were inspired by the benchmark of Time Delay Neural Networks for Phoneme Recognition (TDNN) done by Waibel et al [90]. A Time Delay Neural Network is an implementation of Multilayer Perceptrons. The basic unit used in many neural networks computes the weighted sum of its inputs and then passes this sum through a nonlinear function. The TDNN-unit

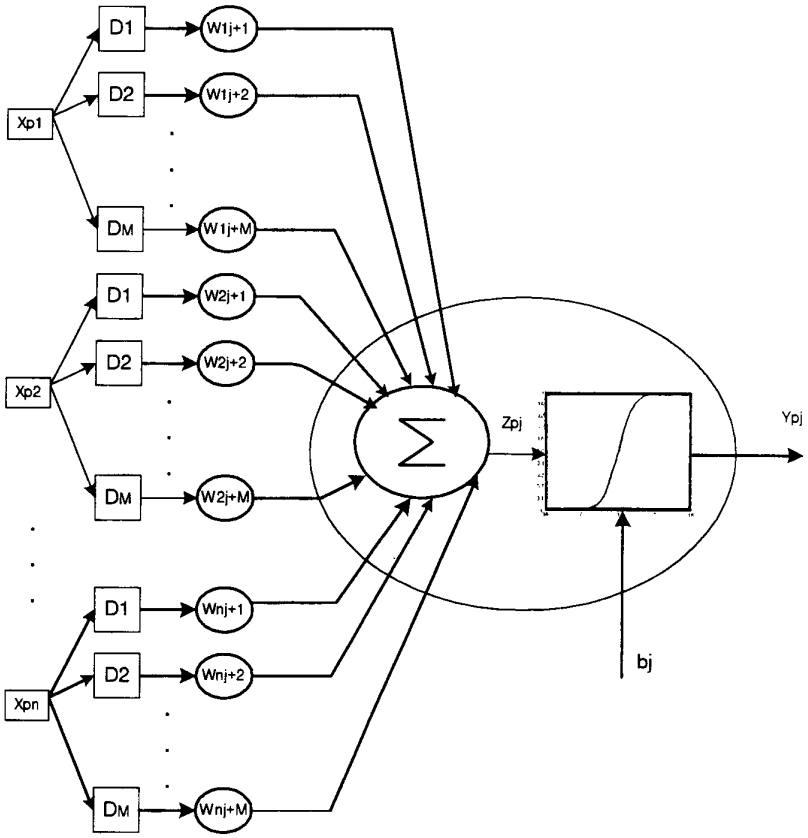

Figure 3.7: A unit in a Time Delay Neural Network

not only computes the weighted sum of their current input features, but also considers the history of these features. This is done by introducing various delays D_1 through D_N as shown in Figure 3.7 on each of the inputs and by processing (weighting) each of these delayed versions of a feature with a weight. In this fashion each unit can learn the dynamic properties of a set of moving inputs. A TDNN unit has the ability to relate and compare current input to the past history of events.

To achieve the desired learning behavior, we need to ensure that the network is exposed to sequences of patterns and that it is allowed (or encouraged) to learn about the most powerful cues and sequences of cues among them. We first apply the regular back propagation forward and

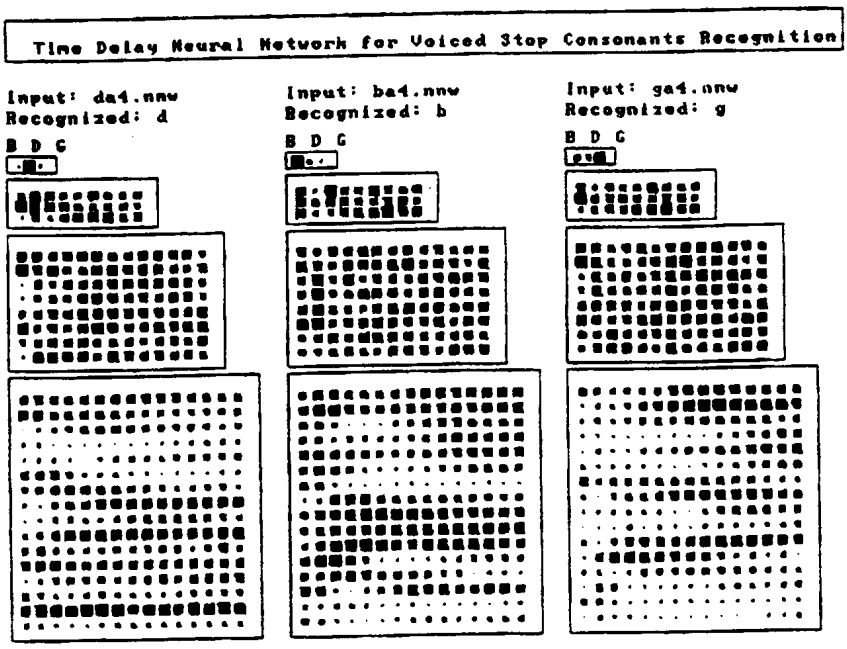

Figure 3.8: TDNN output with input CV syllables *[da, ba, ga]*

backward pass to all time-shifted copies as if they were separate events. This yields different error derivatives for corresponding (time shifted) connections. Rather than changing the weights on time-shifted connections separately, however, we actually update each weight on corresponding connections by the same value, namely by the average of all corresponding time-delayed weight changes.

Compared with the regular static MLP, the procedure described here is computationally rather expensive, due to the many iterations necessary for learning a very complex multidimensional weight space.

Figure 3.8 and Figure 3.9 show the TDNN used for voiced stop consonants *[b, d, g]* recognition. The darkness of the dots stands for the absolute output value on each unit. The output values are positive in all layers except the input layer where there are nearly half of negative values as the normalization is done to speed up the training process.

In Figure 3.8, those three voice stop consonants are recognized with the

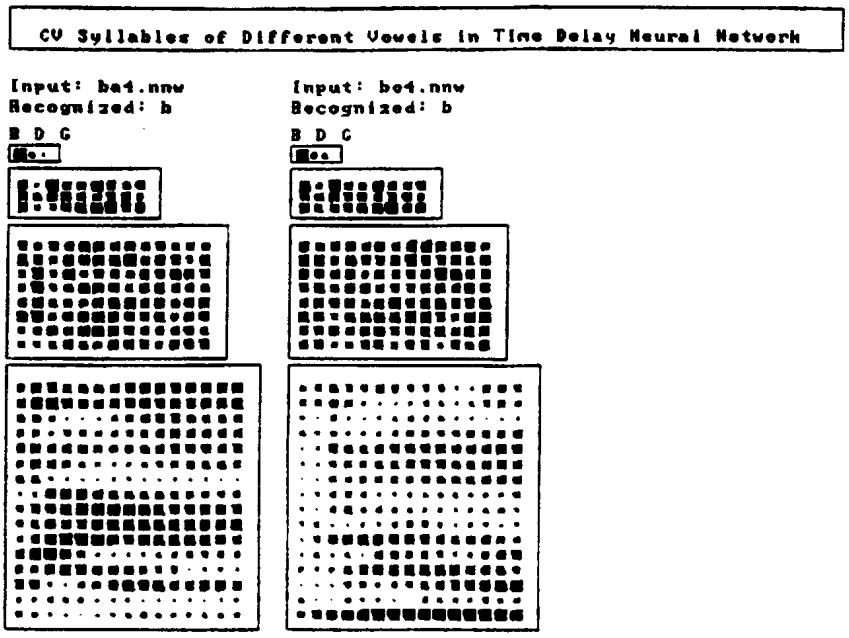

Figure 3.9: TDNN output with input CV syllables *[ba, bo]*

same vowel. Figure 3.9 shows the recognition of the same consonant with different vowels. Comparing the outputs on second hidden layers in Figure 3.8 with those in Figure 3.9, we have found that the outputs on the second hidden layers are almost the same in Figure 3.9 but quite different in Figure 3.8.

Our results are different from what has been presented by Waibel et al [90]. They claim that

> *Each TDNN unit outlined has the ability to encode temporal relationships within the range of the delays. Higher layers can attend to larger time spans, so local short duration features will be formed at the lower layers and more complex longer duration features at the higher layers.*

According to their theory, the second hidden layers should be the same

and the first layer should be different in Figure 3.8, since we have the same vowels but different consonants and *vice versa* in Figure 3.9.

From what we have observed, we have the following conclusions for the Time Delay Neural Network.

- The TDNN has time invariant ability. Each collection of TDNN units described above is duplicated for a frame shift in time. In this way, the whole history of activities is available at once.
- The higher hidden layer carries more distinctive pattern classification information than the lower hidden layer. The stable and non-discriminative information is stored in the lower hidden layer.
- Training the TDNN is very time consuming and it is difficult to converge to a global minimum for a large training database.
- Two hidden layers may be too complicated for speech recognition tasks.

3.6 MLP for Speech Frame Recognition

As we have described in Chapter 2, speech features are frame based and acoustic speech signals can be phonetically segmented. Here we simply use the MLP for frame phoneme classification.

3.6.1 MLP Structure

From the previous section about the TDNN for phoneme recognition, we have observed that two hidden layer networks are very complex, somehow redundant, and difficult to train. It is possible to use one hidden layer with more units instead of two hidden layers. Our experiments have shown that a perceptron with one hidden layer is good enough for the frame classification task. Here we use a very simple MLP structure shown in Figure 3.10.

The great advantage of the MLP is that it is able to gather all kinds of pattern classification information together to precisely present a pattern without any limitations. By taking this advantage, it is possible to use the parameters and their derivatives calculated in Chapter 2. Here we use auditory *Mel* spectra as input speech parameters features to present our method. The inputs are one or multiple adjacent frames of those speech

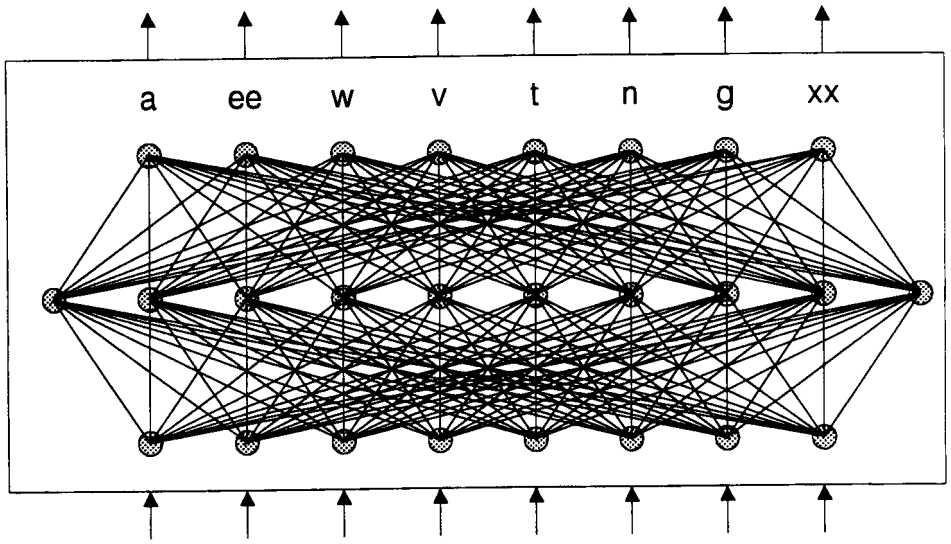

Figure 3.10: The structure of an MLP for frame recognition

parameters. We choose five frames according to our experience with the calculation of first derivatives and second derivatives, which are an effective means of describing the speech dynamics [94]. The use of context frames can increase frame recognition scores substantially.

Each unit corresponds to a phonetic label in the output layer. The target value on that unit corresponding to the current phoneme is set to one. The target values on the other units in the output layer are set to zero. Next step is how to form the training data.

3.6.2 Random Frame Selection

Let us look at the phonetic segments of the Dutch digit `een` shown in Figure 3.11. It is obvious that phoneme `een` has more frames than phoneme `n`. In our experimental Dutch digit database, the hand labeled training data contains 16,300 frames. Among them, there are only 103 frames for the voiced stop `d` and there are 1562 frames for the vowel `ee`. However, the noise

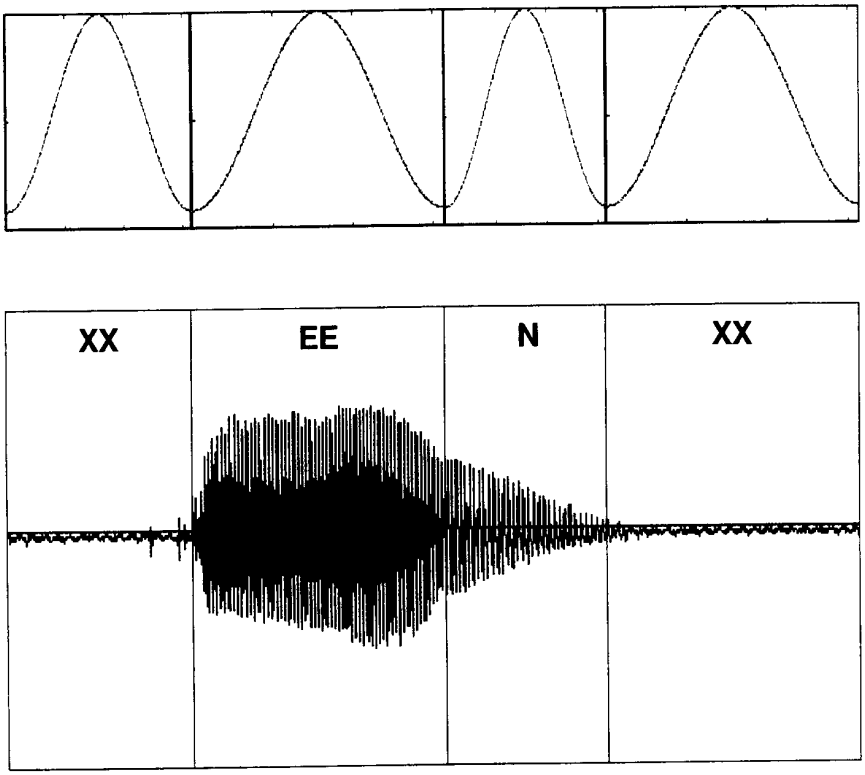

Figure 3.11: The phonetic segments of Dutch digit `een` and its weighting windows.

`xx` has 6337 frames occupying more than one third of the training database. Training based on such a database not only takes an extremely long time but results in a phonetically biased network as well. The network is more sensitive to noise and to vowels than to consonants. This is obviously unacceptable.

We propose the random frame selection strategy to solve this problem. First, we group all frames for each phoneme from the entire database. Then we randomly select the same number of frames for each phoneme from the corresponding group. Finally we obtain the training data by putting the selected data together. We do this for each iteration of the MLP training. We use the whole database and we have a small balanced training database. The result is a robust MLP produced in a short period of

training time. (It also implies that the MLP is trained with equal prior class distributions, which can be used to overcome the probability mismatching problem between MLP and HMM in using MLP as a probability estimator [8]).

Whether for Viterbi segmentation or hand segmentation, the boundary between two phonemes in a word is always difficult to determine. The frames in the middle part of each phoneme are more stable than those around the boundaries. Our purpose is to emphasize the middle frames and to de-emphasize the boundary frames. The network will be robust for the middle part of each phoneme vs. the boundary part. This is done by applying a window function, which has large values in the middle and small values near the phonetic boundaries as we have described in Section 3.3. Here we give an example how to apply weighting factors to phonetically labeled frames shown in Table 3.1.

We integrate the random frame selection with the training database weighting together. After being processed by those two methods, the received training data set is phoneme balanced, feature balanced and much smaller than the original training database. Here 4 frames for each phoneme are randomly selected from Table 3.1. The balanced and weighted training data is illustrated in Table 3.2.

The network is further trained by the above modified Back Propagation algorithm. The disadvantage of using this weighting method is that it may result in a biased NN (the prior probability changes when using the trained MLP as a probability estimator) and the summation of weights are different from phoneme to phoneme. There is a simple way to avoid this problem by using the weighting function to steer frame selection towards the middle of a phoneme, i.e., the window function provides a threshold for frame selection and the unmodified Back Propagation is applied for MLP training.

Although the MLP has shown great promises for static frame recognition, it has not reached the level for the whole word recognition as the MLP itself lacks some of the most important HMM features such as dynamic time warping. Our main task in the next chapter is using the MLP non-linear mapping and multiple feature input capabilities to develop MLP/HMM hybrid speech recognition systems.

labels — weights
xx ——- 0.2
xx ——- 0.4
xx ——- 0.8
xx ——- 0.5
xx ——- 0.2
ee ——- 0.2
ee ——- 0.3
ee ——- 0.5
ee ——- 0.7
ee ——- 0.8
ee ——- 0.9
ee ——- 0.8
ee ——- 0.7
ee ——- 0.5
ee ——- 0.2
ee ——- 0.2
n ——- 0.2
n ——- 0.7
n ——- 0.9
n ——- 0.7
n ——- 0.2
xx ——- 0.3
xx ——- 0.4
xx ——- 0.8
xx ——- 0.4
xx ——- 0.2

Table 3.1: An example of applying weighting factors to phonetically labeled frames

##########
labels — weights
————————
xx ——- 0.2
xx ——- 0.4
xx ——- 0.6
xx ——- 0.4
ee ——- 0.8
ee ——- 0.5
ee ——- 0.8
ee ——- 0.2
n ——- 0.2
n ——- 0.7
n ——- 0.9
n ——- 0.2
##########

Table 3.2: Training data generated by random frame selection

3.7 Analysis of the MLP Output

The winner-take-all labeling strategy takes into account only the highest scoring output of the MLP. However, the other outputs can also yield important information. Especially at phoneme boundaries, it often occurs that several outputs have high values.

We use statistical histograms to analyze the MLP output. Histograms for 200 utterances of the Dutch digit *een* are illustrated in Figure 3.12. The histogram provides an estimation of the probability of occurrence of the MLP output values. The names labeled on the small histogram windows correspond to the labels on the MLP output units.

If the input pattern corresponds to the label on the output unit, the MLP output on that unit typically has a large value with a high probability, for instance, the histogram of `ee` in histogram window `NNWEE`, `n` in `NNWN` and `xx` in `NNWXX`. When the input pattern is totally unrelated to the label on the output unit, the MLP output on that unit has a small value very close to zero with a high probability and its histogram looks approximately Gaussian, for instance, the histograms `xx, ee, n` in `NNWR, NNWW, NNWA` (the output values are less than 0.2 or 0.3). When the input pattern is related to the label on the output unit, the MLP output typically has a large value with a higher probability than when the input pattern is unrelated and the shape of the histogram is uncertain, for instance, the histogram `xx` in `NNWT`, `ee` in `NNWIE` and `n` in `NNWEH`.

It is clear from these observations that not only the unit with the highest output value holds important information but the other units do as well. As in the conventional K-means VQ, the distortion measure reflects the closeness between the input pattern and each codeword. The MLP output value contains the closeness measure between the input speech pattern and each output phoneme in a non-linear (probabilistic or fuzzy) fashion.

3.8 Fuzzy Multi-Layer Perceptron

Here we attempt to build a fuzzy version of the multi-layer perceptron using the gradient-descent-based back-propagation algorithm as we have described in the previous sections, by incorporating concepts of the fuzzy set theory at various stages.

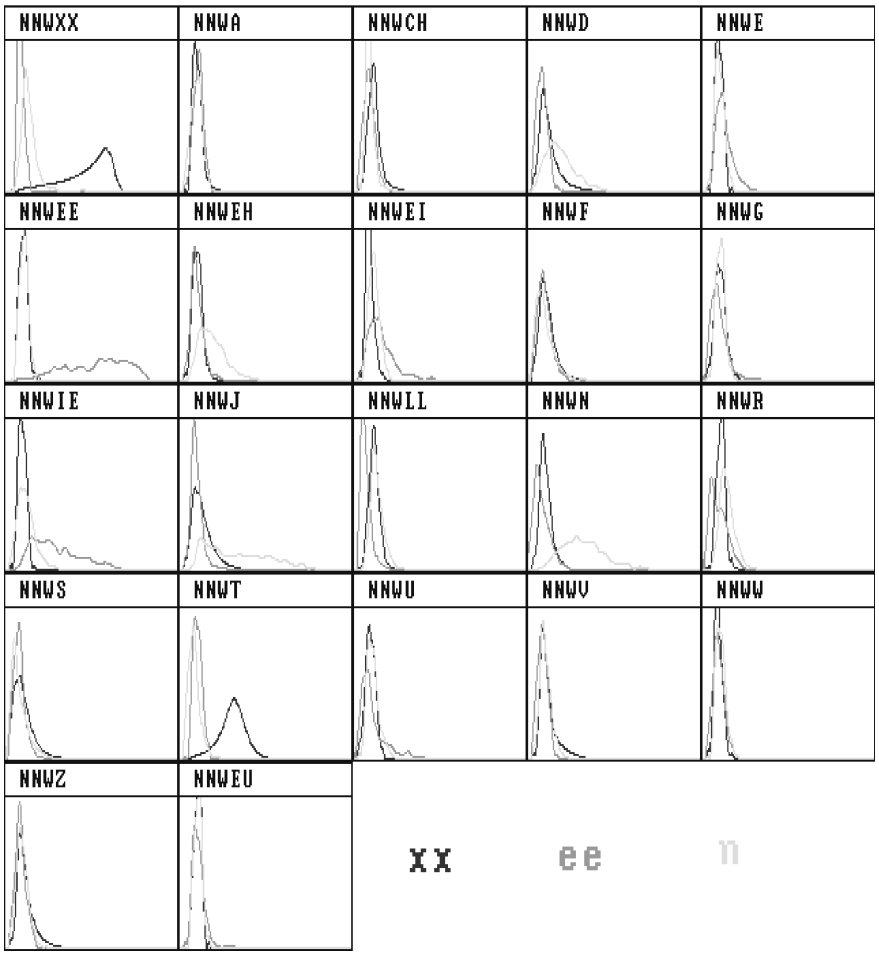

Figure 3.12: Histograms for 200 utterances of Dutch digit `een`. The dark lines represent the output histograms on the phonetically labeled MLP output units with the input data segmented to the noise `xx`. The grey lines stand for those corresponding to the part of `ee`. The light lines are for `n`. The horizontal axes are the MLP output values between [0,1] on each unit in the output layer. The vertical axes are the probabilities of occurrences. The peaks indicate the conresponding values occur most frequently. For example, the majority output values of output units `xx` and `ee`, are around 0.1 and 0.2 when the output unit label is `N` in window `NNWN`.

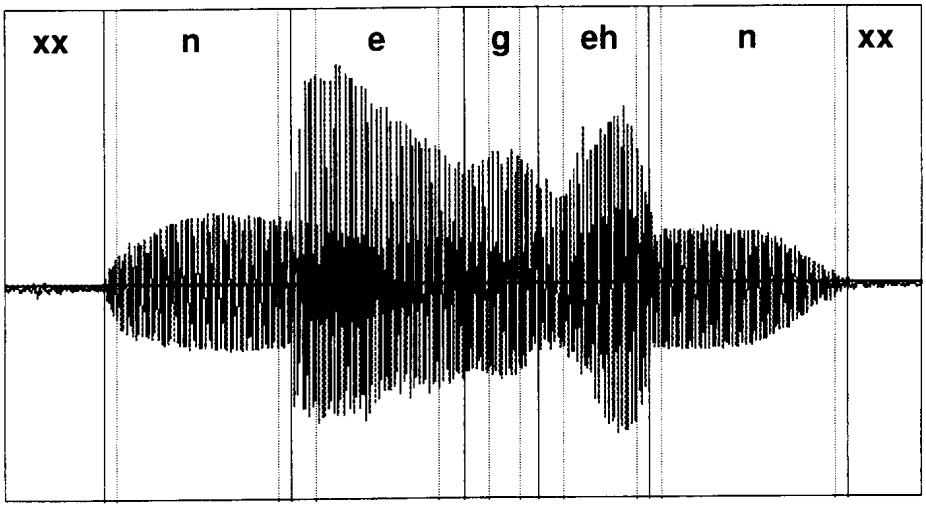

Figure 3.13: The phonetic segments for Dutch digit `negen`. The regions between two dotted lines around the solid segment lines are confusing for phonetic segmentation.

3.8.1 Fuzzy Set Theory in Speech Recognition

A human speech signal is produced by moving the articulators towards target positions that characterize a particular sound. Since these articulatory motions are subject to physical constraints, they commonly don't reach clean identifiable phonetic targets. The utility of fuzzy sets lies in their ability to model those uncertain or ambiguous speech data. Fuzzy set theory was introduced by Zadeh in 1965 (Zadeh[97]). Since then, the theory has been heavily used by researchers to enhance the algorithms. As we know, logic deals with true and false. However, a proposition can be true on one occasion and false on another. Fuzzy logic deals with propositions that can be true to a certain degree. Phonetic segmentation of the speech signal is such a proposition. Because of coarticulation, the segments are very ambiguous. In Figure 3.13, we show the phonetic segments for a Dutch digit `negen`. It is very difficult to decide the boundary in the congruous frames between two phonemes, e.g., `e` and `g`. We may be saying that the

two adjacent phonemes are fuzzy. The overlapping windowed speech features as we have described in Chapter 2 emphasizes the sequential property of speech but also enhances the degree of "belongingness" to other features.

Fuzziness in Neural Networks

Hence, to enable a system to tackle real-life situations in a manner more like humans, one may incorporate the concept of fuzzy sets into neural networks. It is to be noted that although fuzzy logic is a natural mechanism for propagating uncertainty, it may involve in some cases an increase in the amount of computation required (compared with a system using classical binary logic). This can be suitably offset by using fuzzy neural network models having the potential for parallel computation with high flexibility.

This fuzzy neural network model is capable of handling input features presented in quantitative and/or linguistic form. The components of the input vector consist of the membership values to the overlapping partitions of linguistic properties low, medium, and high corresponding to each input feature. The fuzzy neural network provides a scope for incorporating linguistic information in both the training and the testing phases of the model and increases its robustness in tackling imprecise or uncertain input specifications.

The conventional two-state neural network models generally deal with the ideal condition, where an input feature is either present or absent and each pattern belongs to either one class or another. They do not consider cases where an input feature may posses a property with a certain degree of confidence, or where a speech pattern may belong to more than one class with a finite degree of "belongingness". The MLP we describe here incorporates these concepts and is capable of classifying fuzzy patterns.

3.8.2 Fuzzy Multilayer Perceptron

As we have discussed in Section 3.2, the network passes through two phases, viz., forward pass and backward pass in the training process.

During the forward pass, the supervised learning is used to assign output membership values lying in the range $[0, 1]$ to the training vectors. Hence each output unit in the output layer may be assigned a non zero membership value instead of choosing the single unit with the highest activation. This allows modeling of fuzzy data when the feature space involves overlapping

pattern classes such that a pattern point may belong to more than one class with nonzero membership value.

During the backward pass, each error in membership assignment is fed back and the connection weights of the network are appropriately updated. The back-propagation error is computed with respect to each desired output, which is a membership value denoting the degree of belongingness of the input vector to that class. Hence the error (which is back-propagated for weight updating) has inherently more weight in case of units with higher membership values. The contribution of ambiguous or uncertain vectors to the weight correction is automatically reduced. This is natural as vectors that are more typical of their class should have more influence in determining the position and the shape of the decision surface. The learning rate and the momentum coefficient are gradually decreased until the network hopefully converges to a minimum error solution. This heuristic helps to avoid spurious local minima and usually prevents oscillations of the mean square error in the weight error surface. The network sweeps through randomly selected training data on each iteration. The utility of this approach for the modeling of output values can be further appreciated by considering a point lying in a region of overlapping classes in the feature space. In such cases, its membership in each of these classes may be nearly equal. Then there is no reason why we should follow the crisp approach of classifying this pattern as belonging to the class corresponding to that output neuron with a slightly higher activation, thereby neglecting the smaller yet significant responses obtained for the other overlapping classes.

We set the number of output units to correspond to the number of phonemes present. As speech spectra have overlapping or fuzzy class boundaries, each pattern used in training possesses nonzero belongingness to more than one class. To model such data, we clamp the desired membership values, lying in the range $[0, 1]$, at the output units during training. Then, the network back-propagates the error with respect to the desired membership values at the output.

This procedure of assigning fuzzy output membership values, instead of the more conventional binary output values, enables the MLP neural network to more efficiently classify fuzzy data with overlapping class boundaries.

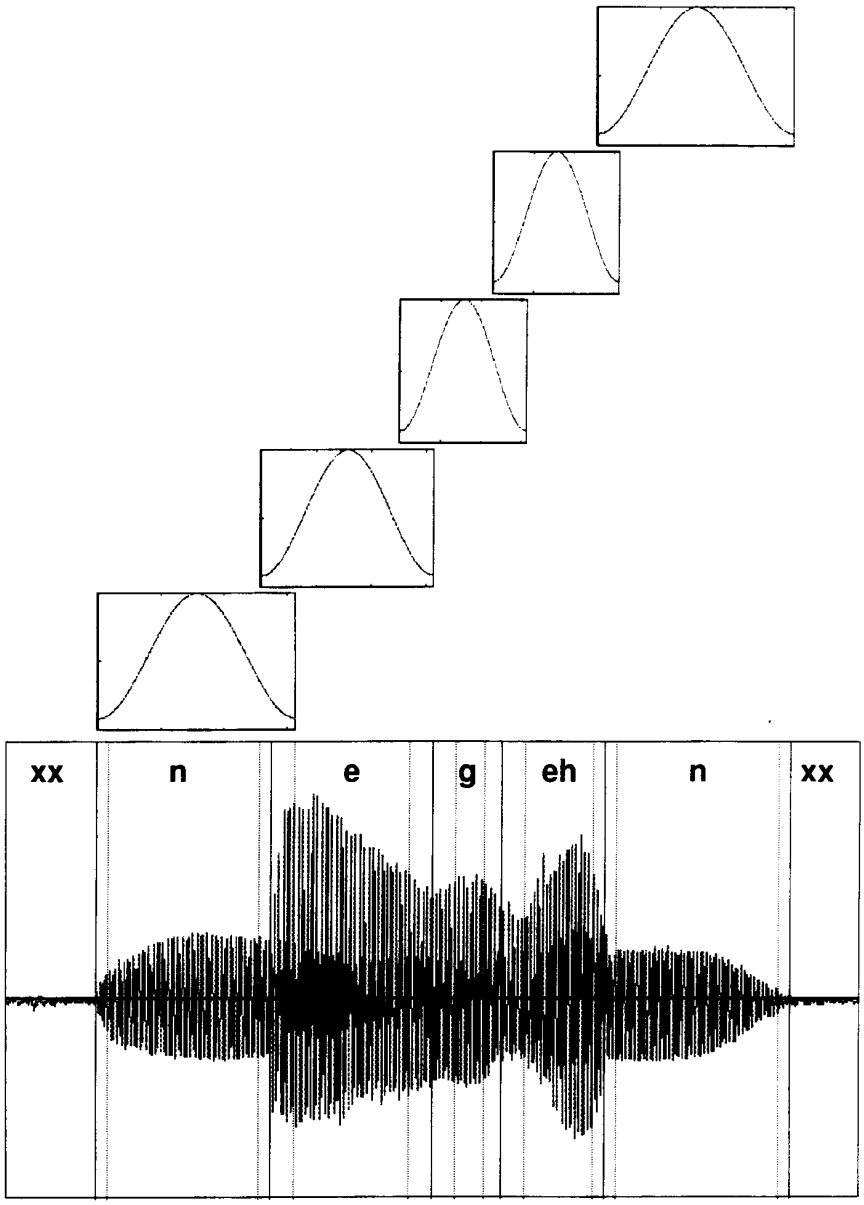

Figure 3.14: The overlapped Hamming windows as the fuzzy membership function

3.8.3 Fuzzy Output and Weighted Input

In the previous section, we have proposed the modified weight adaptation and the weighted training input. In fact, we have taken into account the degree of belongingness of the speech input pattern contributing to the crisp output. The fuzzy MLP, on the other hand, focuses on the MLP output. It uses the nature of *belongingness* or *fuzziness* from input speech patterns and propagates them to the network output.

There are several ways to decide upon the output fuzzy membership functions ([75]). Those calculations seem very complicated. Here we propose a very simple and efficient approach. We simply use the *overlapped Hamming window* as the fuzzy membership function to decide upon the output analog values in the output layer. The target values on the unit corresponding to the current phoneme and its adjacent phoneme are set to the values calculated from the Hamming window. The target values on the other units in the output layer are set to zero. The computation is illustrated in Figure 3.14.

3.9 Summary

In this chapter, we have discussed the Multi-Layer Perceptron and its Back Propagation learning algorithm. The MLP super pattern classification properties lie in its discriminative power, its capability to deal with non-explicit knowledge and its freedom to take all kinds of information into account.

We have proposed modified weight adaptation, network scaling, random data selection and training data weighting strategies aiming at speech recognition tasks.

The MLP has developed alternate internal representations that can link quite different acoustic realizations to the higher level phonetic concept. We have presented two methods for simple phoneme based frame classification using MLPs and the voiced stop consonant recognition using Time Delay Neural Networks respectively.

We have also given the neural network a fuzzy interpretation with the fuzzy set theory. The overlapping speech input, the connectionist weights and fuzzy membership output provide a sound fuzzy MLP classification for speech recognition. We proposed to use the overlapped Hamming window as the fuzzy membership function.

More detailed discussions on how to use MLPs for speech recognition will be presented in the next chapter. The experiments concerning the MLP issues discussed in this chapter will be presented in chapter 5.

Chapter 4

Vector Quantization and Connectionist Vector Quantization

4.1 Introduction

Quantization, the process of approximating continuous amplitude signals by discrete signals, is an important aspect of data compression or coding, the field concerned with the reduction of the number of bits necessary to transmit or store analogue data, subject to a distortion or fidelity criterion. The independent quantization of each single value or parameter is termed *scalar quantization*. In contrast, the joint quantization of a block of parameters is termed *Vector Quantization* (VQ).

In HMM-based speech recognition, vector quantization serves an important role in describing discrete acoustic prototypes of speech signals for the discrete HMM. This chapter will first review the principles of conventional vector quantization, hybrid VQ/HMM modeling, and several related algorithms which have been implemented in our speech recognition system. In Section 4.6, we address the most important issues in this dissertation: using the Multi-Layer Perceptron neural network as a labeler for Hidden Markov Modeling. Using the fuzzy theory described in the previous chapter, we address another approach: the Multi-Layer Perceptron fuzzy labeler. We end with a summary of this chapter.

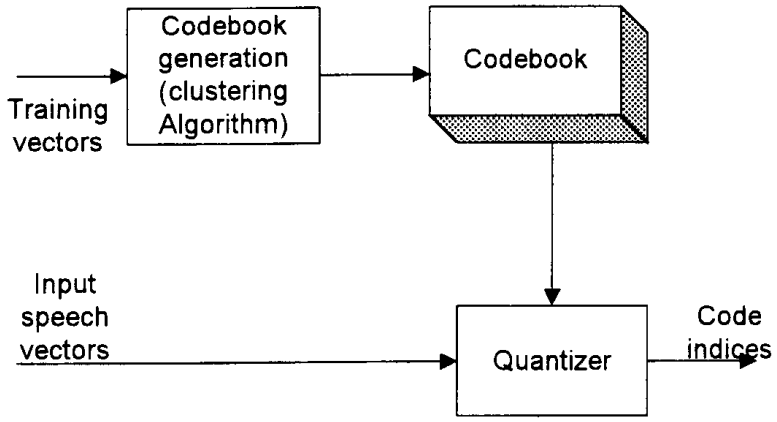

Figure 4.1: A schematic diagram of vector quantization techniques

4.2 Vector Quantization

The basic concept of vector quantization as applied to speech recognition is schematically depicted in Figure 4.1. A training speech sequence is first used to generate the codebook. As we have described in chapter 2, the speech signal is segmented (windowed) into successive short frames and each frame of speech is represented by a vector of finite dimensionality. In our case, the vector is the result of either the filter-bank analysis or the LPC analysis which captures the time-various spectral characteristics of the speech signal.

If we compare the information rate of the vector representation to that of the raw (uncoded) speech waveform, we see that the spectral analysis has significantly reduced the required information rate. Consider, for example, 10-kHz sampled speech with 16-bit speech amplitudes. A raw signal information rate of 160 kbps is required to store the speech samples in PCM format. For the spectral analysis, consider vectors of dimension $p = 10$ using a 10 ms frame rate. If we again represent each spectral component to 16-bit precision, the required storage is about 16 kbps. It is a 10-to-1 reduction over the uncompressed signal. Such compressions in storage rate is very impressive.

Based on the concept of ultimately needing only a single spectral representation for each basic speech unit, it may be possible to further reduce

the raw spectral representation of speech to those drawn from a small, finite number of unique spectral vectors, each corresponding to one of the basic speech units (i.e., the phonemes). Before the neural network was invented, especially the MLP neural network (Ma and Van Compernolle[53]), this ideal representation was, of course, impractical, because there is so much variability in the spectral properties of each of the basic speech units. However, the concept of building a codebook of distinct analysis vectors, although with significantly more code words than the basic set of phonemes, remains an attractive idea and is the basis behind a set of techniques commonly called vector quantization methods.

Assume that we require a codebook with about 1024 unique spectral vectors (i.e., about 25 variants for each of the 40 basic speech phonetic units in American English). Then to present an arbitrary spectral vector all we need is a 10 bit number which is the index of the codebook vector that best matches the input vector. A total bit rate of about 1 kbps is required to present the spectral vectors of a speech signal. This rate is about 1/16 the rate required by the continuous spectral vectors. Hence the VQ representation is potentially an extremely efficient presentation of the spectral information in the speech signal. This is one of the main reasons for the interest in VQ methods.

Before discussing the concepts involved in designing and implementing a practical VQ system, we first discuss the advantages and disadvantages of this type of representation for speech recognition. The key advantages of the VQ representation are

- reducing storage for spectral analysis information. We have already shown that the VQ representation is potentially very efficient. This efficiency can be exploited in a number of ways in practical VQ-based speech-recognition systems.

- reducing computation for determining similarity of spectral analysis vectors. In speech recognition a major component of the computation is the determination of spectral similarity between a pair of vectors. Based on the VQ representation, this spectral similarity computation is often reduced to a table lookup of similarities between pairs of codebook vectors.

- discrete representation of speech sounds. By associating a phonetic label (or possibly a set of phonetic labels or a phonetic class) with

each codebook vector, the process of choosing a best codebook vector to represent a given spectral vector becomes equivalent to assigning a phonetic label to each spectral frame of speech. A range of recognition systems exists that exploit these labels so as to efficiently recognize speech.

The disadvantages of the use of a VQ codebook to present spectral vectors are

- an inherent spectral distortion in representing the actual analysis vector. Since there is only a finite number of codebook vectors, the process of choosing the "best" representation of a given spectral vector is inherently equivalent to quantizing the vector and leads, by definition, to a certain level of quantization error. As the size of the codebook increases, the size of the quantization error decreases. However, with any finite codebook there will always be some nonzero level of quantization error.
- the storage required for codebook vectors is often nontrivial. The larger we make the codebook (so as to reduce quantization error), the more processing time is needed for the codebook vector searching, the more storage is required for the codebook entries. Hence an inherent trade-off among quantization error, processing for choosing the codebook vector, and storage of codebook vectors exists, and practical designs need to balance each of these three factors.

4.3 Conventional Vector Quantization

Vector quantization(VQ) reduces the data redundancy to be transmitted. This inevitably causes distortion between original data and transmitted data as we have discussed above. A key point of VQ is to minimize the distortion. A set of parameters is quantized as a whole, minimizing the global distortion. The finite set of possible quantized vectors is stored in a *codebook*. After quantization, the input parameter vector is represented by the corresponding *label* of the codebook entry that shows the smallest distortion. In HMM-based speech recognition, the goal of VQ is to generate a number of acoustic prototype vectors (VQ codewords) from a large sample of training vectors such that the codewords can represent the distribution of the training vectors; and minimize the total distortion over all

training vectors. The VQ partitions the acoustic feature space into separate regions according to some distortion measure regardless of the probability distributions of the original data. This introduces errors in the partition operations which may destroy the original signal structure.

4.3.1 Vector Quantization

In vector quantization, the real-valued, continuous-amplitude d-dimensional vector $\boldsymbol{x}$ is mapped to another real-valued, discrete(or continuous)-amplitude d-dimensional vector $\boldsymbol{z}$. It is then said that $\boldsymbol{x}$ is quantized to $\boldsymbol{z}$.

$$\boldsymbol{z} = q(\boldsymbol{x}) \tag{4.1}$$

Where

- $q()$ is the quantization operator.
- $\boldsymbol{x} = (\boldsymbol{x}_1, x_2, ..., x_d)^t \in R^d$ is a d-dimensional vector whose components $\{x_k, 1 \leq k \leq d\}$ are real-valued, continuous-amplitude random variables.
- $\boldsymbol{z} = (z_1, z_2, ..., z_d)^t$ $\boldsymbol{z}$ typically takes one of a finite set of values $\boldsymbol{Z} = \{\boldsymbol{z}_i, 1 \leq i \leq L\}$, where $\boldsymbol{z}_i = (z_{i1}, z_{i2}, ..., z_{id})^t$.

The set $\boldsymbol{Z}$ is referred to as the *codebook*, L is the size of the codebook, and $\{\boldsymbol{z}_i\}$ is the set of codewords. The size L of the codebook is also called the number of levels in the codebook. To design a codebook, the d-dimensional space of the original random vector $\boldsymbol{x}$ can be partitioned into L regions or cells $\{C_i, 1 \leq i \leq L\}$ and each cell C_i is associated with a vector $\boldsymbol{z}_i$. The quantizer assigns the codeword $\boldsymbol{z}_i$ if $\boldsymbol{x}$ lies in C_i.

$$q(\boldsymbol{x}) = \boldsymbol{z}_i, \quad if \quad \boldsymbol{x} \in C_i \tag{4.2}$$

This codebook design process is also known as *training* or *populating* the codebook. An example of partitioning a two-dimensional space ($d = 2$) for the purpose of vector quantization is shown in Figure 4.2. The shaded region enclosed by the bold lines is the cell C_i. Any input vector $\boldsymbol{x}$ that lies in the cell C_i is quantized as $\boldsymbol{z}_i$. The shapes of the various cells can be different, and the positions of the codewords corresponding to the cells are determined by minimizing the average distortion D_i associated with the

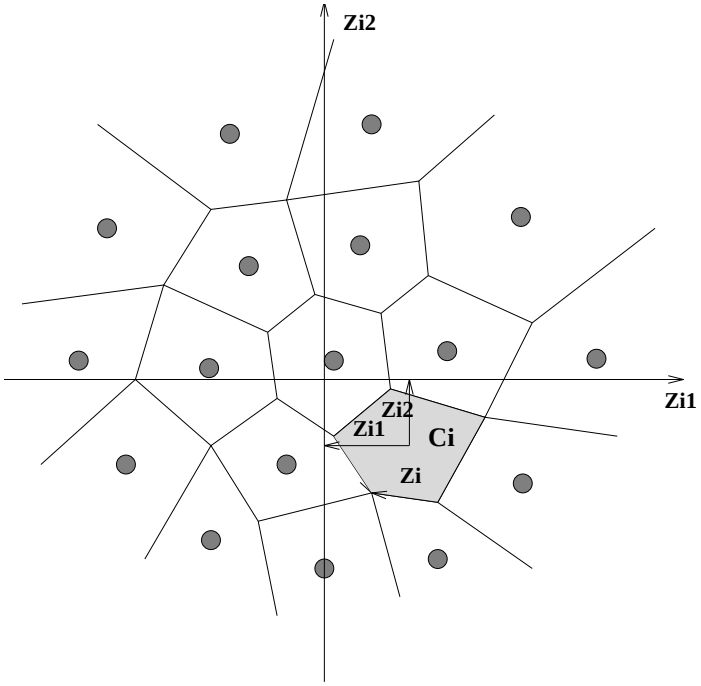

Figure 4.2: Partitioning of two-dimensional space into 18 cells. All input vectors in cell C_i will be quantized as the code vector $\mathbf{z}_i$. The shapes of the various cells can be very different.

corresponding cells. The positions of the codewords within each cell are shown by dots in Figure 4.2.

When $\boldsymbol{x}$ is quantized as $\boldsymbol{z}$, a distortion measure $d(\boldsymbol{x}, \boldsymbol{z})$ can be defined between $\boldsymbol{x}$ and $\boldsymbol{z}$ to measure the quantization quality. The distortion measure between $\boldsymbol{x}$ and $\boldsymbol{z}$ is also known as a distance measure in the speech recognition context. The measure must be tractable in order to be computed and analyzed, and also must be subjectively relevant so that differences in distortion values can be used to indicate differences in speech signals. The most commonly used measure is the *Euclidean distortion measure* which assumes that the distortions contributed by quantizing the different parameters are equal.

4.3.2 Clustering

To design an L-level codebook, it is necessary to partition d-dimensional space into L cells and associate with each cell a quantized vector. When the overall average distortion D is used as a criterion in codebook design, we say a quantizer is *optimal* if the overall average distortion D is minimized over all L-levels of the quantizer. There are two necessary conditions for optimality.

The first condition is that the optimal quantizer is realized by using a nearest neighbor selection rule:

$$q(\boldsymbol{x}) = \boldsymbol{z}_i, \;\; if \;\; d(\boldsymbol{x}, \boldsymbol{z}_i) \leq d(\boldsymbol{x}, \boldsymbol{z}_j) \quad j \neq i, 1 \leq j \leq L \tag{4.3}$$

The quantizer must choose the code word that results in minimizing distortion with respect to $\boldsymbol{x}$, i.e. $\boldsymbol{x}$ is selected for the corresponding cell C_i.

The second condition for optimality is that each codeword $\boldsymbol{z_i}$ is chosen to minimize the average distortion D_i in cells C_i. Since the overall average distortion D is a linear combination of all average distortions D_i in cell C_i, $\boldsymbol{z_i}$ can be independently computed after classification of $\boldsymbol{x}$. Given the average distortion of cluster C_i, the minimization of D with respect to $\boldsymbol{z}_i$ is given by setting the derivative of D_i to zero. We reach a solution where $\boldsymbol{z}_i$ is simply the sample mean of all the training vectors, $\boldsymbol{x}$, contained in cluster C_i[47].

Good codebook construction is rarely possible by rule. Most often they are constructed in an iterative way on the basis of a lot of data. An optimal codebook will be the one that minimizes VQ distortion on the global training database. No globally optimum techniques for codebook construction

are known, most reach a local optimum. The following K-means algorithm has been used in our codebook design.

K-means Clustering

In K-means clustering, the basic idea is to divide the set of training vectors into L clusters $\boldsymbol{z}_i$ $1 \leq i \leq L$ in such a way that the two necessary conditions for optimality described above are satisfied. An existing codebook is optimized in an iterative way so that the global distortion is guaranteed to decrease (at worst stay the same) with every iteration.

In the process of minimizing the average distortion measure, the K-means procedure actually breaks the minimization process into two steps. Assuming that the centroid $\boldsymbol{z}_i$ (or mean) for each cluster center C_i has been found, then the minimization process is found simply by partitioning all the training vectors into their corresponding closest cluster according to the distortion measure. On the other hand, if all of the partitions are obtained, the minimization process involves its corresponding within-cluster average distortion. By iterating over those two steps, a new value of overall distortion which is smaller than that of the previous step can be obtained. However, the K-means clustering can only converge to a local optimum. Global optimality may be approximated by repeating the K-means algorithm for several sets of codebook initialization values and then choosing the codebook that produces the minimum overall distortion. However, such a criterion may not necessarily lead to optimal speech recognition accuracy (Huang [29]).

An extend K-means algorithm, the LBG algorithm proposed by Linde, Buzo and Gray, is also commonly used [45]. The LBG algorithm iteratively splits the training data into $2, 4, ..., 2^m$ partitions, with a centroid for each partition. The centroid is determined by iterative refinement as for the K-means clustering. The procedure of the generalized Lloyd algorithm or the K-means clustering algorithm consists of the following steps:

1. Initialization. Set L (number of partitions or clusters) = 1. Find the centroid of all the training frames.

2. Splitting. Split L into 2L partitions. Set L= 2L.

3. Classification. Classify the set of training data $\boldsymbol{x}_k$ into one of the clusters C_i according to the minimum distance classifier.

4. Codebook updating. Update the codeword of every cluster by computing the centroid in each cluster.

5. Iteration. If the reduction in the overall distortion D at each iteration relative to the value D at the previous iteration is below a selected threshold, go to step 6, otherwise go to step 3.

6. Termination. If L equals the VQ codebook size required, stop clustering, otherwise go to step 2.

4.3.3 Labeling

After receiving the codebook by the K-means cluster, we need to quantize the input spectrum with the codebook. This process is called labeling. The classification procedure for arbitrary spectral vectors is basically a full search through the codebook to find the best match.

Thus if we denote the codebook vectors of an L-vector codebook as $\boldsymbol{z}_i$, $1 \leq i \leq L$, and we denote the spectral vector to be classified as $\boldsymbol{x}$, the index, i^*, of the best codebook entry is

$$i^* = arg \min_{1 \leq i \leq L} d(\boldsymbol{x}, \boldsymbol{z}_i) \tag{4.4}$$

The above equation can be interpreted as

$$\sum_{j=1}^{d} |x_j - z_{i^* j}|^2 \leq \sum_{j=1}^{d} |x_j - z_{ij}|^2 \quad \forall i \quad 1 \leq i \leq L \tag{4.5}$$

For codebooks with large values of L (e.g., $L \geq 1024$), the computation of Equation 4.4 or Equation 4.5 could be excessive, depending on the exact details of the distance measure; hence, alternative procedures for the design of VQ codebooks have been investigated, among which MLP-VQ is one of the most successful methods. We will discuss such methods in the later sections of this chapter.

To receive a generalized codebook, we also need to have a generalized training database. Our database provides more than enough speech data to guarantee the generalized codebook. At this stage, we have finished the basic VQ preparation for a discrete HMM system.

4.4 Hidden Markov Modeling

4.4.1 Discrete HMMs

The normal discrete HMM is usually defined as a 5-tuple (Q, Z, π, A, B). Q is a set of N states $q_1, q_2, ..., q_N$. Z is the codebook, a set of L labels $\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_L$ corresponding to prototypical spectra, π is a vector which specifies the initial distribution, $(\pi_1, \pi_2, ..., \pi_N)$, where $\pi_i = \text{Prob}(q_t = i)$. Denote the sequence of normal observations by $O = (o_1, o_2, ..., o_T)$, where each o_t for $1 \leq t \leq T$ is some $\boldsymbol{z}_i \in Z$. A is a matrix of state transition probabilities, $A = [a_{ij}]$, $1 \leq i, j \leq L$, where $a_{ij} = \text{Prob}(q_{t+1} = j \mid q_t = i)$. B is a matrix of observation probabilities, $B = [b_j(k)]$, $1 \leq j \leq N$,$1 \leq k \leq L$ where $b_j(k) = \text{Prob}(o_t = \boldsymbol{z}_k \mid q_t = j)$. The following HMM algorithms are used for HMM training and test.

4.4.2 Conventional HMM Algorithms

The Viterbi algorithm is often used to save on computations and to obtain the state sequence at the same time. The implementation procedure is described as follows:

- Initialization

$$\delta_1(i) = \pi_i b_i(\boldsymbol{o}_1) \quad 1 \leq i \leq N \tag{4.6}$$

$$\phi_1(i) = 0 \tag{4.7}$$

- Recursion

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(\boldsymbol{o}_t) \quad 2 \leq t \leq T, 1 \leq j \leq N \tag{4.8}$$

$$\phi_t(j) = arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \quad 2 \leq t \leq T, 1 \leq j \leq N \tag{4.9}$$

- Termination Star * indicates the optimised results.

$$Prob(O)^* = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{4.10}$$

$$q_T^* = arg \max_{1 \leq i \leq N} [\delta_T(i)] \tag{4.11}$$

- Path backtracting

$$q_t^* = \phi_{t+1}(q_{t+1}^*) \quad T-1 \geq t \geq 1 \tag{4.12}$$

Where $\delta_t(j)$ is the highest probability along a single path ending at state j, at time t. $\phi_t(j)$ is the Viterbi path array.

4.4.3 Modified Observation Probability

Provided that each codeword of the VQ codebook Z is represented by a probability density function $f(\boldsymbol{x}|q_t)$, for a given state q_t of the HMM, the probability density function that produces a vector $\boldsymbol{x}$ can then be written as (Huang [29]).

$$w_{q_t}(\boldsymbol{x}) = \sum_{j=1}^{L} f(\boldsymbol{x}|q_t) \tag{4.13}$$

$$= \sum_{j=1}^{L} f(\boldsymbol{x}|\boldsymbol{z}_j, q_t) Prob(\boldsymbol{z}_j|q_t) \tag{4.14}$$

where L denotes the VQ codebook level. For simplicity, the probability density function $f(\boldsymbol{x}|\boldsymbol{z}_j, q_t)$ can be assumed to be independent of the Markov states q_t. Thus, for a given state i, Equation 4.14 can be written as (using codebook index j to represent $\boldsymbol{z}_j$):

$$w_i(\boldsymbol{x}) = \sum_{j=1}^{L} f(\boldsymbol{x}|\boldsymbol{z}_j) Prob(\boldsymbol{z}_j|q_t = i) \tag{4.15}$$

$$= \sum_{j=1}^{L} f(\boldsymbol{x}|j) b_i(j) \tag{4.16}$$

In practice, Equation 4.16 can be simplified by using the M most significant values of $f(\boldsymbol{x}|j)$ for each $\boldsymbol{x}$ without affecting the performance. Experience has shown that values in the range of $2-8$ are adequate. This can be conveniently obtained during the VQ operations by sorting the VQ output and keeping the M most significant values. Let $\eta(\boldsymbol{x})$ denote the codeword entries j of the set of VQ codewords, $\boldsymbol{z}_j$, for those most significant values

of $f(\boldsymbol{x}|j)$ of $\boldsymbol{x}$. Equation 4.16 can be rewritten as

$$w_i(\boldsymbol{x}) = \sum_{j \in \eta(\boldsymbol{x})} f(\boldsymbol{x}|j) b_i(j) \tag{4.17}$$

Since the number of VQ codeword entries in $\eta(\boldsymbol{x})$ is of lower order than the VQ level L, Equation 4.17 can significantly reduce the amount of computational load for subsequent modeling compared with Equation 4.16. The computational complexity of the hybrid HMM mainly depends on the VQ level L and the size of $\eta(\boldsymbol{x})$.

4.4.4 Modified HMMs

From the definition in Equation 4.17, we can compute the modified Viterbi algorithm. The modified Viterbi algorithm then becomes:

$$\delta_1(i) = \pi_i w_i(o_1) \quad 1 \le i \le N \tag{4.18}$$

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] w_j(o_t) \quad 2 \le t \le T, 1 \le j \le N \tag{4.19}$$

and the result is

$$Prob(O)^* = \max_{1 \le i \le N} [\delta_T(i)] \tag{4.20}$$

as usual.

4.5 VQ/HMM Hybrid Modeling

Given the VQ codebook index j, the probability density function $f(\boldsymbol{x}|j)$ can be estimated with the parametric Gaussian probability distribution, or by non-parametric or heuristic methods, such as multi-labeling and Fuzzy VQ. The estimation of $f(\boldsymbol{x}|j)$ is crucial in the VQ/HMM system design. The models we describe here will provide the classic approaches compared with the neural network MLP approaches presented in the next section. We first discuss the baseline discrete VQ/HMM, which is the basis for other VQ/HMM hybrid models.

4.5.1 Discrete VQ/HMM

In a discrete HMM, the discrete labels obtained by vector quantization are the observations for the HMM. A discrete HMM system is shown in Figure

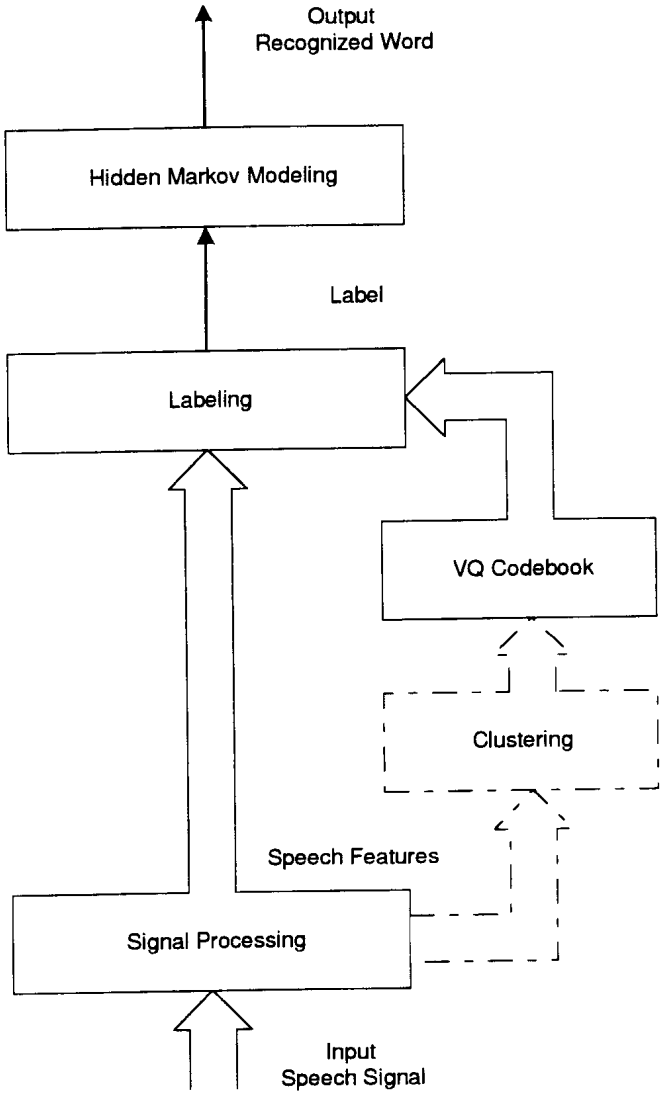

Figure 4.3: A simple VQ/HMM speech recognition system

4.3. Clustering is only used in VQ codebook generation, which is indicated by the dashed lines. The advantage of this approach is that each HMM state must model a finite number of discrete labels. This typically results in faster HMM computation. However, this mapping from continuous acoustic space to quantized discrete space can cause serious quantization errors for subsequent hidden Markov modeling. The following smoothing techniques (Section 4.5.2 through 4.5.6) focus on reducing the vector quantization errors.

4.5.2 Mixture Density HMM

Another disadvantage of the discrete HMM is that the VQ codebook and the discrete HMM are separately modeled, which may not be an optimal combination for pattern classification (Huang et al [30]).

The VQ codebook can be modeled as a family of Gaussian probability density functions (pdfs) as shown in Figure 4.4, where solid lines show conventional VQ partitions, dotted lines show mixture densities and dashed lines indicate the fuzzy observations (in our later discussion). We observe that the probability density functions can overlap, rather than be partitioned, to present the acoustic feature space. The centroid obtained via such a representation may be quite different from that obtained using the conventional K-means algorithm, since the distribution property of signals can be taken into consideration. The use of a parametric family of finite mixture densities (a mixture density VQ) can be closely combined with the HMM methodology. The observation probability is calculated by the following Equation:

$$w_i(\boldsymbol{x}) = \sum_{j \in \eta(\boldsymbol{x})} N(\boldsymbol{x}, \mu_j, \Sigma_j) b_{ij} \tag{4.21}$$

The Gaussian density function $N(\boldsymbol{x}, \mu_j, \Sigma_j)$ can be estimated with the EM algorithm [31], or simple estimates of the covariance matrices based on the conventional VQ codebook [31].

In this approach the VQ problems and HMM modeling problems can be combined under the same probabilistic frame work to obtain an optimized VQ/HMM combination. We are not going to use this model in our MLP approaches as both the phonetic MLP output labels and their corresponding output values are not in the Gaussian parametric distribution, as shown in

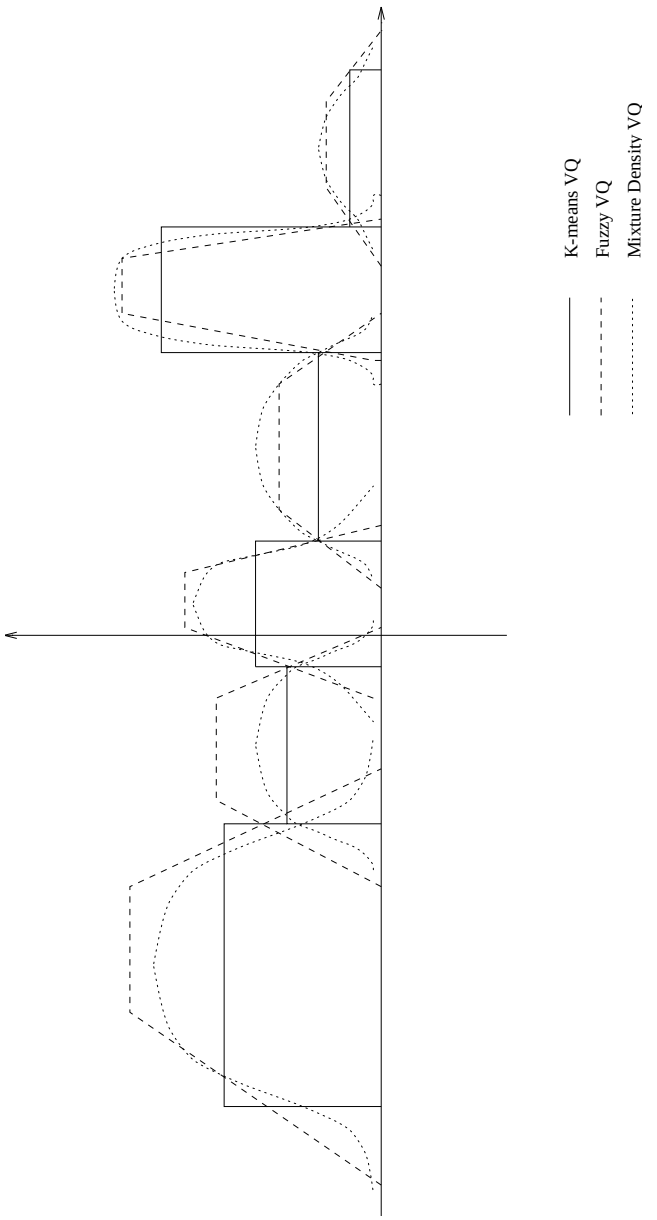

Figure 4.4: Comparison of three different VQ techniques: Mixture density, VQ and Fuzzy VQ

the histograms described in Section 3.7 (Only the irrelevant output values there seem in Gaussian distribution).

4.5.3 Multi-Labeling

Multi-labeling, which generates multiple labels at each frame, was introduced by Nishimura et al [59]. Let $d(\boldsymbol{x}, \boldsymbol{z}_j)$ be a vector quantization error between vector $\boldsymbol{x}$ and the codeword $\boldsymbol{z}_j$ $1 \leq j \leq L$. L is the number of codebook levels. By using the vector quantization error at each frame, $f(\boldsymbol{x}|j)$ is obtained in the following normalization manner.

1. **Initialization:** Set iteration number $j = 1$ and find the minimum distortion
$$d_{min} = \min_{1 \leq j \leq L} d(\boldsymbol{x}, \boldsymbol{z}_j) \quad 1 \leq j \leq L \tag{4.22}$$

2. **Normalization:** Use the minimum distortion to normalize the other distortion
$$d_{norm}(j) = d(\boldsymbol{x}, \boldsymbol{z}_j)/d_{min} \quad for\ each \quad j \quad 1 \leq j \leq L \tag{4.23}$$

3. **Comparison:** Decide the value of $f(\boldsymbol{x}|j)$
$$f(\boldsymbol{x}|j) = \alpha \quad if\ d_{norm}(j) < \eta, \quad else\ f(\boldsymbol{x}|j) = 0 \tag{4.24}$$

Where η is a preselected threshold. It is greater than 1.0; α is a constant value and $\sum f(\boldsymbol{x}|j)$ is constant at every frame; $d_{norm}(j)$ is a normalized distance which indicates the similarity to the top candidate. An example of a multi-labeled sequence is generated as shown in Figure 4.5.b.

Apparently, this multi-labeled sequence is applicable to both training and decoding of an HMM. For estimating the parameters of an HMM, the conventional Baum-Welch algorithm is still available. All labels generated at each frame are equally used for parameter estimation. Thus the HMM with multi-labeling requires as much memory as the HMM with conventional labeling. In the decoding process, the maximum output probability among labels at each frame is used by the Viterbi algorithm. When this method is applied to the training, it seems that it gives an effect similar to the parameter smoothing method (Sugawara, et al [79]), which smoothes the trained parameters by using a distance between label-prototypes.

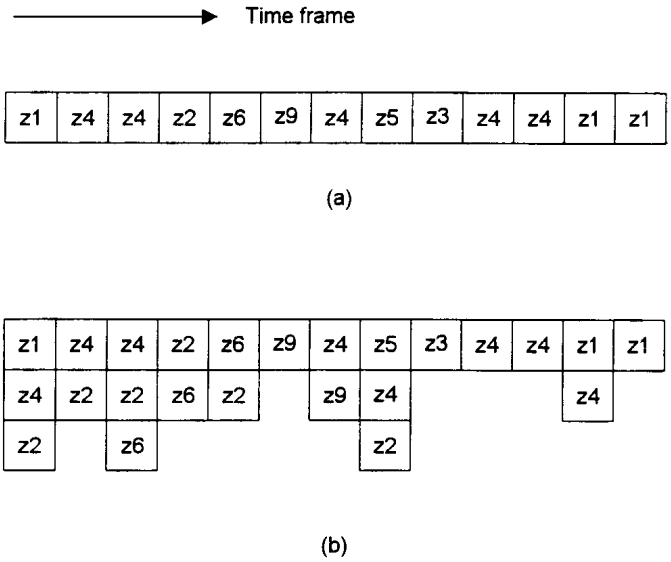

Figure 4.5: Label sequences by the conventional labeling (a) and the multi-labeling (b). $\mathbf{z}_j$ is the codeword and indicates the label number j.

4.5.4 Multi-Dimensional Labeling (Multiple Codebooks)

A distinctive technique is HMM based upon multiple VQ codebooks, which has been shown to offer improved speech recognition accuracy (Lee [43]).

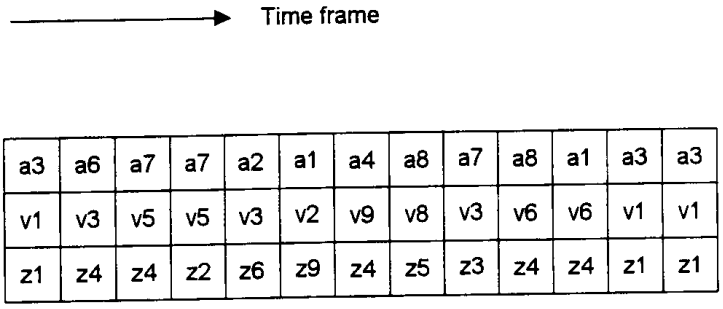

Figure 4.6: Frame sequences labeled by the multi-dimensional VQ.

It is well known that parameter velocity (first derivatives) and parameter acceleration (second derivatives) are an effective means of describing the speech dynamics [94]. Because these 3 parameter sets (the spectral parameters, their first and second derivatives) can be assumed independent, it is possible to make quantization separately, and then those codes are unified into a single label. On the basis of this labeling, multiple features are simultaneously evaluated in an HMM with a conventional formulation. Since these features are appreciably independent of each other, VQ distortion can be significantly minimized by partitioning the parameters into separate codebooks. The observation probability is computed in the following way:

$$w_i(\boldsymbol{x}) = w_i(\boldsymbol{z})w_i(\boldsymbol{v})w_i(\boldsymbol{a}) \tag{4.25}$$

$$= b_i(\boldsymbol{z})b_i(\boldsymbol{v})b_i(\boldsymbol{a}) \tag{4.26}$$

Where vectors $\boldsymbol{z}$, $\boldsymbol{v}$ and $\boldsymbol{a}$ are the speech parameter vector, its first derivatives and its second derivative respectively, and $f(\boldsymbol{x}|j) = 1$. This method is effective at reducing both memory space and processing time for labeling compared with a method which quantizes a vector of multi-features as a whole. On the other hand, the HMM with a labeling of multi-features may require much memory space to save the output probabilities compared with

the HMM with labeling based on only spectral features alone (Lee [43]). The multi-dimensional vector quantization is shown in Figure 4.6.

4.5.5 Multi-Dimensional Multi-Labeling

Combining the multi-dimensional labeling and the multi-labeling, we obtain a *multi-dimensional multi-labeling* VQ HMM system (Nishimura et al [59]). Figure 4.7 shows an example of the multi-dimensional multi-labeling method applied to HMMs. Here multi-labeling is done within each parame-

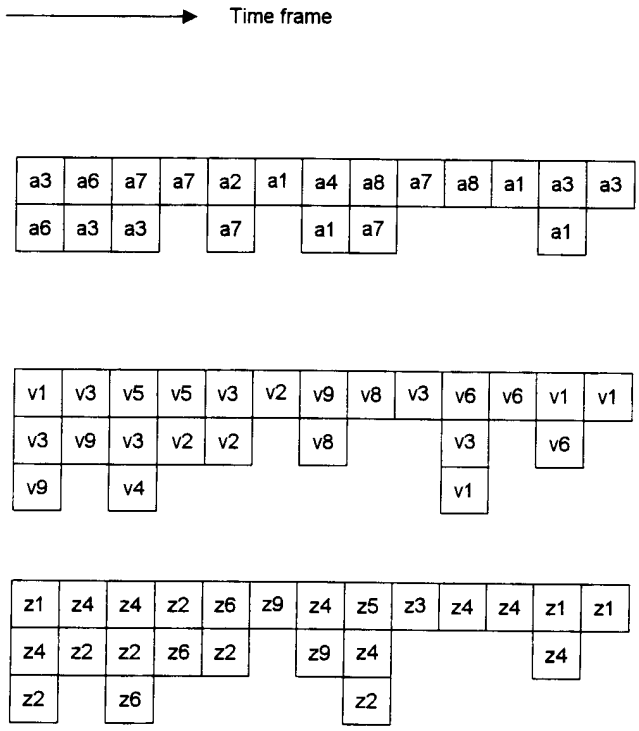

Figure 4.7: Frame sequences labeled by the multi-dimensional multi-labeling VQ.

ter set (spectral parameters, their first and second derivatives). Considering the size of each codebook, the number of code books and the number of labels for each parameter, this method may need more computation and

more memory, compared with the multi-dimensional labeling and the multi-labeling.

4.5.6 Fuzzy Vector Quantization

The Fuzzy Vector Quantization (FVQ) was proposed by Tseng et al [81] in 1987. Fuzzy vector quantization is based on the fuzzy logic theory, the K-means classification method and the VQ/HMM hybrid modeling. The dashed lines indicate the fuzzy observations in Figure 4.4. The fuzzy VQ introduces the closeness measure using the quantization error and smoothes the hard decision boundary generated by the K-mean clustering.

Let $d(\boldsymbol{x}_i, \boldsymbol{z}_j)$ represent the Euclidean distortion between input vector x_i and codeword $\boldsymbol{z}_j$ as we have described in Section 4.2. The FVQ maps an input vector $\boldsymbol{x}_i$ into an output vector $\mathbf{o}_i$ (Tseng [81]):

$$\mathbf{o}_i = (m_{i1}, m_{i2}, ..., m_{iL}) \tag{4.27}$$

Vector $\boldsymbol{o}_i$ is chosen in the following way according to the fuzzy c-means rule (Dezdek [7]),

$$m_{ij} = [\sum_{k=1}^{L} [d(\boldsymbol{x}_i, \boldsymbol{z}_j)/d(\boldsymbol{x}_i, \boldsymbol{z}_k)]^{1/(F-1)}]^{-1} \tag{4.28}$$

in order to minimize the fuzzy objective function (Dezdek [7]):

$$D = \sum_{i=1}^{T} \sum_{j=1}^{L} m_{ij}^F d(\boldsymbol{x}_i, \boldsymbol{z}_j) \tag{4.29}$$

Where $F > 1$ is a constant called the *degree of fuzziness*. It should be noticed that as F tends to infinity, each component of o_i tends to $1/M$; as F tends to 1, then the component corresponding to the minimum value of $d(\boldsymbol{x}_i, \boldsymbol{z}_j)$ tends to 1, and all other components tend to 0. In our implementation, we use $F = 2$ for fuzziness and simplicity. D is the overall average distortion. Note that the components of $\boldsymbol{o}_i$ are positive and sum to 1.

$$\sum_{j=1}^{L} m_{ij} = 1 \tag{4.30}$$

The largest component is the one corresponding to the smallest value of $d(\boldsymbol{x}_i, \boldsymbol{z}_j)$. Thus $\boldsymbol{o}_i$ can be interpreted as a probability mass vector describing

the probability that input vector $\boldsymbol{x}_i$ is represented by the codebook $\boldsymbol{Z}$. Then the observation probability $w_i(\boldsymbol{o}_t)$ is calculated using the equation

$$w_i(\boldsymbol{o}_t) = \sum_{j=1}^{N} m_{tj} b_i(j) \tag{4.31}$$

This method, however, has the disadvantage of requiring much more training and recognition time compared to the normal Viterbi algorithm.

There is a common disadvantage for above approaches: they are all based on K-means clustering which can only provide linear solutions. In the following section, we will provide novel non-linear approaches by taking advantage of non-linear properties of the MLP neural networks.

4.6 Multi-Layer Perceptron Vector Quantization

4.6.1 Introduction

In recent years, there has been considerable interest in the use of Artificial Neural Networks for speech recognition. Because Multi-Layer Perceptrons (MLPs) have excellent pattern recognition properties and Hidden Markov Models (HMMs) have powerful dynamic time warping capabilities, many researchers have tried to combine MLPs with HMM in a hybrid fashion. The benchmark work was done by Bourlard and Wellekens [10] in 1990. They proved that MLPs trained for phonetic classification on the frame level can be regarded as probability estimators for being in the corresponding phonetic HMM state, provided that the network contains enough parameters and has converged to a global minimum. Most research on hybrid neural network/HMM systems up till now has therefore concentrated on the use of MLPs or other types of networks (Recurrent Neural Networks, Radial Basis Functions (RBFs), ...) as probability estimators ([69], [71], [70], [77]).

Here we focus on another approach for combining MLPs and HMMs. Instead of using MLPs as *probability estimators*, we propose to use MLPs as *labelers* for discrete parameter HMMs.

There are other connectionist classification systems which can be also used as a labeler. One of them is the popular unsupervised Kohonen Learning Vector Quantization (Kohonen [36] [37], Iwamida et al. [33], Utela et al.[82]). However, in this work, we focus ourselves on exploring the MLP algorithm and its classification capabilities for speech recognition, i.e., the supervised learning vector quantization.

4.6.2 MLP Labeling

The MLP labeler extents the phonetic frame classification in Chapter 3 to the real speech recognition. Here we use the same MLP which is designed in chapter 3 for the phonetic frame classification. There are three reasons in selecting this structure. First, the TDNN described in section 3.5 is very complicated and not easy to be trained with a large database. Secondly, the Hidden Markov Models will provide time warping to the MLP output, so it is not necessary to keep the TDNN's translation invariant property. And thirdly, the advantages of MLPs, such as high discriminative power, exposing non-explicit knowledge, and including contextual information, can be encouraged by using a large number of frames in the input layer, e.g., five frames or more.

The inputs of the MLP are one or multiple adjacent frames of speech parameters. Each output unit corresponds to a phonetic label. The target value of the output unit corresponding to the current phoneme is set to one, the target values of the other output units are set to zero. Training is done by the error back-propagation algorithm. The training database is obtained by using both random data selection and training data weighting.

Once the MLP is fully trained, the weights are kept fixed. For each frame, the label corresponding to the highest scoring output of the MLP is passed on to the discrete parameter HMM system. Thus we have achieved a supervised VQ which contains phonetic information [53]. The HMMs are then trained using a classical algorithm like the Viterbi algorithm. A graphical representation of the system is shown in Figure 4.8.

In classical HMM training algorithms the models are trained to maximize the likelihood of producing their training examples, but no training is done to minimize some form of probability that other examples are produced by the model. Several researchers have therefore investigated discriminative training methods for HMMs, most notably the *Maximum Mutual Information Criterion* [13], [3] and *Corrective Training* [4] [2]. MLPs, however, incorporate automatically discrimination. In the conventional MLP, used for pattern classification, the number of output units corresponds to the number of pattern classes present. During training, the output unit corresponding to the class of a pattern vector is kept clamped at state 1 while the others are clamped to state 0. Hence the components of the desired output vector take on two crisp state values. During test, a winner-take-all mechanism caused the test pattern to be classified as be-

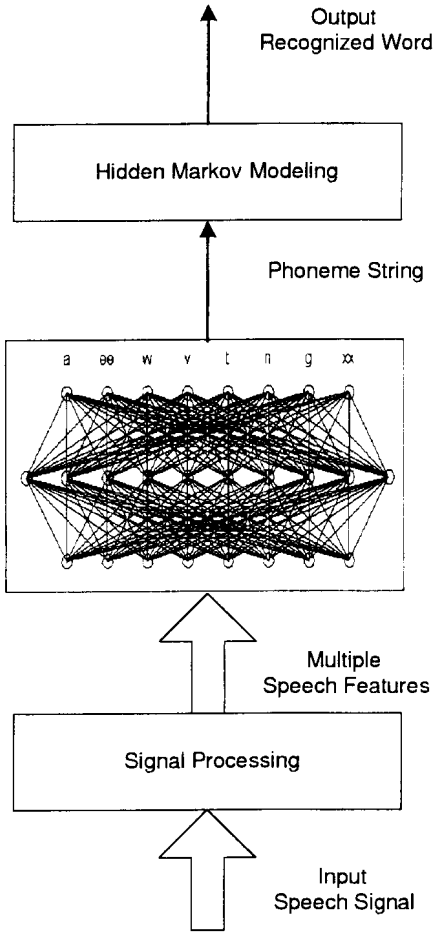

Figure 4.8: A baseline MLP/HMM hybrid speech recognition system

longing to that class corresponding to the output unit with the highest activation. When an MLP is trained for frame classification, it is explicitly demanded that one output is maximal and the other outputs are zero. This provides a discriminating effect.

Our novel labeling approach and its extensions described in the next following sections have several advantages compared to both the classical Euclidean labeling mechanisms and to MLPs used as probability estimators.

4.6.3 Comparison of the MLP Labeler with the Conventional-VQ

When codebooks for an Euclidean distance Vector Quantization (VQ) are designed, the goal is normally the minimization of some distortion measure, e.g., the minimization of a squared error distortion measure [23]. This is a good criterion for speech compression purposes, but for speech recognition purposes it is somewhat overdone. Speech is perceived as a sequence of phonemes, therefore it suffices to know to which phoneme a certain frame corresponds. Indeed, if two frames correspond to the same phoneme then they may be allowed the same label, no matter how close or far they are in the parameter space. MLPs can be trained for phonetic classification and are therefore appropriate labelers for speech recognition systems.

It has been shown that use of multiple features (such as static and dynamic information) can significantly improve speech recognition accuracy (Lee 1989 [43]). One way to incorporate different features into a speech recognition system is to model these multiple features as one vector. Continuous or semi-continuous hidden Markov modeling accommodate different feature representations, but discrete hidden Markov modeling has the independence assumption for the different feature combinations. The MLP non-linear mapping is able to map different features, which may have different physical meanings, or even be strongly correlated, to a domain where the independence assumption is valid.

Figure 4.9 illustrates the frame recognition progress with two cepstral coefficients with the MLP Labeler vs. the conventional VQ for the Dutch digit *negen*. On the left side in Figure 4.9, the frames are for the beginning and the end of speech. The phonetic Euclidean VQ incorrectly labels some of those frames with phoneme *d*, which is not in the *negen* phoneme list. However, the MLP Labeler labels those frames with either *xx* or *n*, which are the phonemes to represent *negen*.

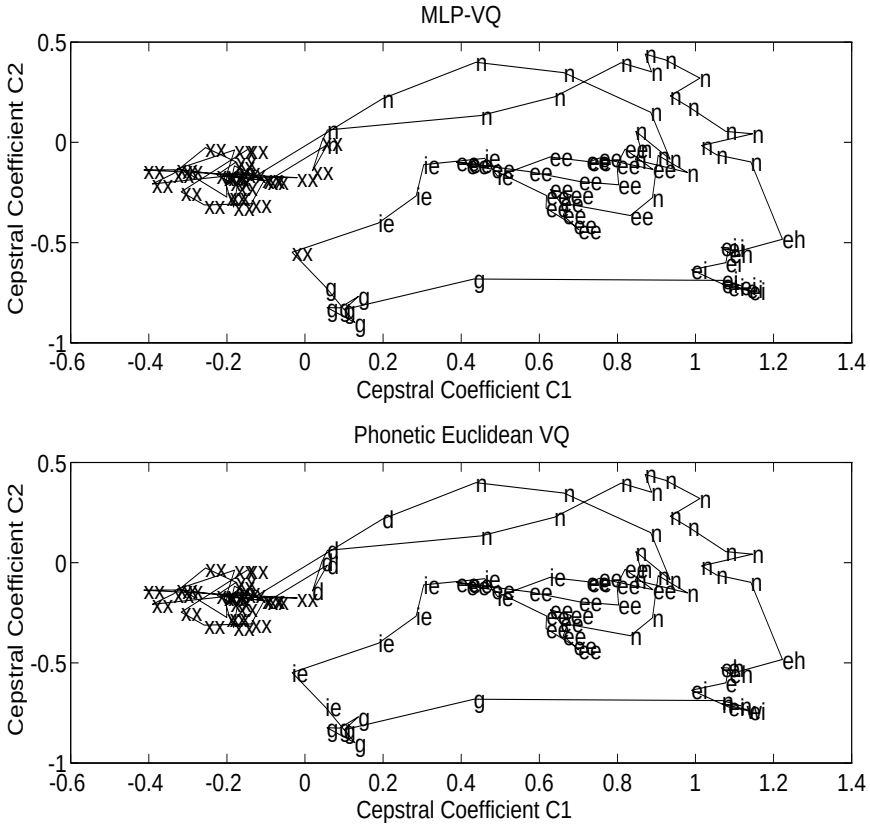

Figure 4.9: The time progress of two cepstral coefficients with the MLP labeler vs. the phonetic Euclidean VQ for Dutch digit *negen*

4.6.4 Comparison of the MLP Labeler with the MLP Probability Estimator

When the MLP works as a probability estimator, it provides estimates of the conditional probability distribution $P(j|\boldsymbol{x}_p)$, the *a posteriori* probability of state j given the input vector $\boldsymbol{x}_p$. Let y_{pj} denote the actual value of the jth output unit produced by the input $\boldsymbol{x}_p$. The target value of the jth output unit is set to one and the target values of the other output units are set to zero.

Using the relative entropy [9] between the *a posteriori* target distribution and the *a posteriori* output distribution as the training criterion, and assuming that the network has enough hidden units and enough training data, and that the training does not get stuck in a local minimum, then the MLP output value y_{pj} approximates the *a posteriori* probability $P(j|\boldsymbol{x}_p)$, as shown by

$$y_{pj} = P(j|\boldsymbol{x}_p) \tag{4.32}$$

Using Bayes' rule, we may then compute the required HMM probabilities as follows:

$$P(\boldsymbol{x}_p|j) = \frac{P(j|\boldsymbol{x}_p)P(\boldsymbol{x}_p)}{P(j)} \tag{4.33}$$

The probability $P(j)$ is the *a priori* probability of state j. It can be estimated by counting the state (class) occurrences in the training data. In the case of class (phoneme) balanced training database, for instance, using our random frame selection method described in Section 3.6, for each training pass an equal number of frames is chosen randomly out of each class. This means that the MLP is trained with equal *a priori* class probability.

The probability $P(\boldsymbol{x}_p)$ is common to all the classes (phonemes) for any given time frame, and may therefore be ignored.

The scaled likelihoods so computed are then used to define an acoustic model for the discriminative HMM. The discriminative training of the HMM involves maximizing the likelihood of the correct state sequence generating the acoustic data, and minimizing the probability of the data being generated by all incorrect state sequences. This requires the update of the initial phonetic segments in the training data.

The challenge of using MLPs as probability estimators is the demand for training a large network towards a global minimum. The hybrid MLP-

HMM systems in $SRI-DECIPHER^{TM}$ need more than 150,000 weights! [69] (Our baseline MLP labeling/HMM system only needs less than 3,000 weights.) Training this kind of network clearly exceeds the capabilities of most current digital computers. Huge amounts of training data are needed to have a generalized distribution. Moreover, MLPs are most often trained for phonetic classification. This implies a restriction to HMMs of which every state corresponds to a phoneme (*phonetic* HMMs). Although for large vocabulary speech recognition phonetic models are the only alternative, many small vocabulary applications use word models instead of phonetic models. The reason is that word models have the advantage of modeling directly whole words, as the word model is able to overcome coarticulation problems. Hence, word models have better recognition performance than phonetic models.

Based upon the above comparisons, our MLP labeling approaches have shown many advantages over the other approaches in constructing hybrid discrete HMM systems.

4.7 Hybrid MLP VQ/HMM Systems

We have presented a baseline MLP VQ/HMM system in Figure 4.8. From the baseline system, we develop several MLP/HMM hybrid systems in parallel with the conventional vector quantization. The new systems are able to adopt the modified *forward and backward* algorithm and the modified *Viterbi* algorithm for HMM training and test used for hybrid VQ/HMM systems in Section 4.3.

4.7.1 MLP Multi-Dimensional Labeling

We are motivated by the histograms of MLP output values and are seeking different ways to use output information. At first, a simple way to incorporate the information of other output units into the system is to use not only the top scoring output, but the N top scoring outputs as labels. For instance, if $N = 2$, then 2 labels are passed on to the HMM system. These labels correspond to the highest and the second highest output of the MLP. This method is called the *Top-N* method (Ma and Van Compernolle [54]). It has the advantage of using information from N output units instead of one. There are now N label streams from the MLP to the HMM instead

of one. This way the MLP input parameter space is much more finely described. The HMM can then use these labels as independent observation variables, just as labels from multiple codebooks [24].

But there is a disadvantage of the top-N method: the assumption that the labels are independent is invalid. It is clear, for example, that the first and second label can never be the same. Although our experimental results will show some improvements when using the top-N method, this theoretical disadvantage led us to investigate another method which uses information from other than the top scoring outputs.

Then we proposed our second top-N method (Ma and Van Compernolle [54]). The idea was to give every combination of highest and the second highest scoring MLP output a separate label. This means that the number of possible labels would now be equal to the square of the number of phonemes. For a vocabulary with 22 phonemes, for example, this means that the number of possible labels would be 484. Normally, this is too much, and the label probabilities need large amounts of training data to be estimated properly. In their work, Le Cerf and Van Compernolle [41] described a modified Top-N method. It reduced the number of possible labels by taking the most occurring second choices since in practice only a few MLP outputs other than the highest scoring output have values significantly different from zero. The equivalent result was produced. However, theoretically, the independent assumption is still invalid for both methods as using different labels does not change the parameter dependency. We still need to seek other strategies to take advantage of the rich information present in the neural network output. We group all three methods into one category: the *MLP multi-dimensional labeling* as they all need the HMM with multiple codebooks. The recognition system is shown in Figure 4.10.

4.7.2 MLP Multi-Labeling

We are inspired by the multi-labeling method in the conventional K-means VQ we have described in the previous section. Here we propose the fourth approach: the *MLP multi-labeling*, in which the HMM independence assumption is met as we only need one codebook instead of forming multiple codebooks as in the above three efforts.

We calculate the statistics from the MLP outputs as follows. As we know, the MLP output values are between $[0, 1]$. We can omit the normalization step used in the conventional K-means multi-labeling.

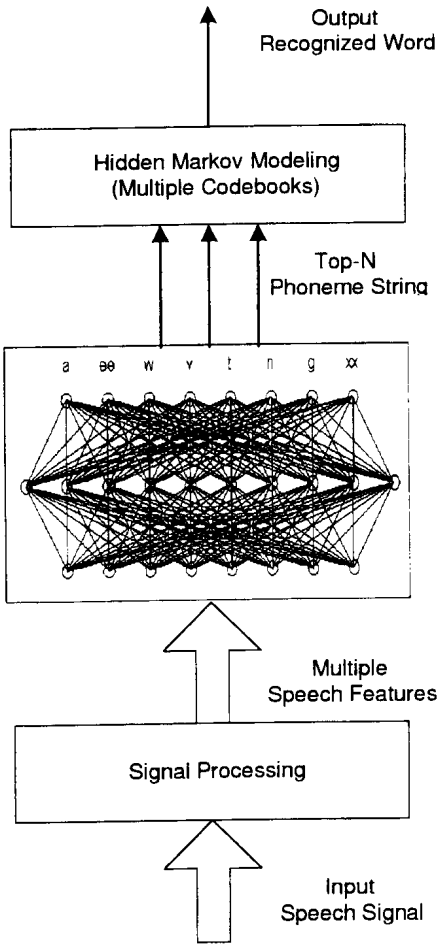

Figure 4.10: An MLP multi-dimensional labeling/HMM system

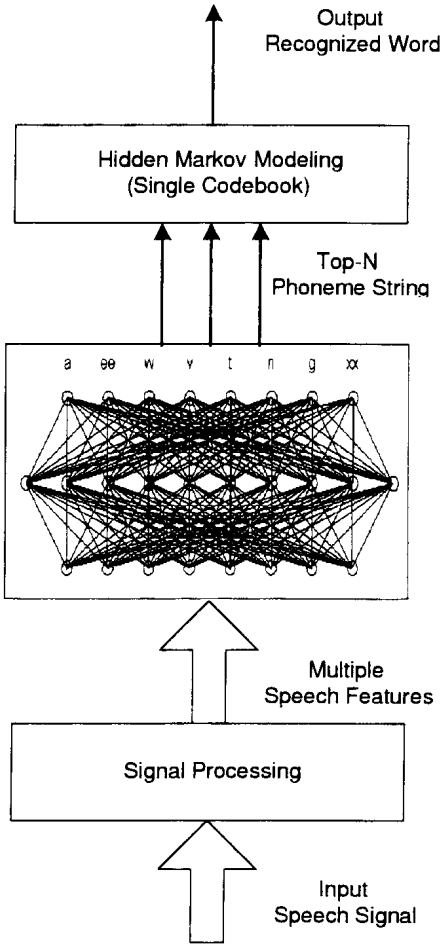

Figure 4.11: An MLP multi-labeling/HMM system

Let $\boldsymbol{y}_p$ denote the MLP output for the input speech parameter vector $\boldsymbol{x}_p$.

$$\boldsymbol{y}_p = (y_{p1}, y_{p2}, ..., y_{pL}) \tag{4.34}$$

Where L is the number of units in the MLP output layer. Then the observation probability is calculated as follows:

$$w_i(\boldsymbol{x}_p) = \sum_{j=1}^{L} f(\boldsymbol{x}_p|j) b_i(j) \tag{4.35}$$

$$f(\boldsymbol{x}_p|j) = \alpha \quad if\ y_{pj} > \eta, \quad else\ f(\boldsymbol{x}_p|j) = 0 \tag{4.36}$$

Where η $(0 \leq \eta \leq 1)$ is a preset threshold, α is a constant value and $\sum_j^L f(\boldsymbol{x}_p|j) = 1$ is constant for every frame. The system configuration is illustrated in Figure 4.11. Although this Top-N method has better theoretical support than the other Top-N methods, it still does not fully use the MLP output information because it treats each selected output equally. To solve this problem, we propose the fuzzy labeling in the later section.

4.7.3 Multi-MLP Labeling

This scheme was proposed by Le Cerf and Van Compernolle [41] based on the method we have described in section 4.5.4.

Instead of training one MLP for phonetic classification with an input of all parameter sets (Ma et al[53]), three independent MLPs are trained for phonetic classification. The first has one frame of basic parameters as input, the second the corresponding set of derivative parameters, and the third the set of second derivatives. Typically, LPC based cepstra were used as MLP input parameters. In that case, a fourth MLP can be trained with energy related parameters (the first and second derivative of energy *den* and *d2en*). So instead of using multiple frames as inputs of an MLP, derivative information is represented more directly by several MLPs.

During recognition, the MLPs work in parallel. For each network, the label corresponding to the highest scoring output provides one label (winner-take-all). Because the parameters of only one frame are used as input for each MLP, the outputs of the networks can be assumed independent. Therefore, the probabilities associated with the labels from the different MLPs can be multiplied in the same way as in Equation 4.26. This corresponds to using multiple codebooks based on different parameter sets [24]. The system configuration is shown in Figure 4.12.

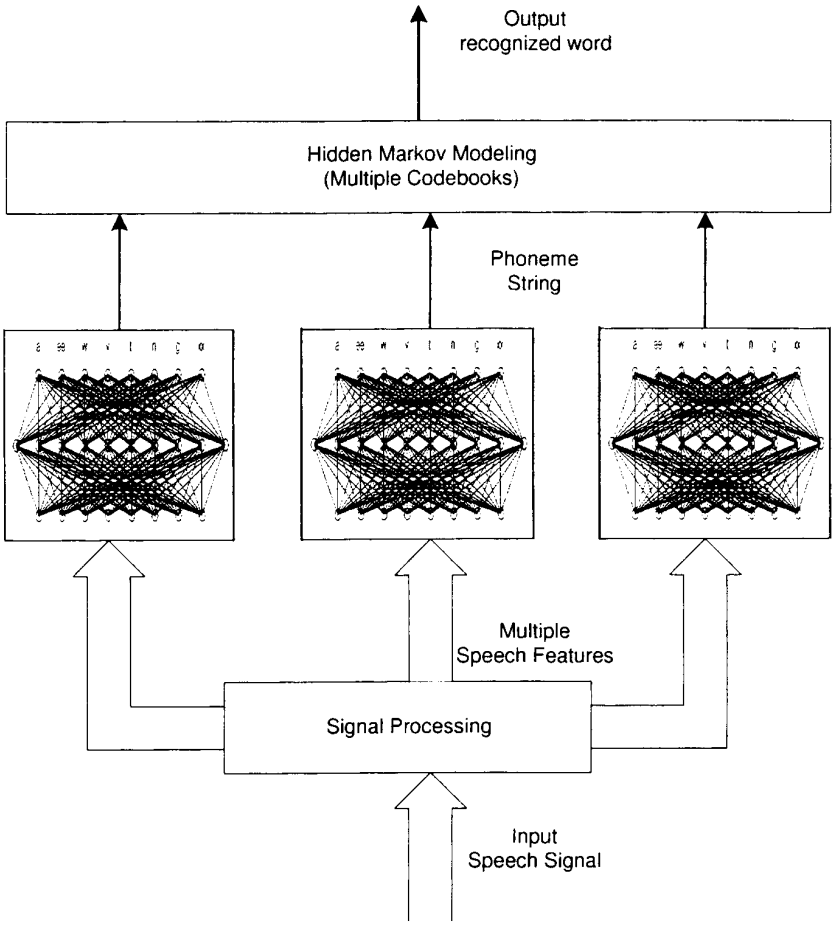

Figure 4.12: A multi-MLPs/HMM hybrid system

4.7.4 Multi-MLP Multi-Labeling

Following the conventional K-means multi-dimensional multi-labeling, here we propose the *multi-MLP multi labeling* method. Each MLP has a multiple labeling output and the computation is in parallel. Because the number of MLP output units is very small (less than 30 in our case) compared with the total size of K-mean codebooks (more than 400), multi-MLP multi-labelers significantly reduce the amount of computational load for HMM modeling compared with the K-means counter part. Figure 4.13 illustrates the system configuration.

4.7.5 MLP Fuzzy Labeling

In chapter 3, we have described the relations between the MLP input and its target output for the MLP training. Here we group them into three categories.

- the normal input and the crisp output;
- the weighted input and the crisp output;
- the normal input and the fuzzy output (weighted output).

Figure 4.14 illustrates the MLP output values for Dutch digit *een* using those three kinds of input-output combinations. The MLP output values are between zero and one but not even close to the binary values for all three categories. This implies that the MLP itself is able to learn the fuzzy nature of the speech signal and the information is encoded among the various connection weights in a distributed manner even in the case of the crisp target output. Our goal is to use the MLP output value as the fuzzy score to measure the closeness between the input pattern and each of output classes (phoneme). For this reason, we use the Fuzzy MLP to present our method. Hence, the method we propose here also applies to the other two cases.

We still use our 3-layered baseline MLP structure shown in Figure 3.10. In a Fuzzy MLP, the speech input parameter vector consists of membership values to linguistic properties while the output vector is defined in terms of fuzzy class membership values at the corresponding outputs. During training, the input spectra are over-lapped in time and randomly selected; the Hamming window function provides the fuzzy membership function to

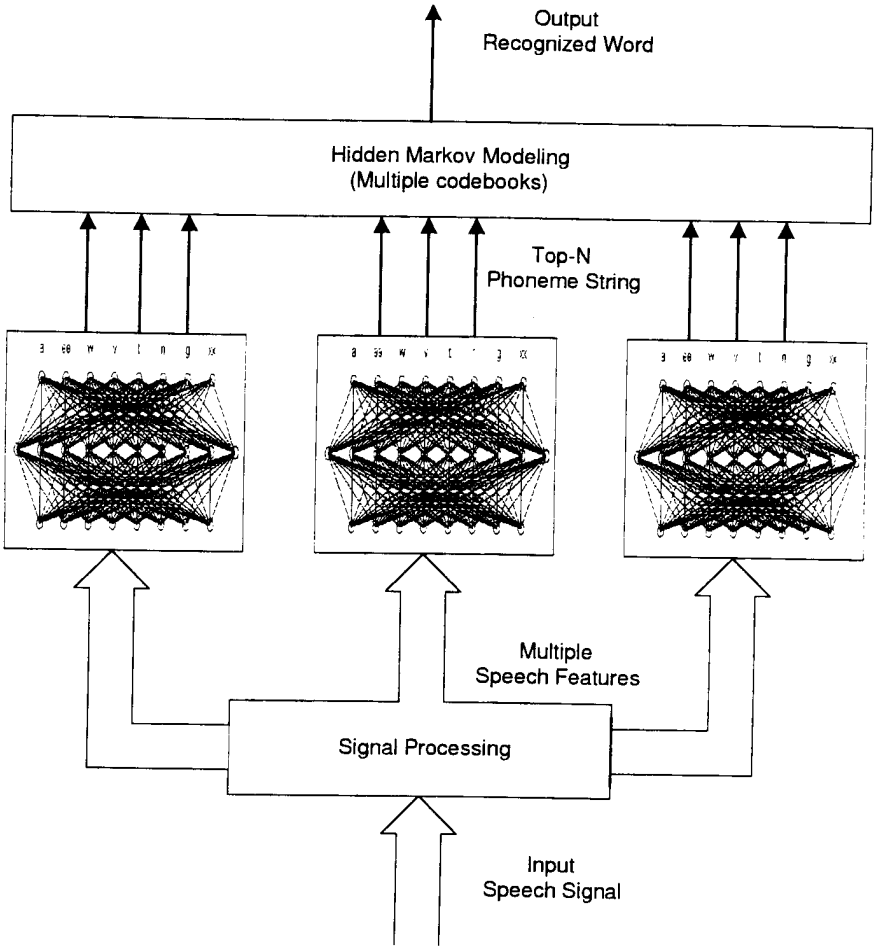

Figure 4.13: A multi-MLP multi-labeling/HMM system

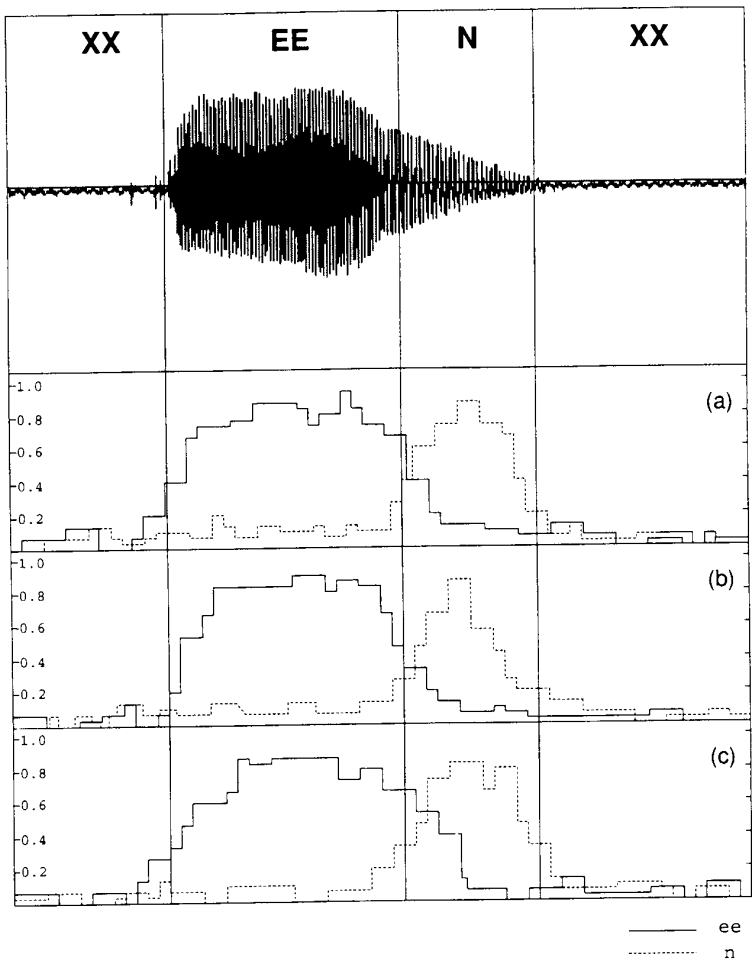

Figure 4.14: The MLP output for Dutch digit *een*: (a) the normal input and the crisp output, (b) the weighted input and the crisp output, (c) the normal input and the fuzzy output.

generate the MLP target output. During MLP labeling, the analog output values between [0,1] are produced from the MLP output layer.

Those analog MLP outputs are interpreted as mass probability. When the observation is fuzzy, the observed sequence is a sequence of probability mass vectors. Denote this fuzzy observation sequence by $Y = (\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_P)$, where each $\boldsymbol{y}_p$ is now a probability mass vector of the form $\boldsymbol{y}_p = (y_{p1}, y_{p2}, ..., y_{pL})$ as described before. An MLP Fuzzy VQ/HMM speech recognition system is shown in Figure 4.15. The modified Viterbi algorithm is used for HMM modeling during both training and test.

MLP Output Normalization

For the MLP network, the value of each output will in any case remain between zero and one because of the sigmoidal function which is typically used in speech processing and recognition. Network outputs should sum to one for each input value if the output accurately estimates posterior probabilities (Bourlard et al [10]). However, if the network converges to a local minimum, it is no longer guaranteed that the network outputs estimate Bayesian probabilities. In practice, this is always true. Besides, if the target output is calculated from the fuzzy membership function during the supervised learning, the network outputs have very little chance of summing to one.

However, for the normalization purpose and the mass probability constraints, we need to normalize MLP outputs. For each input pattern $\boldsymbol{x}_p$ we rescale the top-N MLP output values and sum them to one and reset the rest of the output values to zero. We then obtain an output vector which can be interpreted as a probability mass vector $\boldsymbol{y}_{pi} = (y_{p1}, y_{p2}, ..., y_{pL})$, with only those y_{pi}'s corresponding to the top-N outputs nonzero. The mass probability is computed as follows:

$$f(\boldsymbol{x}_p|j) = \frac{y_{pj}}{\sum_{i=1}^{L} y_{pi}} \tag{4.37}$$

4.7.6 Multi-MLP Fuzzy Labeling

Finally we propose our last hybrid VQ/HMM model, the *multi-MLP fuzzy labeling*. It combines the multi-MLP labeling and MLP fuzzy labeling together. The layout of the system is illustrated in Figure 4.16.

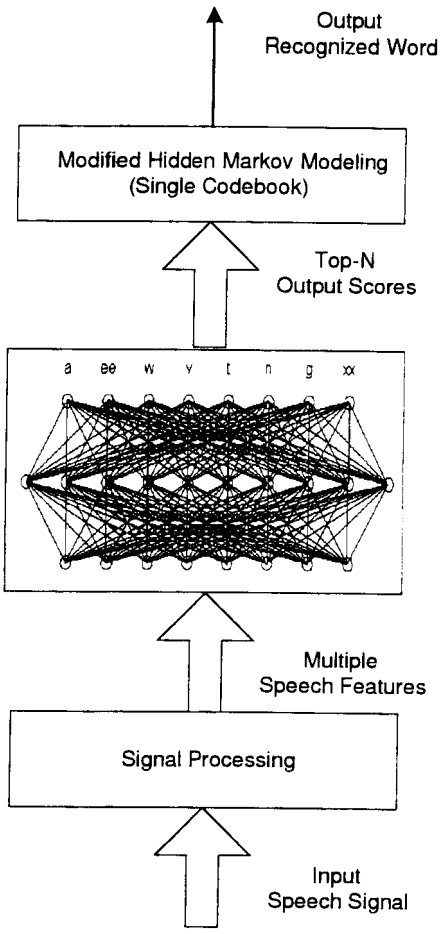

Figure 4.15: An MLP Fuzzy Labeling/HMM system

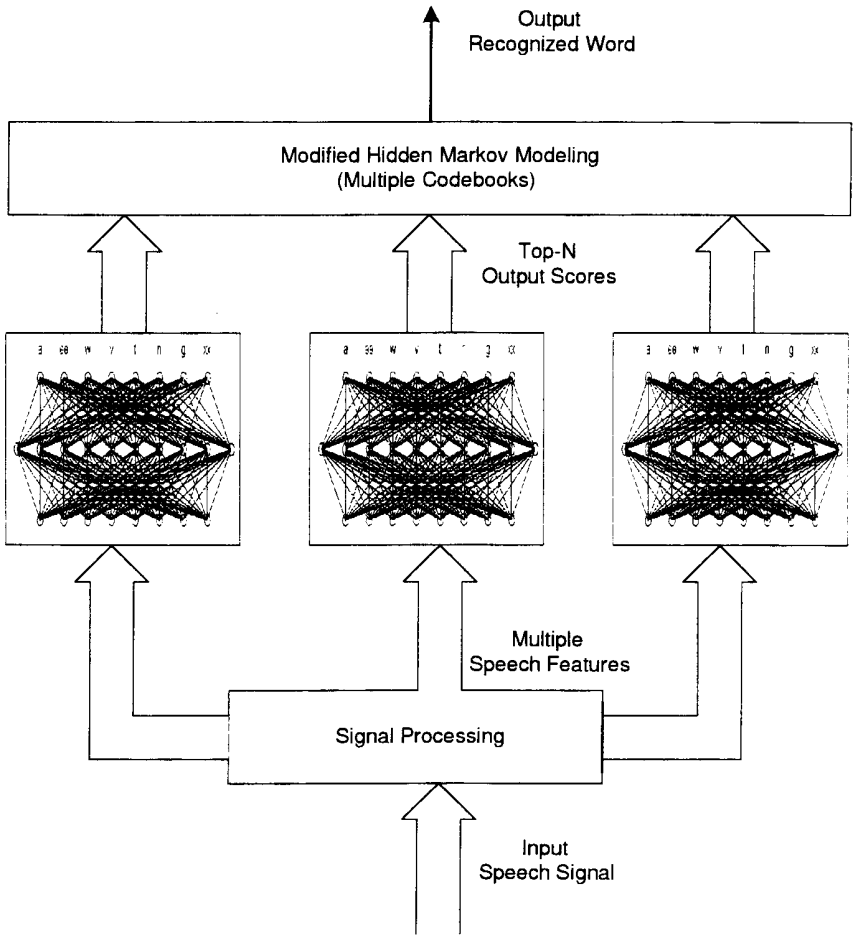

Figure 4.16: A multi-MLP fuzzy labeling/HMM system

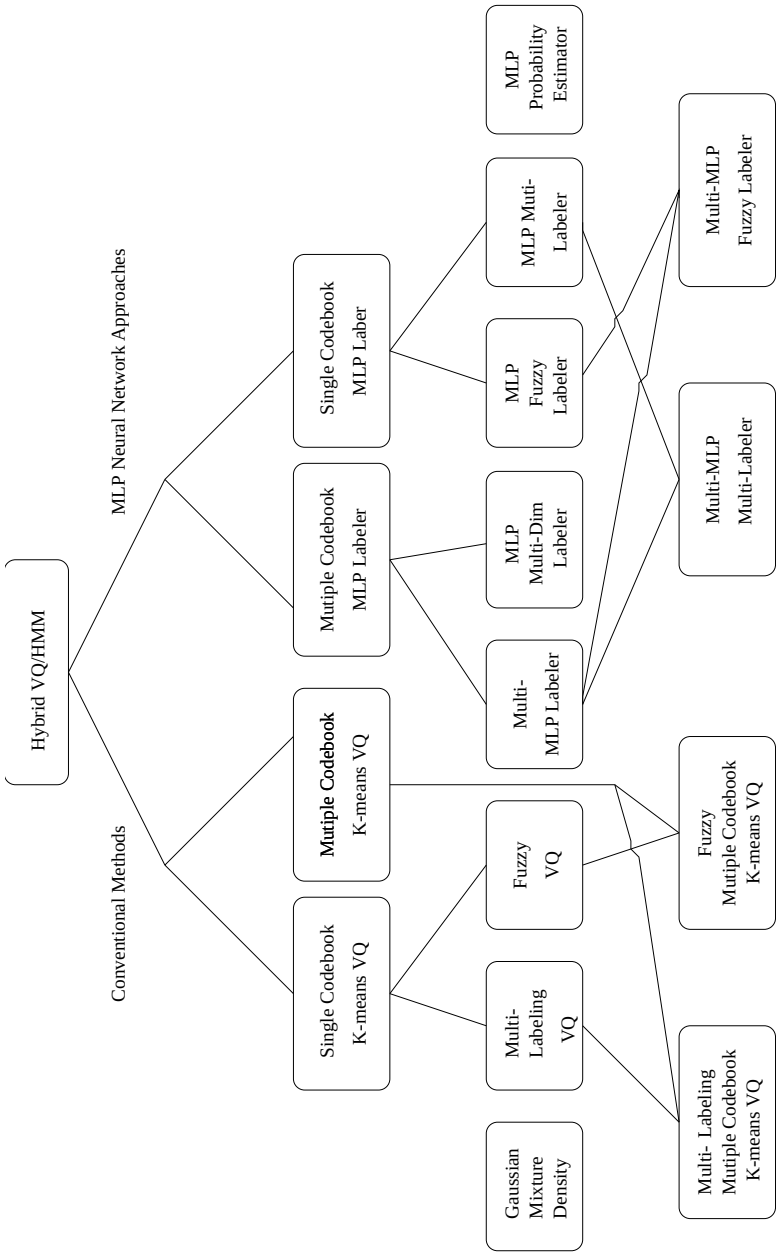

Figure 4.17: Summary of hybrid VQ/HMM techniques

This method is applicable to both training and decoding of an HMM. However, it has the disadvantage of requiring much more training and recognition time, and more memory, compared to any other MLP method. An efficient compromise is to use the multi-MLP fuzzy labeler during training and to use the multi-MLP labeler during recognition.

4.8 Summary

Figure 4.17 illustrates the relations between different hybrid VQ/HMM techniques investigated during this work. We have discussed a novel combination of Multilayer Perceptrons (MLPs) and Hidden Markov Models (HMMs). From the conventional K-means VQ to the MLP labeler, we have presented MLP/HMM hybrid models corresponding to each K-means hybrid VQ/HMM model, and even more.

A comparison has been made between the standard VQ and the Multi-Layer Perceptron labeler. Our MLP labelers have the advantage of needing fewer HMM parameters per state and of obtaining a higher recognition accuracy. Compared to the probabilistic interpretation of MLPs, our MLP approaches need fewer hidden units and less training time, and have flexibility in system design (e.g., use HMM word models instead of HMM phonetic models). We first proposed the baseline MLP labeling, then we developed several improved MLP/HMM models from the baseline model. We designed several multi-dimensional Top-N labeling, multi-labeling and fuzzy HMM hybrid models with a single MLP. Using multiple MLPs, we developed the multi-MLP multi-labeler and the multi-MLP fuzzy labeler for HMM systems.

The experimental results and the detailed system design will be presented in the next chapter.

Chapter 5

Implementation and Experimental Results

5.1 Introduction

In this chapter, we describe the experiments we performed to investigate neural networks for small vocabulary speech recognition. The experiments were carried out on our speech processing system with SUN/UNIX Sparc workstations.

This chapter is structured as follows: Time Delay Neural Networks are described in the next section; in Section 5.3, we discuss the experimental environment, which includes vocabulary, database, signal processing, and hidden Markov modeling; the MLP labeler implementation is discussed in Section 5.4, where we also discuss the MLP implementation issues such as MLP structure, the back-propagation training and test; the MLP multi-dimensional labeler is discussed in Section 5.5; in Section 5.6, we present the MLP fuzzy labeling/HMM system; Section 5.7 contains our summary of this chapter.

5.2 Time Delay Neural Networks

Our first experiment is done based on the milestone paper written by Waibel et al [90]. This is our first experiment using neural networks. The purpose of this experiment is to see how neural networks can be used for speech recognition.

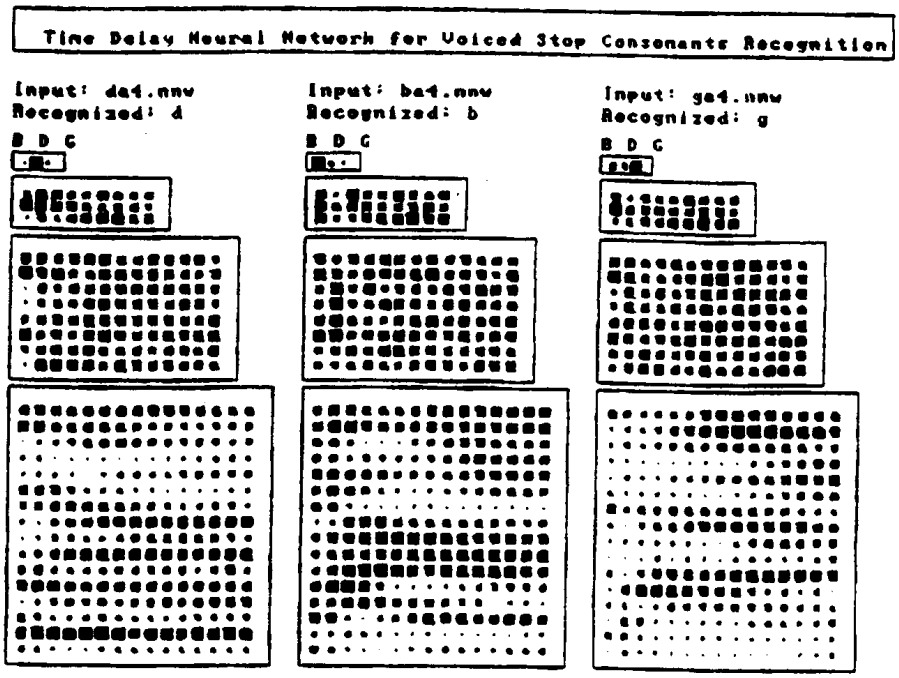

Figure 5.1: TDNN output with input CV syllables [*da,ba,ga*]

5.2.1 Vocabulary and Database

Using the three voiced stops [*b, d, g*], speaker-dependent data was established with six CV syllables: [*ba, bo, da, do, ga, go*], in English. It was sampled at 20 kHz. Each syllable was recorded 10 times. Half of the data is used for training and the other half for testing. Speech signal preprocessing is based on Mel-spectral filter bands. The Mel-spectrum is further normalized by the data normalization method described in the next section.

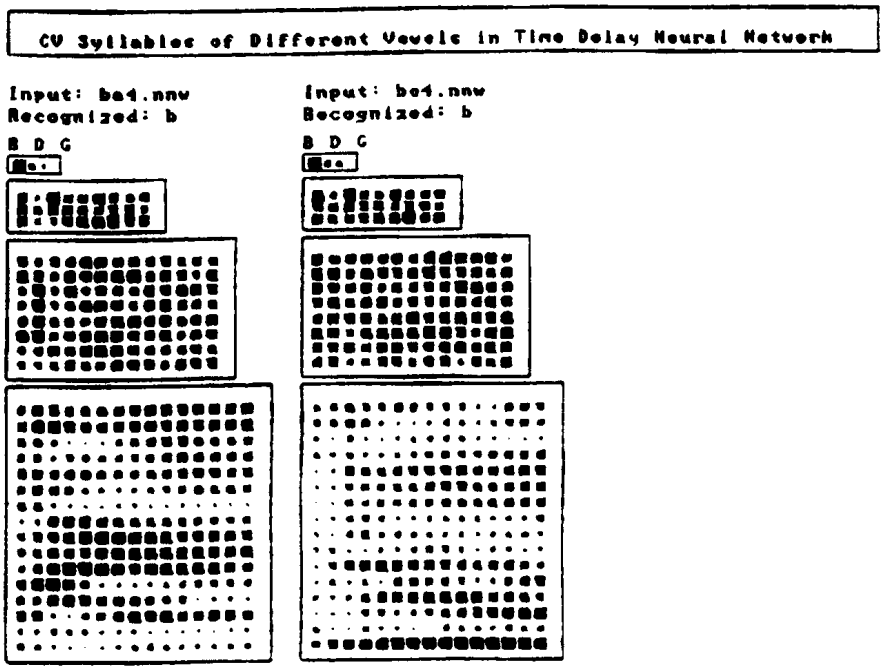

Figure 5.2: TDNN output with input CV syllables [*ba,bo*]

5.2.2 TDNN for Phoneme Recognition

Figure 5.1 illustrates a TDNN trained to perform the discrimination task between the voiced consonants [*b, d, g*]. A four-layered network is constructed with Waibel's weight-tying method. Eight hidden units in the first hidden layer are fully interconnected with a set of 16 spectral coefficients. Each of these eight units in the first hidden layer produces patterns of activation as the window moves through input speech. A five-frame window scans these activation patterns over time then activates each of three units in the second hidden layer. These activations in turn are then integrated into one single output decision. The final decision is based on the combined acoustic evidence, independently of where in the given input

interval (15 frames or 150 msecs) the [*b, d, g*] actually occurred.

The inputs, 16 × 15 mel-spectra, are normalized between [-1,+1]. No pre-selection of tokens was performed. All tokens labeled as one of the three voiced stops are included. The consonant tokens are extracted from the entire set of utterances. A significant amount of acoustic variability is introduced by the phonetic context and the token's position within the utterances. The darkness of the dots stands for the absolute output values of each unit in the following figures. These output values are positive in all layers except the input layer where nearly half of values are negative because of the normalization.

5.2.3 Results and Discussion

In this preliminary experiment, the phoneme recognition rate is 98.5% for training data and about 97% for test data. In Figure 5.1, the three voice stop consonants are recognized with the same vowel. Figure 5.2 shows the same consonant with different vowels. Comparing the second hidden layers in Figure 5.1 and Figure 5.2, we found that the second layers are almost the same in Figure 5.2 but quite different in Figure 5.1. This means that the higher layer can extract more important features of speech than the lower layer since the consonants are more important than vowels here for consonant recognition. However, these figures also illustrate that it is possible to use one hidden layer instead of two because significant information is carried from the second hidden layer to the output layer, while the first hidden layer does not extract as much speech features as the second hidden layer.

Because of the weight-tying strategy, Waibel's TDNNs have time invariant capability. However, this time invariant capability is valid only within the length of time delayed period. Since the window length of the input speech signal is fixed, the TDNN has its own limit to handle the uncertain length of speech signals. In the following experiments, we introduce the MLP/HMM hybrid system to solve this problem.

Table 5.1: The Dutch digit phonetic transcription

Digit	Transcription
nul	xx n u l xx
een	xx ee n xx
twee	xx t w ee xx
drie	xx d r ie xx
vier	xx v (i) ie r xx
vijf	xx v ei f xx
zes	xx z e s xx
zeven	xx z ee v eh n xx
acht	xx a ch t xx
negen	xx n ee g eh n xx

5.3 Experimental Environment

5.3.1 Vocabulary

We use the 10 Dutch digits in our experiments. The phonetic transcription is shown in Table 5.1. Although digits are a small set of words, they contain 20 Dutch phonemes out of a total of about 40. Thus, there are 21 phonetic classes used in our experiments, including noise, which we represent by *xx*. The most confused words among Dutch digits are *vijf* and *zes*, as both words start and end with fricative consonants and, moreover, consist of a single vowel who is neighbor in the vowel triangle.

Digit recognition has been widely used in speech recognition applications. Very often, it is used for recognizer accuracy evaluation. The effort devoted to digit recognition is greater than any other group of words during recognizer development. Most of successful commercial products have special digit recognition models designed to increase the recognition accuracy.

5.3.2 Speech Database

Three sets of databases are used in the following experiments. Our speaker-independent database is a small vocabulary isolated word telephone database. It consists of ten Dutch digits uttered by 400 speakers (200 males and 200

females). The Dutch digit set contains 21 different phonemes. Half of the data (100 males and 100 females) is used for training the HMM parameters and the MLP connectionist weights, and the rest for testing. The data was sampled at 8 kHz because the bandwidth of the telephone filter is about 3.3 kHz. In the following experiments, we refer to this database as the "large database".

Within the large database, there is a small set that is phonetically hand-segmented. This small set includes 40 speakers (20 males and 20 females). We call this the "small database".

We also used some locally collected data with multi-speakers during our initial MLP labeling experiment. We call this "multi-speaker data". The multi-speaker data contains 6 speakers (3 males and 3 females).

5.3.3 Speech Signal Preprocessing

The speech signal preprocessing is based on Mel-spectral filter bands. As we use a single MLP in the following experiments, the Mel spectrum itself is enough for our speech recognition tasks. The typical sequence of operations is as follow:

- Sampling at 8 kHz;
- Applying the Hamming window with a window length of 30 msec (240 samples);
- Using a 10 msec (80 samples) frame shift;
- Calculating 256 points FFT;
- Computing the summation of energy values into 15 contiguous bands spanning the frequency range from 200 Hz to 3125 Hz;
- Converting to log-energy parameters;

5.3.4 Database Initialization

The input Mel-spectra are further normalized between $[-1, 1]$. This is done for two reasons. First, removing the variations in the overall magnitudes of the spectral coefficients may enable the network to focus on the linguistic

information in the speech signal, such as the formant frequencies. Second, Burr (1988) [15] found that the classification performance typically improves when the mean of the input vector is close to zero. Moreover, the normalized values are near the transition regions of the sigmoid function, where learning is faster than it is in the saturation regions. However, the overall magnitudes of the spectral coefficients can't be removed, since they often contain relevant linguistic information.

We propose the following solution to keep the magnitude information. First, let the set $\vec{s}$ denote the set of training samples, $bmfs = [s_1, s_2, ...s_M]^t$ is the mel-spectral vector for a phonetic token. Then, let $\boldsymbol{x}$ denote the set of normalized training samples, $\boldsymbol{x} = [x_1, x_2, ...x_M]^t$. Furthermore, let

$$x_j = \frac{s_j - \mu_j}{D_j} \quad (1 \leq j \leq M) \tag{5.1}$$

Where $D_j = max(s_j) - min(s_j)$ and μ_j is the mean of s_j over all training tokens. D_j and μ_j are constants for the given training tokens and also used for test data. The normalization is done component by component.

5.3.5 Hidden Markov Models

We use the ESAT HMM software package (Van Compernolle[87]) in our MLP/HMM hybrid system experiments. The package is developed by the ESAT speech group. The speech recognition algorithms described in Chapter 2 have been built in the package. Our experiments follow the package growth from its starting version up to the mature version 4.0. The following features of our speech recognition system are used in our experiments:

- speaker-dependent/independent or multi-speaker,
- isolated word recognition with phonetic models and word models,
- multiple codebook discrete parameters and semi-continuous parameters,
- Viterbi search for both training and recognition,
- small vocabulary,
- graphic demos with *spchlab*.

More detailed information regarding the HMM package is described in Van Compernolle [87]. We developed a speech interpreter, namely, *spchlab* during this work. *spchlab* is interactive, X-window based and speech-oriented. It uses the same principle as the popular *matlab* does. The detailed description for *spchlab* is in Ma [55].

5.4 Multi-Layer Perceptron Basics and MLP Labeling

The following experiments deal with the techniques to build MLP networks suitable for speech recognition tasks. We discuss how to choose the number of hidden units, how to scale the network, how to utilize the speech data to train the MLP network and how to build the MLP/HMM hybrid system.

5.4.1 Frame Classification

First of all, we need to decide on the network structure. As we have pointed out in Chapter 3, a three-layered network is good enough for our purposes. Therefore, we use three layers in our MLP networks. Now we need to choose the number of units in each layer. As we know, using the difference of the spectrum at time $t+2$ minus the spectrum at time $t-2$ is a simple way to include knowledge about spectral transitions. Hence, we use five frames in our input layer, i.e., $5 \times 15 = 75$ units. Our database covered 21 Dutch phonemes (including noise), so the output layer has 21 output units corresponding to each of the 21 phonemes. Hence, the purpose of this experiment is to determine the optimal number of units in the hidden layer. The network's capability to capture the underlying characteristics of the input data can be affected by the number of hidden units. The optimal number of units in the hidden layer should optimize the classification accuracy.

The classification accuracy is a function of both numbers of hidden units and the amount of training data. With the inclusion of more hidden units, performance typically improves; however, performance tends to degrade when there are too many hidden units. As long as the number of hidden units is reasonably chosen, increasing the size of the training set typically improves the performance of the network. According to Mirchandani et al (1989) [58], the number of hidden units in a feed-forward neural network

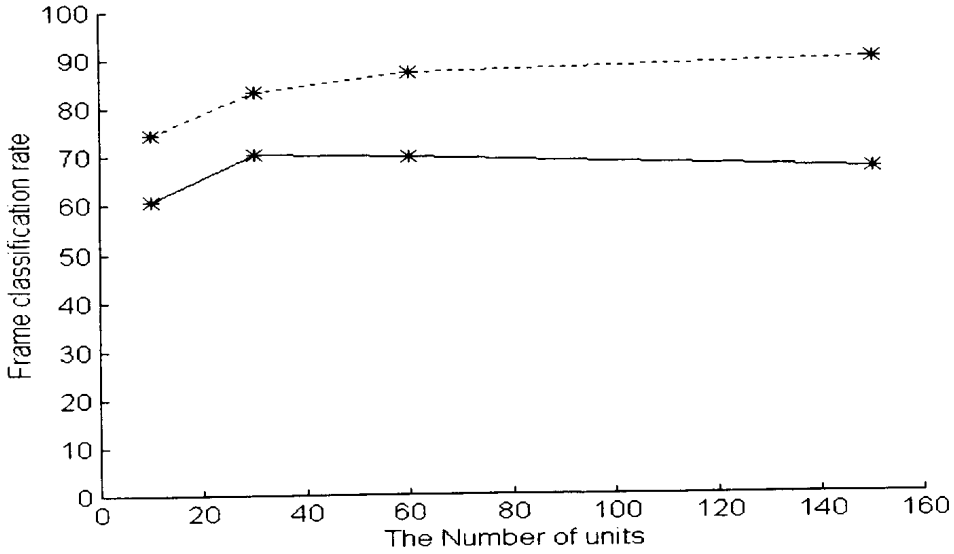

Figure 5.3: The results of frame classification vs. the number of hidden units (the dotted line for training and the solid line for testing).

depends on the number of input training patterns T. It can be chosen from $\log_2 T$ to $(T-1)$. This region is too wide to decide upon a suitable number of hidden units.

The total number of network connectionist weights, including biases is $(n_i+1)n_h+(n_h+1)n_o \approx (n_i+no)n_h$, a linear function of n_h. Here n_h, n_o and n_i are the number of units in the hidden layer, the output layer and the input layer respectively. The network will grow very fast by increasing the number of units in the hidden layer. Training the large network will take a long time and need large memory.

In our experiment, the number of hidden units varies from 10 to 150. The training and test results are illustrated in Figure 5.3. As we use the small database in this experiment, *cross validation* is not necessary. Our experimental results show that the performance on the training data constantly improves as the number of hidden units increases, whereas the performance on the test data improves progressively more slowly as the number of hidden units increases. When there are too many hidden units, the network will be too flexible and as a result will capture irrelevant information

from the training data. This will reduce the recognition accuracy for the test data. Figure 5.3 shows the reasonable size of the hidden layer is $30-60$ units for the present database. In the following experiments, we fixed the size of hidden layer at 30 units.

5.5 MLP Labeling/HMM Hybrid System

In this experiment, we use the multi-speaker database. The main reason to use this data is to describe the Viterbi alignment. The goal of this experiment is

- providing the step by step implementation of the MLP/HMM hybrid system,
- presenting the automatic segmentation method by using the Viterbi alignment,
- giving a primary experiment for scaling the network from a small one to a large one.

The whole system training consists of three parts:

- the HMM training for phonetic segmentation,
- the MLP training for labeling,
- the second HMM training for speech recognition.

5.5.1 HMM Training for Phonetic Segmentation

Since the connectionist networks require an extremely large labeled training corpus, and since the training of our system requires that an utterance be labeled at the frame level (windowed frames), it is necessary to use an automatic alignment procedure to generate initial labels for supervising the neural network targets for the training set. The Hidden Markov Models are trained on this task first, and a forced alignment procedure is used to generate phoneme or noise (silence) labels for each frame of speech in the training database.

Of course phonetic labeling with HMMs is far from perfect, but on its own training database it is quite good indeed. Furthermore no exact

phoneme recognition is required as they are used to supervise the MLP training procedure and any reasonable selection of classes should suffice.

Our hidden Markov model system is a discrete parameter system that uses a VQ codebook of size 200 and 21 phonetic left-to-right phonetic models. The Viterbi style training is run over a sufficient number of passes to obtain full convergence.

5.5.2 Building the MLP Training Database

First, a Viterbi alignment against the computed phonetic models is used to produce a phonetically labeled database frame by frame. This procedure produces 8,000 representations of 21 phonemes in total. The frames at the boundaries of each phonemic segment are deleted in order to avoid labeling errors around phonemic boundaries. It is considered preferable to exclude the phonemic transitions from the MLP training. The remaining frames are used in an overlapping fashion like multiple realizations of a phoneme in the MLP training database. This procedure reduced the number of representations of the 21 phonemes by 5,000, or on average 250 representations per phoneme.

5.5.3 MLP Training

Our MLP consists of 3 layers (an input layer, a hidden layer, and an output layer). The input layer has 75 units (5 frames $\times$ 15 spectral coefficients). The hidden layer has 30 units. And the output layer has 21 units corresponding to 21 phonemes as we described in the previous section.

The input data is normalized between $[-1, 1]$. The training algorithm used is the widely applied back propagation algorithm. The connection weights are randomly initialized to ± 0.3. As we do not use the random frame technique here, all training samples are presented to make the weight adaptation. In order to do the network scaling, the training is further divided into three stages:

- A short training with one representation per phoneme occurrence in a small network,
- A short training with all training material in the small network,
- A full training with all training material in the scaled network.

First a small network (3×15 units at input layer) is initialized. The training then starts from the random initial weights with one representation per phoneme. After 20 iterations, all the training data are applied and the training goes on. After 50 iterations, the network is rescaled by adding one frame at both sides of the small network in the input layer and adjusting weights in the way described in Section 3.4. The training of the enlarged network starts from those redistributed weights using all training data. The training stops after 2,000 iterations. The values of connectionist weights are stored every 100 training iterations for later evaluation.

Here, the network first learns some initial knowledge to find the good convergence direction to save searching time with small training data in the initial stage. It is very difficult to converge to scaling the network after the initial stage because the small network reaches its own local minima which are quite different from the local (global) minima of the enlarged network.

5.5.4 MLP Labeling

The most straightforward implementation of MLP labeling is to use the phonetic label corresponding to the highest analog output value as a discrete output, i.e., using a winner-take-all strategy. Because of the sigmoid function, the analog outputs are in [0,1]. Empirically, the confidence level is 0.5 for the highest analog output.

5.5.5 Second HMM Training

In the second HMM training, word models or phonetic models are both applicable. In this experiment, we choose phonetic models. The same HMM phonetic models and training methods are used as in the original Viterbi segmentation. However, instead of Euclidean VQ with a codebook size of 200, we use the MLP labeler of size 21.

5.5.6 Discussion and Results

The purpose of the MLP labeler is to find a way to improve on standard vector quantization for HMM recognizers. First of all, we have compared the 21 parameter MLP labeler with an Euclidean VQ with the same codebook size. Results for the MLP labeler are 93% correct vs. only 90% correct on the Euclidean VQ. Hence, the intrinsic acoustic phonetic distortion pro-

vided by the MLP labeling must be significantly smaller than the distortion provided by Euclidean labeling.

Figure 5.4 shows the MLP outputs, Viterbi segmentation results and internal activations at frame 20. The size of the black spirals stands for the output value of the MLP. The largest spiral in each column corresponding to the MLP output label which is indicated by the corresponding row index. It is obvious that Viterbi alignment provides a very good phonetic segmentation here as the output phoneme labels of the MLP match the real speech signal very well, especially for vowels.

The training and test results for each word are shown in Figure 5.5. The reference models are indicated by the row indexes and the tested words by the column indexes. Since the test set is smaller than the training set, the size of the spiral is smaller in the test figure than in the training figure.

5.6 Random Frame Selection and Weighted Training Data

From the above experiments, we found that the following problems should be considered:

1. When the training data set is small, it is possible to use the whole data set to update weight changes each time. However, for the large training database, it will take an extremely long time to finish a single iteration. Hence, we need to input to the MLP a small amount of selected data from the large training data set each time;

2. We also need to have a phonetic balance between short and long phonemes to avoid over-training on vowels vs. consonants.

In this experiment, weighted training data and random frame selection are applied to solve the above problems. The network scaling is not applied here as scaling from 3 frames to 5 frames does not make much difference in training; besides, scaling may introduce oscillation. Our MLP consists of three layers (an input layer, a hidden layer, and an output layer). The input layer has 75 units (5 frames $\times$ 15 spectral coefficients from frame $t-2$ to $t+2$), i.e., the input layer covers a speech dynamic range of 70 msec. There are 30 units in the hidden layer. The output layer contains 21 units corresponding to 21 phonemes.

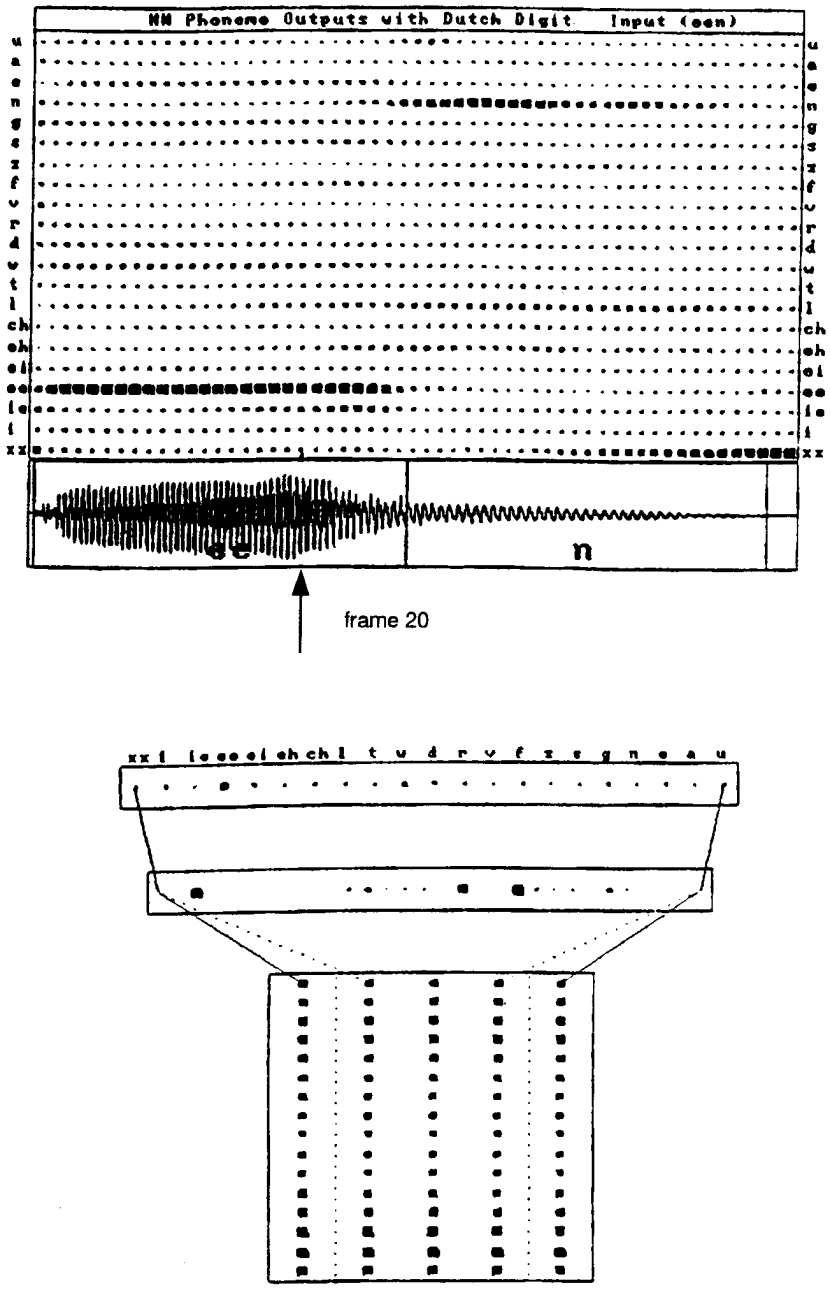

Figure 5.4: The MLP output (top), the Viterbi-labeled speech signal (middle), and internal activations with scaling (bottom).

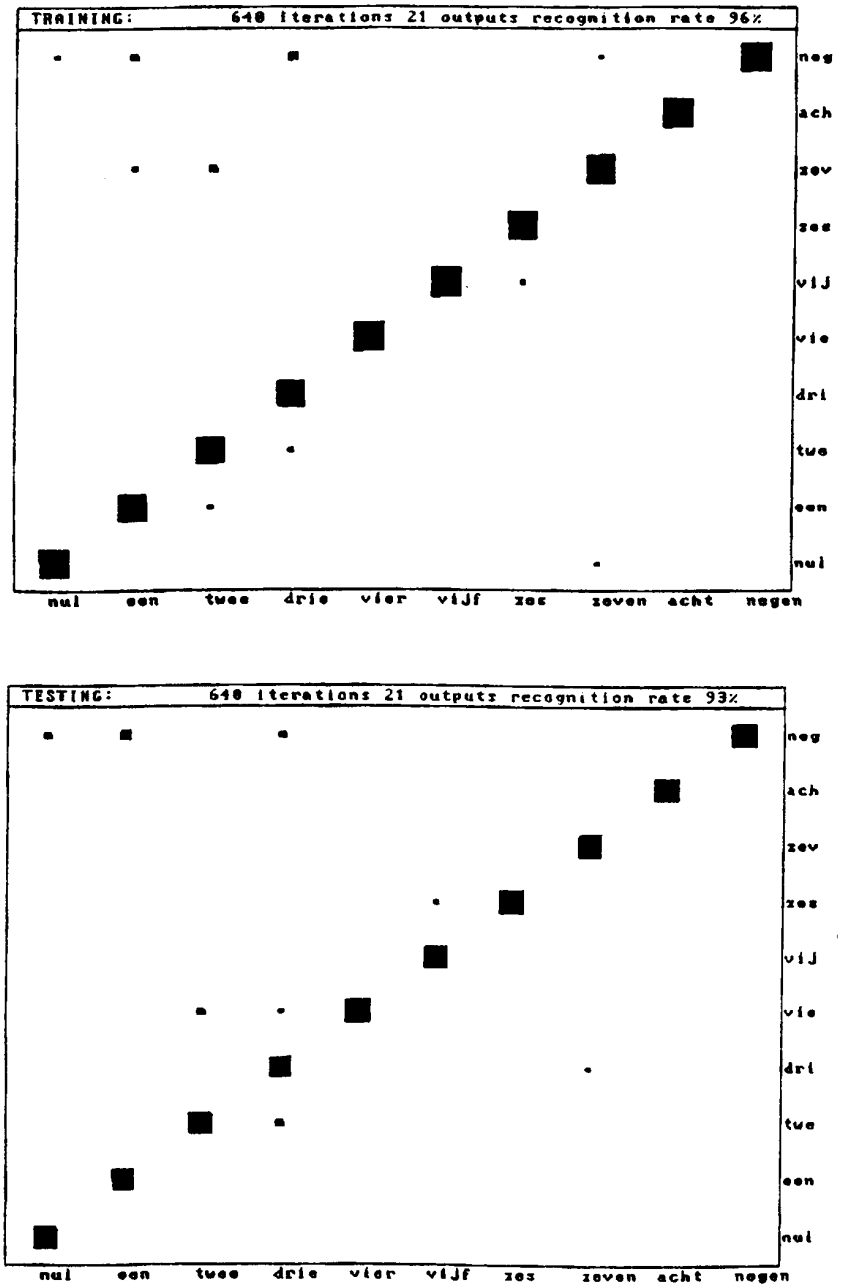

Figure 5.5: The digit recognition results for MLP labeling

phonetic labels	weights
xx	0.1
xx	0.2
xx	0.1
a	0.4
a	0.6
a	0.6
a	0.4

Table 5.2: Training data transcription with labels and weights for each frame

In this experiment, we use the small database. As we have the hand-segmented data already, Viterbi alignment is not applied to this experiment. We first apply weights to the training data. Because we use the context independent phonemes, the frames in the middle part of each phoneme are more important than those on the boundaries. Our purpose is to emphasize the middle frames and de-emphasize the boundary frames. The network will be robust to the middle part of each phoneme vs. the boundary part. This is done by applying a window function, which has large values in the middle and small values near the ends, for instance, a Hamming window. An example of how to put the weighting factors in a phonetic label file is shown in Table 5.2.

The pattern for the third frame is "xx" and the weight factor $\gamma = 0.1$, for instance. If the weight factor γ has the same value for every frame, the training data is not weighted. We use Equation 3.28 to adapt connectionist weights. The deletion of the boundary frame is not necessary, instead, the small weights were provided to those boundary frames by the Hamming window.

Then we apply the random data selection to the weighted training data. The hand-labeled training data contains a total of 16,300 frames. The shortest time phoneme is *d* , which has only 103 frames, and the longest one is *ee*, which has 1,562 frames. However, the noise, which has 6,337 frames, occupied more than one third of the training database. We proposed using random frame selection to generate the phonetically balanced training data. First, all frames for each phoneme are extracted from the whole data. This

No. of iterations	Without Weighting		With Weighting	
	Training	Testing	Training	Testing
100	80.5	63.5	87.5	64.5
500	96.0	74.5	93.0	75.0
1000	93.0	72.5	94.0	75.5
10000	95.0	68.5	97.0	71.5
20000	96.0	63.5	96.5	71.5

Table 5.3: The recognition results for using data weighting method

produces 21 collections corresponding to each phoneme. Then, 200 frames are selected randomly from each collection. In this way, we obtain a training set with 4,200 frames in total. Each phoneme has 200 representations. Our database has 20 speakers, so there are 10 representations per phoneme for each speaker on the average.

The modified Back-Propagation algorithm has been applied to train the MLP. The connectionist weights (including the offsets) are randomly initialized between [-0.3,+0.3]. For each iteration, a new training set generated in the way described above is used to adapt the connectionist weights. The HMM training is the same as the one in the previous experiment.

5.6.1 Discussion and Results

Table 5.6 shows the results of this experiment. The low recognition rate is due to our small database. 20 speakers are not enough to train HMMs and MLP for speaker independent speech recognition. If the Viterbi alignment is employed, the recognition rate will significantly increase for the test data which is shown in the experiments in the next section.

The results with weighting or without weighting are illustrated in Table 5.6. There are three cases: First, the number of iterations is small (< 500), the network is under-trained, the performance on both the training data and the test data improves as the number of iterations increases. Second, the network is well-trained, the performance on training data improves much more than that on the test data as the number of iterations increases. Third, the number of iterations is too big (> 1000), the network is over-trained, the performance on the test data degrades, and the performance on the training data improves very little. Table 5.6 shows that the results with weighting are better than those without weighting, especially for small

training data.

We have compared the 21-parameter MLP labeler with an Euclidean VQ with the same codebook size. Results for the MLP labeler are 75% correct vs. only 59% correct on the Euclidean VQ.

In our MLP, there are five frames in the input layer, from $t-2$ to $t+2$, so there is spectral derivative information in the network. Comparing the MLP VQ with the spectral derivative VQ with the code-book size 232 (100 cepstra + 100 delta-cepstra + 32 residual energy parameters), we obtained 73.5% correct for spectral derivative VQ based on the same database. The MLP-VQ with only codebook size 21 received better results than 232 spectral derivative VQ. This proves that the intrinsic acoustic phonetic distortion provided by the MLP labeling must be significantly smaller than the distortion provided by spectral derivative labeling. The MLP VQ has very high discriminative power.

Figure 5.6 shows the MLP outputs and manual segmentation results for training and test data. The size of the black spirals stands for the output value of MLP. The training and testing results for each word based on the weighted training database are shown through confusion matrices in Figure 5.7, Figure 5.8 and Figure 5.9. They represent three states of the network: under-trained, well-trained, and over-trained. The reference models are indicated by the row indexes, and the tested words by the column indexes.

From the above experiments, we see that the MLP labeler has very high performance in capturing speech features. Our MLP-VQ HMM recognizer has a lot of advantages over the normal VQ HMM recognizer such as its small codebook size and good recognition results. If we do not consider the long training time and the large amount of supervised training data, MLP-VQ can do better than normal VQ.

5.7 MLP Multi-Dimensional Labeling

In the following experiment, we focus on analyzing the MLP outputs. The strongest output represents the input pattern more powerfully than the others. However, the other outputs are also relevant to the input pattern as shown in the histogram in Section 3.7. Here we use the MLP multi-dimensional labeling strategy to handle the MLP output information.

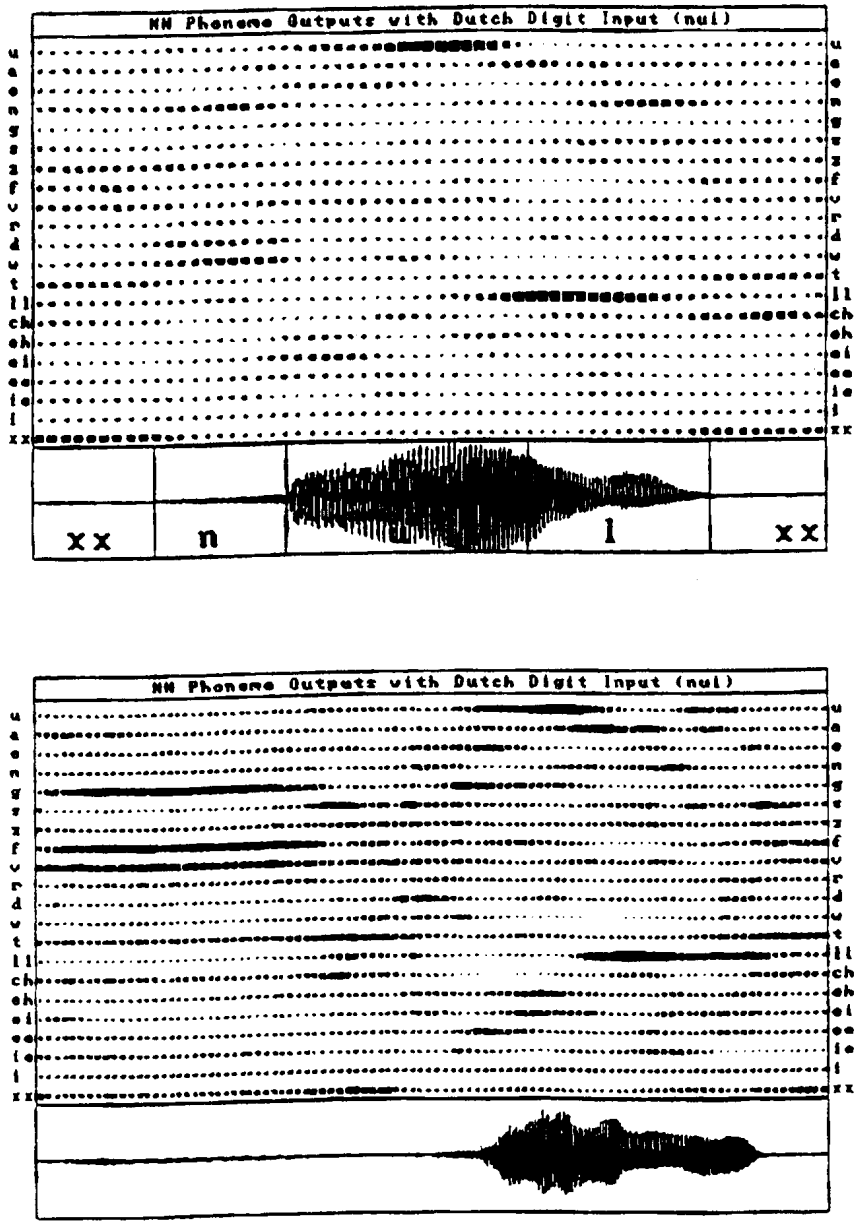

Figure 5.6: The MLP output and its manually labeled speech signal (top), and the MLP output and its testing speech signal (bottom)

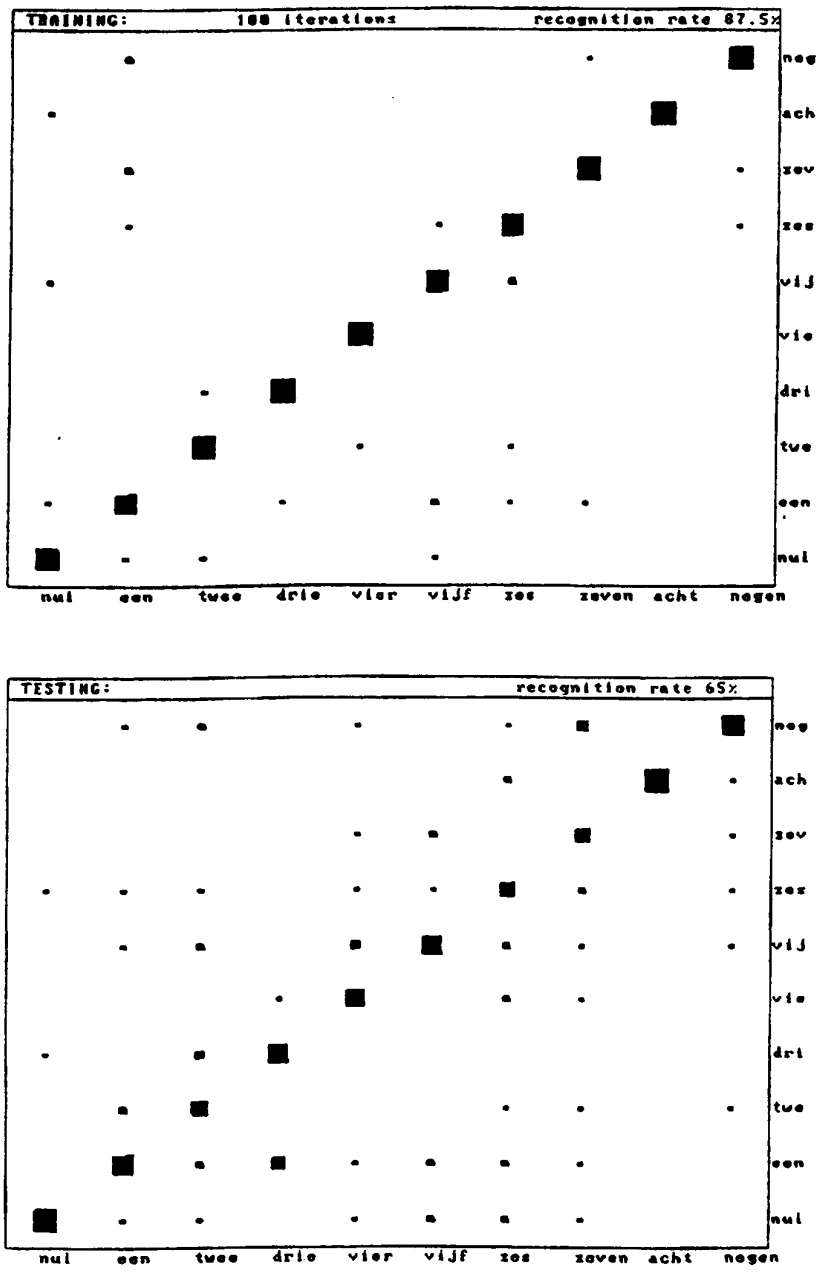

Figure 5.7: The recognition results for each digit when the network is under-trained.

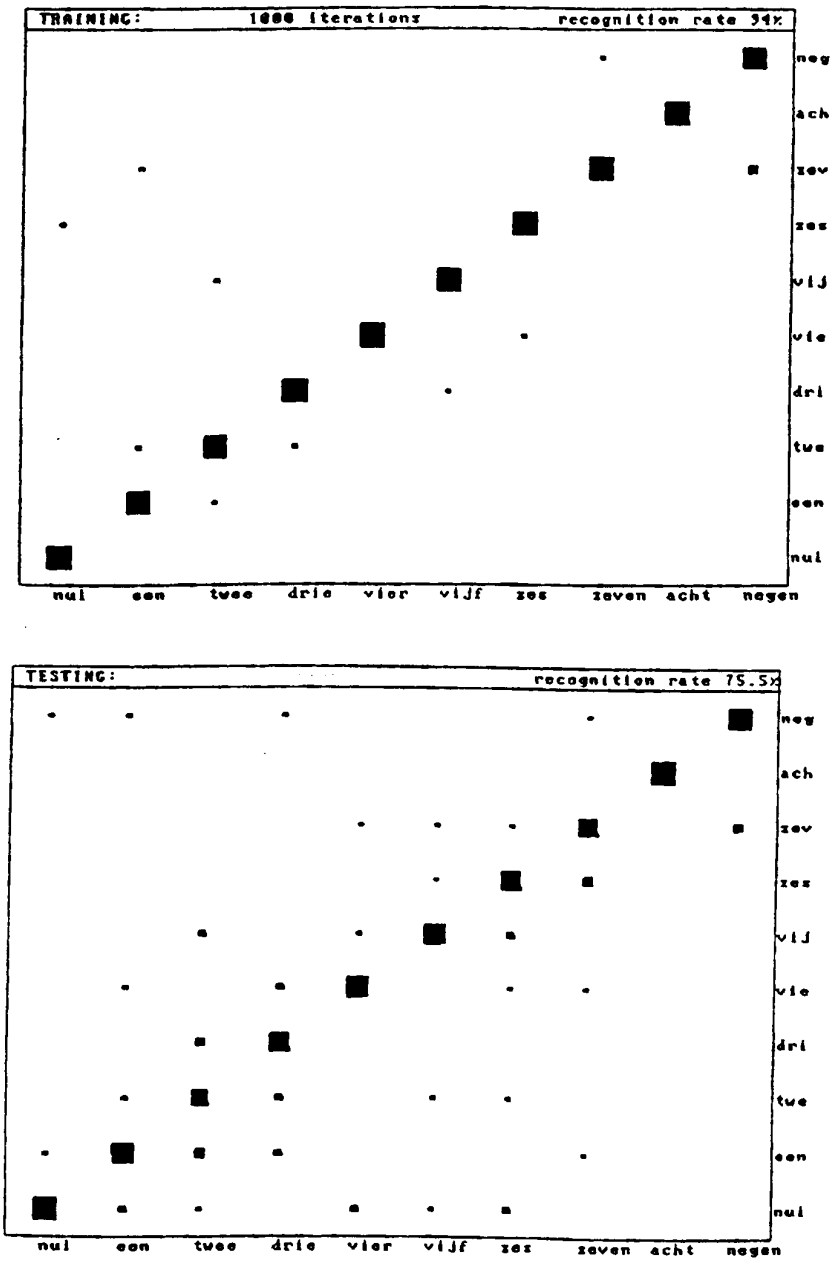

Figure 5.8: The recognition results for each digit when the network is well-trained.

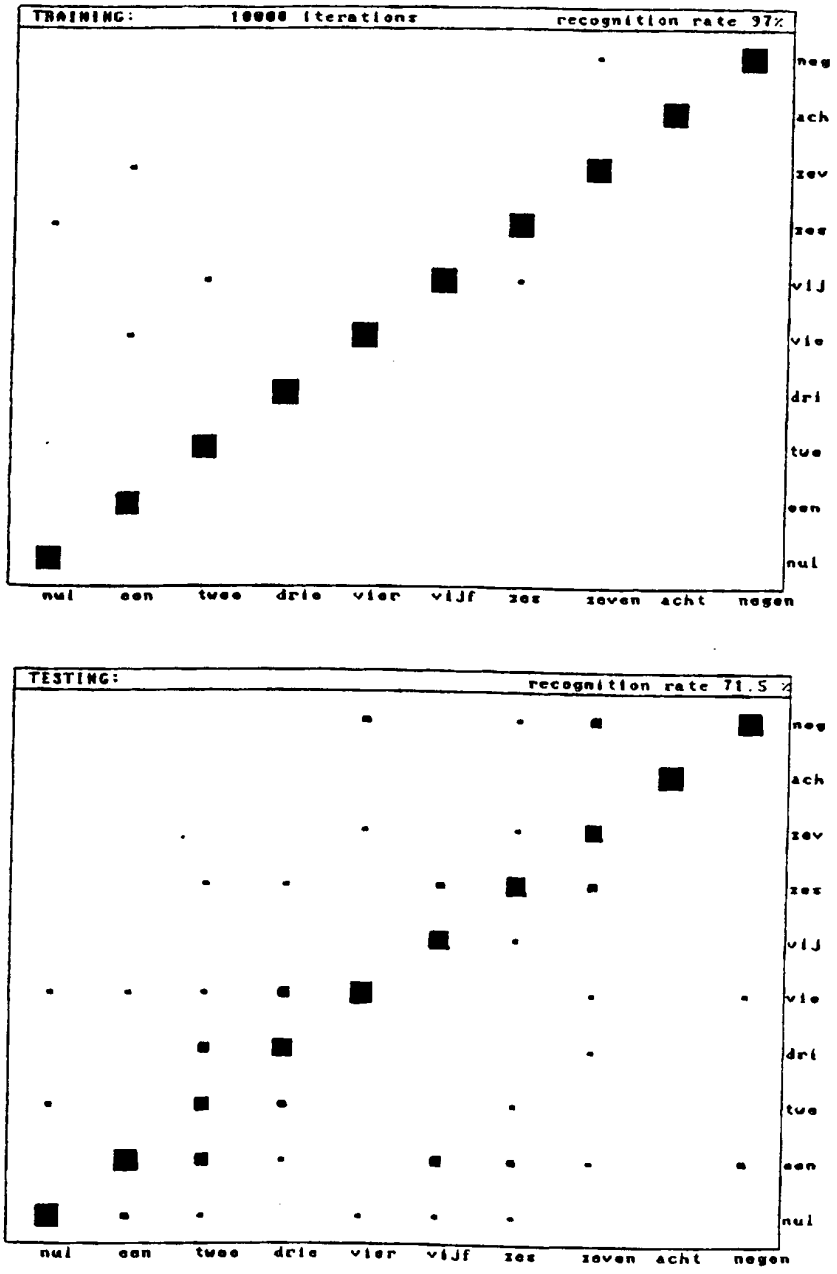

Figure 5.9: The recognition results for each digit when the network is over-trained.

5.7.1 Building the MLP Training Database

In order to supervise the MLP learning, the automatic phonetic segmentation has been obtained by Viterbi alignment from a HMM-like system bootstrapped from hand segmentation on 20 speakers.

The phonetically labeled database has been obtained from Viterbi alignments. This procedure results in 150,000 frames representing 21 phonemes. The shortest phoneme *d* has only 1100 frames and the longest one *ee* has 17,700 frames. The noise which has 55,000 frames occupies more than one third of the training database. 600 frames of each phoneme are selected randomly from the whole training database for each MLP training iteration. On average, there are 3 representations per phoneme per speaker, since our training data includes 200 speakers. The random frame selection method is used for every iteration during the connectionist weight adaptation.

5.7.2 System Training

The MLP training is summarized in the following steps:

1. Give an initial random seed S_{int}, the number of iterations N_{stop} for stopping the MLP training and the number of iterations N_{rept} for repeating the MLP training data.
2. Set the seed $s = S_{int}$, the iteration counter $i = 0$, the repetition counter $r = 0$.
3. Randomly generate 600×21 frames MLP training data from the training database.
4. Do the forward and the backward propagation and adapt the weights.
5. Increase the counters by 1.
6. Stop the MLP training if $i > N_{stop}$.
7. Set $s = S_{int}$ and $r = 0$ if $r = N_{rept}$, otherwise, generate a new seed.
8. Go to step 3.

The network is more robust in this way compared to using fixed training data for each iteration. After the network is ready, we take more than one

output label (the single winner-take-all label) as the HMM input labels. The HMM system uses multiple codebook linear word models with 15 states per word (In order to demonstrate the flexibility of MLP labeler over the MLP probability estimator, word models are applied in this experiment and the following ones. Therefore, the previous phonetic model experiments results are not applicable to the following experiments.).

5.7.3 Top-N Labels

We described the top-N labeling method in chapter 4. There we considered the top-N labels as a label vector in the HMM system. The elements of the label vector were assumed to be statistically independent.

The 21 labeled outputs (labels) of the neural network are sorted according to the values of the real MLP outputs from large to small. From the histogram shown in the Figure 3.12, we have found that the first three or four labeled outputs have a high probability of getting large output values and the rest have very a low probability near zero of getting large output values, so the suitable number of MLP labeled outputs should be less than 4. From this point of view, we moved from the winner-take-all method to the Top-N method, by which the top-N labels are used instead of the top-one label.

We consider the top-N labels as a real observation vector for the HMM system. If the vector length is equal to 2, there are $21 \times 21 = 441$ kinds of output vectors. When the vector length is longer than 2, there are too many combinations and a lot of them do not occur at all. To avoid this problem, we strip out all the combinations which do not or seldom occur to a null observation.

The results for the different number of labels are illustrated in Table 5.4. The best one is printed in bold face. We obtain 3% improvement compared to the winner-take-all method. The performance improves very little when the selected number is larger than three, so the optimal number of labeled outputs is less than four. The system with three output labels has approximately the same recognition accuracy as with four labels to twenty one output labels, but has much fewer parameters.

Nr of Labels	Rec. Rate (%)	Nr of Labels	Rec. Rate (%)
1	86.94	7	89.44
2	88.94	8	89.64
3	**89.99**	9	89.34
4	89.39	16	90.04
5	89.39	21	90.74
6	89.39	-	-

Table 5.4: The recognition results for multi-dimensional labeling

5.7.4 Output Adaptation

In the MLP training method (the back propagation algorithm), the desired output is one and the others are zeros, so the label corresponding to the largest output contains the major information of the input patterns. In the previous methods, the main performance of the largest output value is omitted since the selected labeled outputs are considered equal. Here the top-N label vector method with two labels was used. Three selecting rules are formulated to adjust the label vector using the MLP output values. The first one is by adapting the second output with the first output according to the gain factor α shown in Equation 5.2; the second is according to the threshold β which only considers the first element shown in Equation 5.3; and the third is according to the threshold γ which only considers the second element in the output shown in Equation 5.4.

$$\begin{aligned} if \quad (\alpha o_1 > o_2) \quad & v_1 = p_1 \; and \; v_2 = p_1 \\ otherwise \quad & v_1 = p_1 \; and \; v_2 = p_2 \end{aligned} \tag{5.2}$$

$$\begin{aligned} if \quad (o_1 > 1 - \beta) \quad & v_1 = p_1 \; and \; v_2 = p_1 \\ otherwise \quad & v_1 = p_1 \; and \; v_2 = p_2 \end{aligned} \tag{5.3}$$

$$\begin{aligned} if \quad (o_2 < \gamma) \quad & v_1 = p_1 \; and \; v_2 = p_1 \\ otherwise \quad & v_1 = p_1 \; and \; v_2 = p_2 \end{aligned} \tag{5.4}$$

α β γ	Recognition rate (%)		
	Equation 5.2	Equation 5.3	Equation 5.4
0.0	88.94	88.94	**88.94**
0.1	88.94	–	**88.94**
0.2	88.99	–	87.79
0.3	**89.09**	**89.09**	87.39
0.4	88.34	88.69	87.34
0.5	88.49	88.04	87.14
0.6	88.29	88.04	87.09
0.7	87.84	87.14	86.94
0.8	–	86.99	–
1.0	86.94	86.94	86.94

Table 5.5: The word recognition results obtained respectively from Equation 5.2 with α various, Equation 5.3 with β changing, and Equation 5.4 with γ having the different values. The best result is printed in bold face.

$$0 \leq \alpha, \beta, \gamma \leq 1$$

Here o_1 and o_2 are the MLP analog output value, p_1 and p_2 are the phonetically labeled outputs, v_1 and v_2 are the final MLP label outputs. In Equation 5.2 , all p_2 are replaced by p_1 when $\alpha = 1$, which is the same as in the one output case, and no replacement takes place when $\alpha = 0$, which is the same as in the previous case of two selected outputs. In Equation 5.3, for $\beta = 1$ and $\beta = 0$, and, in Equation 5.4, for $\gamma = 1$ and $\gamma = 0$, the results are the same as the one when $\alpha = 1$ and $\alpha = 0$, respectively. The experimental results are shown in Table 5.5. We observe that the performance is a little bit better with modified two labels than the two labels without modification.

5.7.5 Discussion

In these experiments, we have described the method to use the Top-N labels and we further get optimized discrete MLP output with output adaptation. A better performance with those methods has been achieved. We summarize our conclusions as follows:

- The MLP top-N labeling is more powerful than the MLP winner-take-all labeling. We obtain 3% improvement compared to the winner-take-all. The best performance is obtained by using the top 3 labels for the top-N labeling.

- The output adaptation, which uses the influence of the MLP analog output values, does not significantly improve the recognition performance. The following MLP fuzzy method, which fully utilizes the MLP analog output values, will further improve the recognition performance.

5.8 MLP Fuzzy Labeling

An MLP Fuzzy VQ (MLP-FVQ) combines both an MLP as a labeler and an MLP as a probability generator. The MLP provides the fuzzy credibility for the vector quantization. We are basically interested in using the soft decision of MLP-FVQ to reduce the quantization errors inherent when using a small code-book. We have analyzed some histograms of the phonetically labeled continuous MLP outputs that motivated our proposals in Section 3.7, and investigated the use of the Multi-Layer Perceptron (MLP) as a fuzzy credibility estimator in Section 4.7. Here, we provide some experimental results to prove our theory.

The experimental environment is the same as the one for MLP multi-dimensional multi-labeling. We use the MLP network generated in the previous experiment. The fuzzy scores are the MLP analog outputs. The modified Viterbi algorithm is used for the HMM system training and testing.

5.8.1 Results and Discussion

Two sets of experiments were performed to compare the performance of MLP-FVQ/HMM with MLP multi-dimensional VQ/HMM. First, training and recognition were both done using MLP-multi-dimensional-VQ/HMM. Then both training and recognition were done using MLP-FVQ and the modified HMM algorithm.

Table 5.6 shows the results of the recognition experiments based on the different number of the top-N labels. When $N = 1$, the fuzzy score is

NrOfLabels	MLP Multi-Dimensional VQ (%)	MLP VQ (%)
1	86.94	86.94
2	88.94	89.06
3	**89.99**	**91.14**
4	89.39	89.22
5	89.39	88.26
6	89.39	87.10

Table 5.6: The recognition results for MLP fuzzy labeling

always equal to one and the MLP-FVQ is just an MLP-VQ (winner-take-all). When $N = 3$, we have obtained the best results in all cases. When $N > 3$, the improvement seems to level off. We believe this is attributable to the random labels when N is very large.

In our experiments, the speech recognition rate on a speaker-independent (400 speakers) database increased by 4 % at least when using MLP fuzzy VQ compared with the winner-take-all method. It is 1 %better than the MLP multi-dimensional labeling. The MLP-FVQ has a smoothing capability. It reduces the number of labels needed for the parameter estimation during training and decreases some of the effects of VQ errors, especially for the small codebook, during recognition.

5.9 Summary

We have devoted most of our efforts in this chapter to exploring some practical issues concerning the implementation aspects of the MLP/HMM hybrid system.

We have designed an MLP-VQ HMM recognizer. The training of the MLP labeler is supervised by a Viterbi alignment from an HMM using normal VQ. The Viterbi alignment provides a way to train MLP for a very large amount of unlabeled training data.

Building the MLP training database is very important. A random frame selection method has been applied to overcome the huge size of the database and the various lengths of phonemes. The method of weighting the training database can emphasize important speech features - e.g., the middle parts of

phonemes. Our MLP is made more robust for short phonemes (consonants) by using a random frame selection method. In small scale experiments, we have observed that by weighting MLP training data, the obtained results are much better than those obtained without using this method.

The MLP VQ outperformed standard Euclidean VQ for identical codebook sizes (21). The MLP VQ with a small codebook size had higher performance than the spectral derivative VQ with a very large codebook size (232).

The MLP multi-dimensional labeling method expands the MLP-VQ winner-take-all method; the MLP fuzzy labeling produces both labels and closeness measures for the input speech features. In our experiments, the speech recognition rate increases 3 % by using the MLP multi-dimensional labeling compared with the MLP winner-take-all labeling; and the MLP fuzzy labeling has reached 1 % higher accuracy than the MLP multi-dimensional labeling.

Chapter 6

Conclusions

6.1 Introduction

In this dissertation, we have described the state-of-the-art speech processing technologies and Multi-Layer Perceptron (MLP) neural networks for speech recognition. We have integrated the Multi-Layer Perceptron with Hidden Markov Models (HMM) and presented several MLP/HMM hybrid speech recognition systems, where the MLP works as a labeler or a fuzzy membership function generator. In this chapter, we summarize our main findings and contributions, and we discuss further work related to the topic of this dissertation.

6.2 Contributions and Findings

6.2.1 Multi-Layer Perceptron

We have designed and implemented the Multi-Layer Perceptron to be used as a labeler for the Hidden Markov Modeling system. We have presented the Back Propagation theory and the implementation issues in detail.

Modified Back Propagation Algorithm: We have proposed a modified back-propagation algorithm, which we have implemented based on the phonetic characteristics of the speech signal. Because the phonemes used in this work are context-independent, the frames in the middle part of each phoneme are more important than those on the boundaries. We have used

the Hamming window to emphasize the frames in the middle of each phonetic class in the speech training data and to de-emphasize the boundary frames. In our small scale experiments, we have obtained better results by using the weighted training data than we did without using the weighted training data.

MLP Training: We have proposed and designed the MLP labeler and MLP-VQ HMM hybrid systems. The training of the MLP labeler is supervised by a Viterbi alignment from an HMM using normal VQ. The Viterbi alignment provides a way to automatically generate phonetic segments for a very large amount of unlabeled data. The MLP training data directly influences the MLP classification capability. We have proposed a random frame selection method for building the MLP training data. This method has allowed us to overcome the huge size of the database and the various lengths of phonemes. Moreover, the MLP network is more robust for speech recognition when we use random data selection together with the data weighting method than when these two techniques are not be used.

MLP Output Analysis: We have used histograms to illustrate the MLP output values for each phonetic class. From the histograms, we have observed the relativity of different phonetic classes. We have further proposed to use the MLP analog output values and the top-N labels in speech recognition on top of our conventional winner-take-all MLP method.

Fuzzy MLP: We have given the MLP a fuzzy interpretation and have proposed using the overlapping Hamming windows as the fuzzy membership function for the MLP output. The fuzzy MLP model is more suitable for speech recognition in situations where speech features overlap because of coarticulation, than the conventional crisp MLP models.

6.2.2 Multi-Layer Perceptron Labeling

MLPs have most of the desirable properties needed for robust speech recognition. These properties lie mainly in their discriminative power and their ability to deal with non-explicit knowledge. Moreover, contextual information can easily be taken into account. MLPs have very high performance for phoneme recognition using multiple frames as inputs. On the other

hand, Hidden Markov Models have a good dynamic time warping capability but low discriminative power. The combination of the two systems can lead to high performance recognizers. We proposed using the MLP as a labeler for a discrete HMM system. Our approach is different from Bourlard's MLP/HMM hybrid system [8], where the MLP works as a probability estimator. Compared to the probabilistic interpretation of MLPs, our MLP approaches need fewer hidden units and less training time, and are more flexible in system design (e.g., use HMM word models instead of HMM phonetic models).

We also proposed using the MLP as a fuzzy membership generator. The MLP output measures the closeness between the input pattern and each output phonetic class.

Our experiments indicate that the MLP VQ outperforms the standard Euclidean VQ for identical codebook sizes and that the MLP VQ with a small codebook size has higher accuracy than the spectral derivative VQ with a very large codebook size.

6.2.3 MLP/HMM Hybrid Models

We have also performed a systematic analysis of the MLP labeling and compared it with the conventional K-means VQ. The simplest implementation of MLP labeling consists in using the output of the strongest unit as a discrete output. The implementation ignores other MLP outputs that contain a lot of other speech information. We have proposed a series of MLP/HMM hybrid models to fully use the MLP output information and to increase the speech recognition performance. The proposed models are:

- **MLP Multi-Dimensional Labeling** This is our initial method to use more labels in addition to only the MLP-winner-take-all label. At first, a simple way to incorporate the information of other output units into the system is to use not only the top scoring output, but the N top scoring outputs as labels. This method has the advantage of using information from N output units instead of just one. There are now N label streams from the MLP to the HMM instead of one. This way the MLP input parameter space is much more finely described. The HMM can then use these labels as independent observation variables, just as labels from multiple codebooks were used. We have further proposed to apply the output adaptation method to

the top-N labels based on the output scores instead of treating them equally. (For results, see Section 5.7.)

- **MLP Multi-Labeling** This model has a stronger theoretical foundation than the above MLP multi-dimensional labeling. Instead of using multiple codebooks, which violates the HMM independent assumption, the MLP multi-labeling uses the VQ/HMM unification theory to deal with the multiple labels corresponding to the N highest outputs or the outputs above a preset threshold. The modified HMM algorithm is applied here. (Only a theoretical analysis is made with this method.)

- **MLP Fuzzy-Labeling** In this model, the MLP analog outputs provide the fuzzy closeness measure of the input speech pattern to the output phonetic classes. The HMM observation is a vector consisting of all MLP analog outputs. Those MLP analog outputs are further interpreted as mass probabilities. The HMM models are generated by using the modified HMM algorithm. (For results, see Section 5.8.)

- **Multi-MLP Multi-Labeling** Instead of training only one MLP for phonetic classification with an input of all parameter sets, multiple independent MLPs are trained for phonetic classification. The input to those multiple MLPs are basic spectral parameters, first derivatives, and second derivatives of the basic spectral parameters, respectively. LPC based cepstra were used as MLP basic spectral parameters. Parameters are chosen if they meet the HMM independent assumption. Therefore, the outputs of the networks can be assumed to be independent. Hence, the probabilities associated with the labels from the different MLPs can be multiplied. During recognition, the MLPs work in parallel. For each network, the labels corresponding to the N highest scoring outputs or the outputs larger than a preset threshold provide top labels to the modified HMM system. (Only a theoretical analysis is made with this method.)

- **Multi-MLP Fuzzy Labeling** In this model, the MLP fuzzy labeling replaces the MLP Multi-Labeling from the previous model. Because of both multiple MLP labeling computation and the multiple modified HMM calculation, the multi-MLP fuzzy labeling needs more computation power and memory than the other methods. On

the other hand, we expect that this method yields the highest recognition rate above all other models. Future experiments are needed for this model.

The detailed description of each model is presented in chapter 4. The above models outperform the winner-take-all MLP labeling. The major drawback of the above models, however, is the amount of extra computation time needed for calculations involving the modified Viterbi and/or forward-backward algorithm. From the experiments we have done so far, the extra computation time is too little to effect the system performance. The system, on the other hand, has higher recognition accuracy.

6.3 Future Work and Suggestions

6.3.1 MLP/HMM Hybrid Models

We have described MLP Labeling/HMM hybrid systems, and have presented all their possible models in detail. We have done our experiments on MLP multi-dimensional labeling and MLP fuzzy labeling. We have not done experiments on MLP multi-labeling, multi-MLP multi-labeling and multi-MLP fuzzy-labeling. Although we can predict the results for those untested models from the experiments we have done so far, it is necessary to perform further experiments to test all MLP labeling/HMM hybrid models.

6.3.2 Output Feed Forward Multi-Layer Perceptron

We have presented the Multi-Layer Perceptron to deal with the coarticulation problem in speech recognition. For future work, we propose another structure: the *output feed forward Multi-Layer Perceptron* neural network.

Figure 4.14 shows the output of the different phoneme classes with input presenting a Dutch digit *een*. The curves reveal the obvious coarticulation phenomenon. When the speech begins, the value for phoneme *ee* increases while the value for noise *xx* decreases; as time goes on, the value for phoneme *ee* increases up to the saturation and then decreases. As the value for phoneme *ee* decreases, the value for phoneme *n* increases, and so on for noise *xx*.

Speech recognition accuracy may be increased if we can represent this basic coarticulation characteristic inside a neural network rather than in the

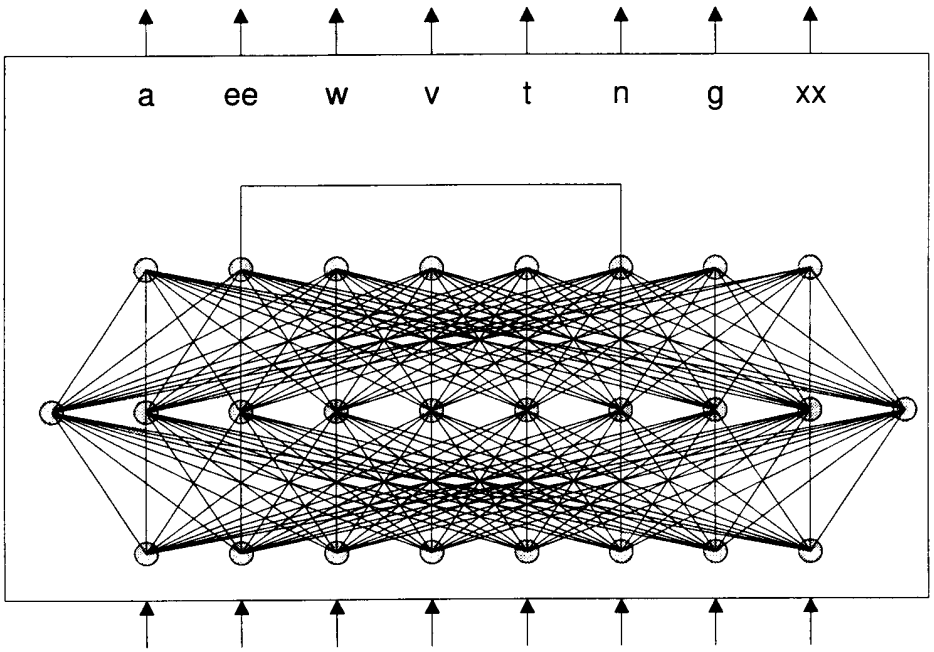

Figure 6.1: A feed forward neural network

MLP input (by using the weighted training data), or by the MLP output (as in fuzzy MLP). The feed forward neural network takes the coarticulation into account by connecting the units in the output layer. The initial structure is shown in Figure 6.1. We consider only the current phoneme and its adjacent previous phoneme as at those places where the coarticulation happens. There is no connection between the current class output unit and other units. For example, there is a feed forward connection between phoneme *ee* and phoneme *n* at beginning of phoneme *n* for Dutch digit *een*. The output feed forward Multi-Layer Perceptron can also be considered as a recurrent neural network at the output layer. We need to further explore this idea in detail.

Bibliography

[1] **Amit D. J.** Neural Networks for Counting Chimes. *Proceedings National Academy Science, USA, Biophysics 85*, pages 2141–2145, 1988.

[2] **Applebaum T. H.** and **Hanson B. A.** Enhancing the Discrimination of Speaker Independent Hidden Markov Models with Corrective Training. In *STL Research Report number 2*. Speech Technology Laboratory, 1990.

[3] **Bahl L. R.**, **Brown P. F.**, **De Souza P. V.**, and **Mercer R. L.** Speech Recognition with Continuous-parameter Hidden Markov Models. *Computer Speech and Language*, 2(3/4):219–234, 1987.

[4] **Bahl L. R.**, **Brown P. F.**, **De Souza P. V.**, and **Mercer R. L.** A New Algorithm for the Estimation of Hidden Markov Model Parameters. In *Proc. Int. Conf. Acoust. Speech & Signal Processing*, page S11.2, New York, U.S.A., 1988.

[5] **Bahl L. R.**, **Brown P. F.**, **De Souza P. V.**, and **Mercer R. L.** Estimating Hidden Markov Model Parameters So As To Maximize Speech Recognition Accuracy. *IEEE Transactions on Speech and Audio Processing*, 1(1):77–83, 1993.

[6] **Baum L.E.** An inequality an associated maximization technique in Statistical estimation of probabilistic functions of Markov Process. *Inequalities*, 2(1):1–8, 1972.

[7] **Bezdek J. C.** *Pattern Recognition With Fuzzy Objective Function Algorithms*. Plenum Pub Corp, June 1981.

[8] **Bourlard H.** and **Morgan N.** A Continuous Speech Recognition System Embedding MLP into HMM. In Touretzky D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 186–193. Morgan Kaufmann Publishers, 1991.

[9] **Bourlard H.** and **Morgan N.** *Connectionist Speech Recognition – A hybrid Approach*. Kluwer Academic Publishers, 1994.

[10] **Bourlard H.** and **Wellekens C. J.** Links Between Markov Models and Multilayer Perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12(12):1167–1178, December 1990.

[11] **Bridle J. S.** and **Brown M. D.** A Data-Adaptive Frame Rate Technique and its Use in Automatic Speech Recognition. In *Proc. Inst. of Acoustics Autumn Conf.*, pages C2.1–C2.6, 1982.

[12] **Bridle J. S.**, **Brown M. D.**, and **Chamberlain R. M.** An Algorithm for Connected Word Recognition. In *Proc. Int. Conf. Acoust. Speech & Signal Processing*, pages 899–902, 1982.

[13] **Brown P. F.** *The Acoustic-Modeling Problem in Automatic Speech Recognition*. PhD thesis, Carnegie-Mellon, 1987.

[14] **Buhmann J.** and **Schilten K.** Noise-Driven Temporal Association in Neural Networks. *Europhysics Letters 4*, pages 1205–1209, 1988.

[15] **Burr D. J.** 'Experiments on Neural Net Recognition of Spoken and Written Text. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36:1162–1168, November 1988.

[16] **Chow Y. L.**, **Dunham M. O.**, **Kimball O. A.**, **Krasner M. A.**, **Kubala G. F.**, **Makhoul J.**, **Price P. J.**, and **Roucos S.** BYBLOS: The BBN Continuous Speech Recognition System. In *Proc. Int. Conf. Acoust. Speech & Signal Processing*, pages 89–92, 1987.

[17] **Claes T.** and **Van Compernolle D.** SNR-Normalization for Robust Speech Recognition. In *Proc. Int. Conf. Acoust. Speech & Signal Processing*, volume I, pages 331–334, 1996.

[18] **Clark H. H.** *Using Language*. Cambridge University Press, 1996.

[19] **Grossberg S. Cohen,M. A.** and **Stork D.** Recent Developments in a Neural Model of Real-Time Speech Analysis and Synthesis. In *First International Conference on Neural Networks, IEEE*, pages 899–902, 1987.

[20] **Dehaene S.**, **Changeux J.**, and **Nadal J.** Neural Networks that Learn Temporal Sequences by Selection. *Proceedings National Academy Science, USA, Biophysics 82*, pages 2724–2733, 1989.

[21] **Duda R. O.** and **Hart P. E.** *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[22] **Fu L.-M.** Analysis of the Dimensionality of Neural Networks for Pattern Recognition. *Pattern Recognition*, 23(10):1131–1140, 1990.

[23] **Gray R. M.** Vector Quantization. *IEEE ASSP Magazine*, pages 4–29, April 1984.

[24] **Gupta V. N.**, **Lenning M.**, and **Mermelstein P.** Integration of Acoustic Information in a Large Vocabulary Word Recognizer. In *Proc. Int. Conf. Acoust. Speech & Signal Processing*, pages 697–700, 1987.

[25] **Haykin S.** *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company, Inc., 1996.

[26] **Hermansky H.** Perceptual Linear Predictive (PLP) Analysis of Speech. *J. Acoust. Soc. Am.*, 87(4), 1990.

[27] **Bayya A. Hermansky H.**, **Morgen N.** and **Kohn Ph.** Compensation for the Effect of the Communication Channel in Auditory-Like Analysis of Speech (RASTA-PLP). In *Proc. EUROSPEECH 91*, pages 1367–1370, Genua, Italy, September 1991.

[28] **Hornik K.**, **Stinchcombe M.**, and **White H.** Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 2:359–366, 1989.

[29] **Huang X. D.** and **Jack M. A.** Semi-Continuous Hidden Markov Models for Speech Recognition. *Computer Speech and Language*, 3:239–251, 1989.

[30] **Huang X. D.** and **Jack M. A.** Unified Modeling of Vector Quantization and Hidden Markov Model Using Semi-Continuous Hidden Markov Models. In *Proc. Int. Conf. Acoust. Speech & Signal Processing*, pages 639–642, 1989.

[31] **Huang X. D., Ariki Y.** and **Jack M. A.** *Hidden Markov Models for Speech Recognition*. Edinburgh University Press, 1990.

[32] **Huang X. D., Hon H.W., Hwang M.Y.** and **Lee K.F**. Comparative Study of Discrete, Semi-continuous and Continuous Hidden Markov Models . *Computer Speech and Language*, 7:359–368, 1993.

[33] **Iwamida H.**, **Katagiri S.**, and **McDermott E.** Speaker-Independent Large Vocabulary Word Recognition Using an LVQ/HMM Hybrid Algorithm. In *Proc. Int. Conf. Acoust., Speech & Signal Processing*, pages 497–500, Toronto, Canada, 1990.

[34] **Soderstrom T. Janssen P.** , **Stoica P.** and **Eykhoff P.** Model Structure Selection for Multi-Variable Systems by Cross-Validation. *International Journal of Control*, 47:1737–1758, 1988.

[35] **Kleinfield D.** Sequential State Generation by Model Neural Networks. *Proceedings National Academy Science, USA, Biophysics 83*, pages 9469–9473, 1988.

[36] **Kohonen T.** Learning Vector Quantization. *Neural Networks*, 1(9):303, September 1988.

[37] **Kohonen T.** . *Self-organization and Associative Memory (2nd Ed.)*. Springer, Berlin-Heidelberg-New Yok-Tokyo, 1988.

[38] **Le Cerf Ph.**, **Ma W.**, and **Van Compernolle D.** Using multilayer perceptrons as labelers for hidden Markov models. *Interdisciplinair Centrum voor Neurale Netwerken*, K.U.Leuven, November 19 1992.

[39] **Le Cerf Ph.**, **Ma W.**, and **Van Compernolle D.** Multilayer Perceptrons as Labelers for Hidden Markov Models. *IEEE Transactions on Speech and Audio Processing, 2(1)*, pages 185–193, January 1994. Special Issue on Neural Networks for Speech Processing.

[40] **Le Cerf Ph.** and **Van Compernolle D.** Speaker Independent Small Vocabulary Speech Recognition Using MLPs for Phonetic Labeling. In *Proceedings EUROSPEECH 93*, Berlin, Germany, September 21-23 1993.

[41] **Le Cerf Ph.** and **Van Compernolle D.** Using Parallel MLPs as Labelers for Multiple Codebook HMMs. In *Proc. Int. Conf. Acoust., Speech & Signal Processing*, pages I.561–I.564, Minneapolis, U.S.A., April 27-30 1993.

[42] **Le Cerf Ph.** and **Van Compernolle D.** Using MLP's as Probability Generators vs. as Labelers: A Comparative Study. In *In Holt M.J. J., Cowan C.F.N., Grant P. M., and Sandham W.A., editors,* Signal Processing VII: Theories and Applications: Proceedings of EUSIPCO-94, pages 1629–1632, Edinburgh, U.K., September 1994.

[43] **Lee, K.F.** *Automatic Speech Recognition: The Development of the SPHINX System.* Kluwer Academic Publishers, 1989.

[44] **Lee K. F.** and **Hon H. W.** Speaker-Independent Phone Recognition Using Hidden Markov Models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-37:1641–1648, November 1989.

[45] **Linde Y.**, **Buzo A.**, and **Gray R. M.** An Algorithm for Vector Quantizer Design. *IEEE Trans. on Communications*, COM-28(1):84–95, January 1980.

[46] **Lippmann R. P.** An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine*, 4:4–22, April 1987.

[47] **Roucos S. Makehoul J.** and **Gish H.** Vector Quantization in Speech Coding. *Proceedings of the IEEE*, 73:1551–1588, 1985.

[48] **Markel J. D.** and **Gray A. H.** *Linear prediction of Speech.* Communication and Cybernetics. Springer-Verlag, Berlin, 1976.

[49] **Markowitz J. A.** . *Using Speech Recognition.* Prentice Hall PTR, 1996.

[50] **Ma W.** Time Delay Neural Networks and TDNN-VQ HMM Recognizer. Master's thesis, K. U. Leuven, E.S.A.T., August 1990.

[51] **Ma W.** Unisys NLU ASR Evaluation Report. Technical report, Unisys Corporation, April 1998.

[52] **Ma W.**, **Le Cerf Ph.**, and **Van Compernolle D.** A Time Delay Neural Network Labeler for a Hidden Markov Model Speech Recognizer. Presented at the *Internal Workshop of the Interdisciplinary Center for Neural Networks*, K. U. Leuven, October 21 1991.

[53] **Ma W.** and **Van Compernolle D.** TDNN Labeling for a HMM Recognizer. In *Proc. Int. Conf. Acoust. Speech & Signal Processing*, pages 421–424, Albuquerque, U.S.A., 1990.

[54] **Ma W.** and **Van Compernolle D.** TDNN Fuzzy Vector Quantizer for a Hybrid TDNN/HMM System. In *Proceedings of the 1992 International Joint Conference on Neural Networks*, pages III581–III586, Beijing, China, November 1992.

[55] **Ma W.** and **Van Compernolle D.** SPCHLAB USER'S GUIDE (V1.0). Technical report, K.U.Leuven-ESAT-MI2-Speech Group, December 1993.

[56] **Ma W.**, **Van Diest M.**, and **Van Compernolle D.** Discrete TDNN Output Optimization for a Hybrid TDNN/HMM System. Internal Report MI2-SPCH-91-2, K. U. Leuven, E.S.A.T., June 1991.

[57] **Mercier G.**, **Bigorgne D.**, **Miclet L**, **Le Guennec L.**, and **Querre M.** Recognition of Speaker-dependent Continuous Speech with KEAL. In *Proceedings of the IEEE*, volume 136(2), pages 145–154, 1989.

[58] **Mirchandani G.** and **Cao W.** On Hidden Nodes for Neural Nets. *IEEE Transactions on Circuits and Systems*, 36(4):661–664, May 1989.

[59] **M. Nishimura** and **K. Toshioka**. HMM-based speech recognition using multi-dimensional multi-labeling. In *Proc. Int. Conf. Acoust. Speech & Signal Processing*, pages 1163–1166, 1989.

[60] **Morgan D. P.** and **Scofield C. L.** *Neural Networks and Speech Processing*. Kluwer Academic Publishers, 1991.

[61] **Ochsman, R.B.** and **Chapanis**. The effects of 10 communication modes on the behavior of teams during cooperative problem solving. *Intern. J. Man-Machine Studies*, 6:579–619, 1975.

[62] **Oppenheim A. V.** *Digital Signal Processing.* Prentice Hall, 1974.

[63] **O'Shaughnessy D.** *Speech Communication: Human and Machine.* Addison-Wesley Publishing Company, 1987.

[64] **Poritz A. B.** and **Richter A. G.** On Hidden Markov Models in Isolated Word Recognition. In *Proc. Int. Conf. Acoust. Speech & Signal Processing*, pages 262–265, 1986.

[65] **Rabiner L. R.** and **Sondhi M. M.** *Digital Processing of Speech Signals.* Pretice-Hall, New Jersey, 1978.

[66] **Rabiner L. R.**, **Sondhi M. M.**, and **Levinson S. E.** A Vector Quantizer Incorporating Both LPC Shape and Energy. In *Proc. Int. Conf. Acoust. Speech & Signal Processing*, 1984.

[67] **Rabiner L. R.**, **Wilpon J. G.**, and **Soong K. F.** High Performance Connected Digit Recognition Using Hidden Markov Models. In *Proc. Int. Conf. Acoust. Speech & Signal Processing*, New York, U.S.A., April 1988.

[68] **Rao V. B.** and **Rao H. V.** *Neural Networks and Fuzzy Logic.* MIS:Press, 1995.

[69] **Renals S.**, **Morgan N.**, **Cohen M.**, and **Franco H.** Connectionist Probability Estimation in the Decipher Speech Recognition System. In *Proc. Int. Conf. Acoust., Speech & Signal Processing*, volume I, pages 601–604, San Francisco, U.S.A., March 1992.

[70] **Robinson T.** A Real-Time Recurrent Error Propagation Network Word Recognition System. In *Proc. Int. Conf. Acoust., Speech & Signal Processing*, volume I, pages 617–620, San Francisco, U.S.A., March 1992.

[71] **Robinson T.** and **Fallside F.** A Recurrent Error Propagation Network Speech Recognition System. *Computer Speech and Language*, 5:259–274, 1991.

[72] **Ross T. J.** . *Fuzzy Logic With Engineering Applications.* McGraw-Hill, Inc., 1995.

[73] **Rumelhart D. E.**, **Hinton G. E.**, and **Williams R. J.** Learning Representations by Back-Propagating Errors. *Nature*, 323(9):533–536, October 1986.

[74] **Rumelhart D. E.** and **McClelland J.L.** . *Parallel Distributed Processing*. Cambridge, MA. MIT Press, 1986.

[75] **Sankar K. P.** and **Sushmita M.** Multilayer Perceptron, Fuzzy Sets and Classification. *IEEE Transactions on Neural Networks*, 3(5):683–697, September 1992.

[76] **Scholz K.W.** Simplifying the Development of Speech-Enabled IVR Applications. In *SpeechTEK Conference & Exhibition*, pages 237–266, New York, U.S.A., September 1997.

[77] **Singer E.** and **Lippmann R.** A Speech Recognizer using Radial Basis Function Neural Networks in an HMM Framework. In *Proc. Int. Conf. Acoust., Speech & Signal Processing*, volume I, pages 629–632, San Francisco, U.S.A., March 1992.

[78] **Specht D.** Probabilistic Neural Networks. *Neural Networks*, 3:109–118, 1990.

[79] **Sugawara K., Nishimura M., Toshioka K., Okochi M.** and **Kameko T.** Isolated Word Recognition Using Hidden Markov Models. In *Proc. Int. Conf. Acoust. Speech & Signal Processing*, pages 1–4, 1985.

[80] **Thimm G.** and **Fiesler E.** High Order and Multilayer Perceptron Initialization. *IEEE Transactions on Neural Networks*, 8(2):1045–9227, 1997.

[81] **Tseng H.**, **Sabin M.**, and **Lee E.** Fuzzy Vector Quantization Applied to Hidden Markov Modeling. In *Proc. Int. Conf. Acoust. Speech & Signal Processing*, volume 1, pages 641–644, 1987.

[82] **Utela P.**, **Kaski S.**, and **Torkkola K.** Using Phoneme Group Specific LVQ-codebooks with HMMs. In *Int. Conf on Spoken Language Processing*, pages 551–554, March 1992.

[83] **Van Compernolle D.** The ESAT Hidden Markov Model Speech Recognition System. Internal Report MI2-SPCH-90-2, K. U. Leuven, E.S.A.T., December 1990.

[84] **Van Compernolle D.** Development of a Computational Auditory Model. Technical Report 784, Institute for Perception Research, Eindhoven, February 1991.

[85] **Van Compernolle D.** Signal Processing for Speech Applications - Formulas and Algorithms. Internal Report MI2-SPCH-91-5 Revision V2.1, K. U. Leuven, E.S.A.T., May 1992.

[86] **Van Compernolle D.** Speech Recognition by Hidden Markov Models: An Introduction. Internal Report MI2-SPCH-92-2, K. U. Leuven, E.S.A.T., May 1992.

[87] **Van Compernolle D.** The ESAT Hidden Markov Model Speech Recognition System. Internal Report MI2-SPCH-92-3, K. U. Leuven, E.S.A.T., May 1992.

[88] **Van Compernolle D.**, **Ma W.**, and **Van Diest M.** Speech Recognition in Noisy Environments with the Aid of Microphone Arrays. In *Proceedings EUROSPEECH 89*, pages 2:657–660, Paris, France, September 1989.

[89] **Van Diest M.** *Studie van Probabilistische en Afstandsscores bij Spraakherkenning: Ontwerp van een Heuristische Spraakherkenner*. PhD thesis, K. U. Leuven, E.S.A.T., October 1991.

[90] **Waibel A.**, **Hanazawa T.**, **Hinton G.**, **Shikano K.**, and **Lang K. J.** Phoneme Recognition Using Time-Delay Neural Networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-37:328–339, March 1989.

[91] **Werbos P.** *Beyond regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University, Cambridge, MA, 1974. Ph.D. Thesis.

[92] **Werker J. F.** Becomimg a native listenner. *American Scientist*, Vol. 77:54–59, Janurary 1989.

[93] **Widrow B.** and **Hoff M. E.** Adaptive Switching Circuits. *IRE WESCON Conv. Record*, 4:96–104, 1960.

[94] **Wilpon J. G.**, **Lee C.-H.**, and **Rabiner L. R.** Improvements in Connected Digit Recognition Using Higher Order Spectral and Energy Features. In *Proc. Int. Conf. Acoust. Speech & Signal Processing*, pages 349–352, Toronto, Canada, 1991.

[95] **Wilpon J. G.** and **Rabiner L. R.** A Modified K-Means Clustering Algorithm for Use in Isolated Word Recognition. *IEEE Trans. on Acoustics, Speech and Signal Processing*, ASSP-33(3):587–594, June 1985.

[96] **Winograd T.** *Language as a Cognitive Process*, volume I Syntax. Addison-Wesley Publishing Company, 1983.

[97] **Zadeh L. A.** Fuzzy Sets. *Information Control*, 8:338–353, 1965.

[98] **Zimmermann H.J.** . *Fuzzy Set Theory and Its Applications*. Kluwer Academic Publishers, 1991. second edition.