2003

Phoneme-based speech recognition using self-organizing map.

Xiaofeng Dong
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.
Abstract

Automatic speech recognition by machine is a challenging task for man-machine communications. Because speech waveform is nonlinear and variant, a speech recognition algorithm requires much intelligence and an ability to accommodate variations. In this thesis, a hybrid speech recognizer based on self-organizing map (SOM) and fuzzy neural network (FNN) is proposed. The SOM is used to obtain the optimal phoneme response patterns of speech signal by Viterbi search algorithm and the FNN is applied for the recognition matching of these 2D speech response patterns on the SOM to fulfill the speech recognition tasks. Experiment results show that this hybrid speech recognizer is a feasible approach and could provide meaningful recognition results for dependent speech recognition. This thesis also compares this hybrid speech recognizer with the Hidden Markov Model, analyzes two types of misclassification for independent speech recognition and provides some suggestions for future research.
Dedicated to my wife Mary for her love and support and to my son Daniel.
Acknowledgements

I would like to express my sincere gratitude to my thesis advisor Dr. H. K. Kwan, for his suggestions, guidance, support and encouragement throughout the course of this research work, and to acknowledge him for suggesting me to combine SOM and FNN as a 2-dimensional phoneme-response-sequence recognition system.

I wish to thank my department reader, Dr. C. H. Chen and my external reader, Dr. David Ting, for their valuable advice toward the fulfillment of the thesis work.

I would also like to thank all my classmates in the ISPLab who have given me support during my study and research.
# Table of Contents

Abstract ........................................................................ iii
Dedication ....................................................................... iv
Acknowledgements ................................................................. v
List of Abbreviations ............................................................ ix
List of Figures .................................................................. x
List of Tables .................................................................. xiii

Chapter 1 Introduction ......................................................... 1
  1.1 Background ............................................................... 1
  1.2 Applications of Speech Recognition Technology ....................... 2
  1.3 Motivation for the Research ............................................... 5
  1.4 Organization of the Thesis ............................................... 6

Chapter 2 Review of Speech Recognition .................................. 8
  2.1 Fundamental of Speech Recognition ....................................... 8
    2.1.1 Fundamental Concepts .............................................. 8
    2.1.2 Speech Recognition Difficulties .................................. 12
  2.2 Hidden Markov Model ..................................................... 13
    2.2.1 Basic Concepts ....................................................... 13
    2.2.2 Algorithm ........................................................... 15
    2.2.3 Limitation of HMMs ................................................. 18
Chapter 5 Hybrid Speech Recognizer by SOM and FNN ........................................... 52
  5.1 Speech Recognition Application by SOM ......................................................... 52
  5.2 Hybrid Digit Speech Recognizer ........................................................................ 59
    5.2.1 Structure ........................................................................................................ 59
    5.2.2 Strategy of Speech Phonemes Feature Extraction ........................................... 63
    5.2.3 Learning Process of Phonemes on SOM ....................................................... 69
    5.2.4 Optimal Response Patterns of Speech Signal on SOM by Viterbi Algorithm ... 71
    5.2.5 Training and Recognition Process by FNN .................................................. 81
  5.3 Speech Database ................................................................................................. 82
  5.4 Simulation and Results ....................................................................................... 83
  5.5 Analysis of the Experiment Results .................................................................... 88
  5.6 Comparison with Hidden Markov Model based on SOM ................................... 94

Chapter 6 Conclusion and Suggestions for Future Work ......................................... 103
  6.1 Conclusion .......................................................................................................... 103
  6.2 Suggestions for Future Work ............................................................................. 104

References .................................................................................................................. 106

Vita Auctoris .............................................................................................................. 109
List of Abbreviations:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>ASR</td>
<td>Automatic Speech Recognition</td>
</tr>
<tr>
<td>FN</td>
<td>Fuzzy Neuron</td>
</tr>
<tr>
<td>FNN</td>
<td>Fuzzy Neural Network</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov Model</td>
</tr>
<tr>
<td>LVQ</td>
<td>Learning Vector Quantization</td>
</tr>
<tr>
<td>MFCC</td>
<td>Mel-Frequency Cepstral Coefficients</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi-Layer Perceptron</td>
</tr>
<tr>
<td>SOM</td>
<td>Self-Organizing Map</td>
</tr>
<tr>
<td>SOFM</td>
<td>Self-Organizing Feature Map</td>
</tr>
<tr>
<td>SOPM</td>
<td>Self-Organizing Phoneme Map</td>
</tr>
<tr>
<td>VQ</td>
<td>Vector Quantization</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2.1</td>
<td>Structure of standard speech recognition system</td>
</tr>
<tr>
<td>2.2</td>
<td>Acoustic model: template and state representation for the word 'cat'</td>
</tr>
<tr>
<td>2.3</td>
<td>A simple Hidden Markov Model</td>
</tr>
<tr>
<td>2.4</td>
<td>A hierarchically structure HMM</td>
</tr>
<tr>
<td>2.5</td>
<td>The forward pass recursion</td>
</tr>
<tr>
<td>3.1</td>
<td>Schematic view of the human speech apparatus</td>
</tr>
<tr>
<td>3.2</td>
<td>Cross-section of human ear</td>
</tr>
<tr>
<td>3.3</td>
<td>Time-domain waveforms for three talkers saying the word 'zero'</td>
</tr>
<tr>
<td>3.4</td>
<td>Spectrograms for three talkers saying word 'zero'</td>
</tr>
<tr>
<td>3.5</td>
<td>Spectral analysis of the speech signal</td>
</tr>
<tr>
<td>3.6</td>
<td>Mel filtering of the power spectrum</td>
</tr>
<tr>
<td>4.1</td>
<td>An illustrative example of the probability density functions</td>
</tr>
<tr>
<td></td>
<td>of the classes substantially overlap</td>
</tr>
<tr>
<td>4.2</td>
<td>Four-layer feedforward FNN</td>
</tr>
<tr>
<td>4.3</td>
<td>Fuzzification function (β=0.3)</td>
</tr>
<tr>
<td>5.1</td>
<td>An example of a phoneme map</td>
</tr>
<tr>
<td>5.2</td>
<td>A simplified configuration of the system</td>
</tr>
<tr>
<td>5.3</td>
<td>Self-Organizing Phoneme Map and Training</td>
</tr>
<tr>
<td>5.4</td>
<td>Structure of Fuzzy Neural Network</td>
</tr>
<tr>
<td>5.5(a)</td>
<td>Waveform of the phoneme /t/</td>
</tr>
<tr>
<td>5.5(b)</td>
<td>14 MFCC curve of five frames extracted from phoneme /t/</td>
</tr>
</tbody>
</table>
Figure 5.6(a): Waveform of the phoneme /ou/ .................................................66
Figure 5.6(b): 14 MFCC curve of 16 frames extracted from phoneme /ou/ ...............66
Figure 5.7: 3 continuous and 5 separated frames feature extraction
of short and long phonemes ..............................................................................67
Figure 5.8: short and long phoneme maps learning and LVQ ..................................70
Figure 5.9: Fine-tuned self-organizing short and long phoneme maps
(only main phoneme domains show) .................................................................70
Figure 5.10: Speech signal response on phoneme maps ........................................71
Figure 5.11: Two maximum response patterns of ‘zero’ by same person ..............73
Figure 5.12: Two maximum response pattern of ‘six’ by same person .................74
Figure 5.13: The phoneme sequence of ‘zero’ hidden in the response sequence ....75
Figure 5.14: The phoneme sequence of ‘six’ hidden in the response sequence ......75
Figure 5.15: 19 phoneme trellis along voice frames sequence ..............................77
Figure 5.16: Transition value defined between phonemes ....................................77
Figure 5.17: Two optimal response patterns of ‘zero’ in Figure 5.11
after viterbi algorithm .......................................................................................79
Figure 5.18: Two optimal response patterns of ‘six’ in Figure 5.11
after viterbi algorithm ......................................................................................80
Figure 5.19: Dependent Recognition rate of each digit of 240 digit samples
(α=2.0; β=0.3 for FNN) .................................................................86
Figure 5.20: Independent Recognition rate of each digit of 240 digit samples
(α=2.0; β=0.3 for FNN) ...............................................................................87

xi
Figure 5.21: Wrong phoneme sequence (WPS) for testing pattern ‘two’ ............................................ 89
Figure 5.22: Pattern Mixing Up (PMU) for testing pattern ‘three’ .................................................... 91
Figure 5.23: Pattern Mixing Up (PMU) for testing pattern ‘six’ ......................................................... 93
Figure 5.24: The Markov generation mode of word ‘zero’ ................................................................. 95
Figure 5.25: Isolated word recognition .................................................................................................. 97
Figure 5.26(a): Viterbi calculation for first speech frame ....................................................................... 98
Figure 5.26(b): Viterbi calculation for Second speech frame ................................................................. 99
Figure 5.26(c): Viterbi calculation for Nth speech frame ....................................................................... 99
Figure 5.26(d): Viterbi calculation for whole speech frames .................................................................100
Figure 5.27: Viterbi search path for ‘zero’ and ‘nine’ .............................................................................101
List of Tables

Table 5.1: Phoneme component for all the 10 digit voices ........................................... 68
Table 5.2: 13 short phoneme and 7 long phonemes ..................................................... 68
Table 5.3: Dependent Recognition Rate of FNN with different $\alpha$ and $\beta$ ($T_f=0.4$)
by 240 2D response patterns by viterbi algorithm .................................................. 84
Table 5.4: Dependent Recognition Rate of FNN with different $\alpha$ and $\beta$ ($T_f=0.4$)
by 240 2D response patterns without viterbi algorithm .......................................... 85
Table 5.5: Independent Recognition Rate of FNN with different $\alpha$ and $\beta$ ($T_f=0.4$)
by 240 2D response patterns by viterbi algorithm .................................................. 87
Table 5.6: Two types of misclassifications over 10 digits for all the 21
misclassifications in dependent digit speech recognition by FNN .......................... 93
Table 5.7: Two types of misclassifications over 10 digits for all the 55
misclassifications in independent digit speech recognition by FNN ...................... 94
Chapter 1

Introduction

1.1 Background

In its most general form, speech recognition is the conversion of an acoustic waveform into the text equivalent of the information being conveyed by the spoken word. Then ultimate goal in most research into speech recognition systems has been to develop a machine that has the ability to understand continuous speech with an unlimited vocabulary. This goal remains, as yet, unfulfilled although major developments have been made in the field of speech recognition.

There are many advantages to the use of speech which can be gained with the development of effective recognition systems which can be used for communication between man and the machines. When speech communications is used, problems can usually be solved faster, the hands and eyes are left free; the user is able to move about the room and still remain in contact with the machine and communication can take place over the telephone without the need for additional equipment such as modems.

Man-machine communication has been dominated by typing which is an accurate but relatively slow method of communication. Writing is a faster form of communication for most people (i.e. untrained typists) but has a major disadvantage for machine recognition.
in that a machine would have to be able to recognize a large variation in form for most letters. Speech has potential as the faster form of man-machine communication. It is the most natural form of human communication and for the majority is learned from early childhood. It is also faster than both typing and handwriting although it shares a major disadvantage with handwriting recognition. There exists such a wide variation in the way different letters and words can be acceptably pronounced that the machine would have to be able to recognize all these variations to achieve the ultimate goal in speech recognition.

1.2 Applications of Speech Recognition Technology

Initially it was believed that speech would be recognized using the information contained within the acoustic signal. It was realized that simple matching of acoustic patterns was limited since the same word, spoken on different occasions, differed in duration and intensity even when spoken by the same person. When the same word is spoken by different speakers the speech signal will also differ in frequency content.

Since the mid 1970s research in the field of speech recognition has increased dramatically. Many mathematical based algorithms have been developed and a number of speech recognition systems are now commercially available. The greatest success has been achieved using pattern recognition approaches to speech recognition by machine. This approach is based on the idea that if a system has “seen” a pattern then it can recognize it. These systems are trained to recognize patterns and require an adequate training set and a suitable pattern recognition model.
Template matching techniques match the input pattern to a set of templates and the category to which the input pattern belongs is determined using a similarity measure between the template and input pattern. A major advance was made with the introduction of dynamic programming algorithms introduced initially in 1970. Dynamic programming (DP) is a mathematical concept and the theory behind it is based on the principle of optimality which is a simple property of multi-stage decision processes.

An algorithm based on stochastic models is hidden Markov modeling (HMM) in which a generative model of each class is built. Therefore, in the HMM approach, the reference pattern is represented by a model as opposed to the pattern itself. An HMM consists of two parts; a finite state Markov chain and a finite set of probability distribution [1]. The finite state machine comprised of a number of states and the probabilities of transition from one state to another. The output probabilities consist of the probabilities that, for a particular state transition, a certain output will be observed. An observer will see the outputs produced but not the states and transitions as there are “hidden”. Recognition of the input patterns is achieved by determining which model has the greatest probability of generating the output sequence observed. One method of find the mostly likely state model is the Viterbi algorithm which is a stochastic version of dynamic programming [2]. Many new developments in HMMS have taken place and since then HMMs are the most widely used speech recognition tool in state-of-the-art systems.

In the 1980s multi-layer perceptrons (MLP) were recognized as being a potentially suitable speech recognition tool. An MLP, also known as neural network, is a general
pattern recognition mechanism which can learn to discriminate between different categories of speech signals by “seeing” examples during training.

In order to take into account contextual information, the time delay neural network (TDNN) approach has been adopted in some research. An example of this work is Waibel et al. who used TDNNs for the recognition of the phonemes “B”, “D” and “G” [3][4]. Waibel, Sawai and Shikano used TDNNs in a speech recognition system where smaller networks were trained to learn phonemic subcategories [5]. These subcategory networks were then used to “grow” larger networks without any degradation in performance. Several techniques for growing these networks without degrading performance, incurring large training times or requiring more data, were proposed.

An alternative neural network approach is the recurrent neural network also referred to as a dynamic neural network [6]. In this method the contextual information does not need to be used explicitly at the input to the neural network. Instead, a feedback delay loop is added to each neuron in the hidden and output layers. Contextual information can be used explicitly by using a network like that of a TDNN and having feedback loops on the hidden and output layer neurons. Another alternative architecture for recurrent neural networks is to feedback the delayed hidden neuron output values to the input layer [7]. A modification to the error back propagation algorithm is required to allow the looped signals to reach a steady state [6][8].
Neural networks can also be used to implement the algorithms used in conventional speech recognition systems [9]. During unsupervised training for self organizing using Kohonen’s algorithm, continuous valued input vectors are presented to the network sequentially and no desired output vectors are specified [10]. The weights will organize such that topologically close nodes become sensitive to similar inputs forming feature maps which associate different clusters with different output nodes [11]. This acts like a neural network implementation of a k-nearest neighbor classifier. A Viterbi net is a neural network implementation of a Viterbi decoder used in HMM recognizer [9].

1.3 Motivation for the Research

During the latest 20 years, however, several new research areas in computer science have been founded that focus on solving problems by mimicking nature. A number of new techniques and concepts have thus been introduced, Artificial Neural Network and Fuzzy techniques being prime examples. This has in turn led to a number of novel approaches to the problems of creating speech understanding and speaking machines, and while the level of expectation to these machines have somewhat declined, results obtained by applying the new techniques are promising.

In the late 1980’s, the Finnish computer scientist Teuvo Kohonen presented a specialized version of an artificial neural network - the Self-Organizing Map [12], SOM has the special property the internal representation of information are generally organized spatially, just like the topographically organized maps found in the brain. So how to use the spatially organized maps of information representation for the speech processing application is an interesting and challenging work.
Kohonen also presents an application with SOM that in some way realized the old dream: A “typewriter” that was controlled not by a keyboard, but by the human voice. The device was capable of classifying the sounds found in Finnish speech into text with an accuracy of about 97%. Unfortunately, one of the keys to achieving the high accuracy is the fact that Finnish possesses a number of acoustic properties that make it especially well suited for the task. – the typewriter is not easily generalized to an arbitrary language.

Most of the speech applications of SOM are only focused on phoneme recognition and phoneme transcription because there is lack of effective ways to deal with 2D spatial information of SOM. In this thesis, a novel fuzzy neural network (FNN) is introduced to apply to the phoneme and word recognition based on the phoneme maps. This FNN will fully use the 2D spatial information of phoneme maps which are the spatial responses of speech signal and broaden the research perspectives of speech recognition by SOM.

1.4 Organization of the Thesis

In chapter 2, a literature survey is reviewed on speech recognition. It gives an introduction to the fundamental concepts and difficulties of speech recognition, and the dominant Hidden Markov Model (HMM) for speech recognition is described in details about the basic concepts, algorithms and the limitations.

Chapter 3 describes the production and perceptron of speech signal, and the algorithm for speech feature extraction is also presented, which is the first step in the whole process of speech recognition.
Chapter 4 gives us a clear understand of how the neural network technologies were developed and the more detailed will be focused on the Self-Organizing Maps (SOM) and Fuzzy Neural Network (FNN) which will be used in our speech recognizer.

In Chapter 5, the purpose of using SOM and FNN in our speech recognizer will be presented, and the structure and algorithm of hybrid digit speech recognizer is also described in details. It also includes the recognition results and analysis.

Chapter 6 gives the conclusion and suggestions for future research.
Chapter 2

Review of Speech Recognition

2.1 Fundamental of Speech Recognition

In this chapter we will present a brief review of the field of speech recognition. After reviewing some fundamental concepts and recognition approaches, we will discuss Hidden Markov Models in some detail, offering a summary of the algorithm, variations, and limitations that are associated with this dominant technology.

2.1.1 Fundamental Concepts

Speech recognition is a multileveled pattern recognition task, in which acoustical signals are examined and structured into a hierarchy of subword units (e.g., phonemes), words, phrases, and sentences. Each level may provide additional temporal constraints, e.g., known word pronunciations or legal word sequences, which can compensate for errors or uncertainties at lower levels. This hierarchy of constraints can best be exploited by combining decision probabilistically at all lower levels, and making discrete decisions only at the highest level.

The structure of a standard speech recognition system is illustrated in Figure 2.1. The elements are as follows:
- **Raw speech.** Speech is typically sampled at a high frequency, e.g., 16 KHz over a microphone or 8 KHz over a telephone. This yields a sequence of amplitude values over time.

![Diagram of speech recognition system]

Figure 2.1 Structure of standard speech recognition system

- **Signal analysis.** Raw speech should be initially transformed and compressed, in order to simplify subsequent processing. Many signal analysis techniques are available which can extract useful features and compress the data by a factor of ten without losing any important information. Among the most popular:

  1. Fourier analysis (FFT) yields discrete frequencies over time, which can be interpreted visually. Frequencies are often distributed using a Mel scale, which is linear in the low range but logarithmic in the high range, corresponding to physiological characteristics of the human ear.

  2. Perceptual Linear Prediction (PLP) is also physiologically motivated, but yields coefficients that cannot be interpreted visually.
(3) Linear Predictive Coding (LPC) yields coefficients of a linear equation that approximate the recent history of the raw speech values.

(4) Cepstral analysis calculates the inverse Fourier transform of the logarithm of the power spectrum of the signal.

In practice, it makes little difference which technique is used. Afterward, procedure such as Linear Discriminate Analysis (LDA) may optionally be applied to further reduce the dimensionality of any representation, and to decorrelate the coefficients.

- **Speech frames.** The result of signal analysis is sequence of speech frames, typically at 10 msec intervals, with about 16 coefficients per frame. These frames may be augmented by their own first and/or second derivatives, providing explicit information about speech dynamics; this typically leads to improved performance. The speech frames are used for acoustic analysis.

- **Acoustic models.** In order to analyze the speech frames for their acoustic content, we need a set of acoustic models. There are many kinds of acoustic models, varying in their representation, granularity, context dependence, and other properties.

Figure 2.2 shows two popular representations for acoustic models. The simplest is a template, which is just a stored sample of the unit of speech to be modeled, e.g., a recording of a word. An unknown word can be recognized by simply comparing it against all known templates, and finding the closest match. Templates have two major drawbacks: (1) they cannot model acoustic variability, except in a coarse way by assigning multiple templates to each word; and (2) in practice they are limited to whole-word models, because it’s hard to record or segment a sample shorter than a word.
A more flexible representation, used in larger systems, is based on trained acoustic models, or states. In this approach, every word is modeled by a sequence of trainable states, and each state indicates the sounds that are likely to be heard in that segment of the word, using a probability distribution over the acoustic space. Probability distributions can be modeled parametrically, by assuming that they have a simple shape (e.g., a Guassian distribution) and then trying to find the parameters that describe it; or non-parametrically, by representing the distribution directly.

Figure 2.2 Acoustic model: template and state representation for the word ‘cat’

- **Acoustic analysis and frame scores.** Acoustic analysis is performed by applying each acoustic model over each frame of speech, yielding a matrix of frame scores. Scores are computed according to the type of acoustic model that is being used. For template-based acoustic models, a score is typically the Euclidean distance between a template’s frame and an unknown frame. For state-based acoustic models, a score represents an emission probability, i.e., the likelihood of the current state generating the current frame, as determined by the state’s parametric or non-parametric functions.
2.1.2 Speech Recognition Difficulties

Because speech waveform is nonlinear and dynamic, speech recognition is an inherently difficult task. There are several main variabilities of speech signal including within-speaker variability, across-speaker variability, transducer and transmission variability, language complexity, and the environmental conditions under which a speaker is talking.

*Within-speaker variability* is caused by inconsistent pronunciation, speaking speed and different emotions when the words or phrased are spoken by same speaker.

*Across-speaker variability* is due to the physiological differences, regional accents, foreign languages, etc. The physiological correlates are associated with the size and configuration of the components of the vocal tract of each individual. The variations in the vocal tract can cause different resonance frequencies (formants) and pitch frequency of the same words.

*Transducer and transmission variability* is because the words are spoken over different microphone/handsets and the speech signal could be transmitted by all kind of communication systems (telecommunication networks, cellular phones, etc), in which unexpected noises are introduced into the signal.

*Language complexity* makes speech recognition an extremely difficult job. So far, the task of speech recognizers is simplified by limiting the number of possible utterances by the imposition of semantic constraints. On the other hand, we shall obey multi-
disciplinary natures of speech signal and be adaptive to the language complexity because speech is a completely natural activity of human beings.

Environmental condition is also a main concern of speech recognizers while real applications usually are conducted in adverse conditions which may drastically degrade the system performance. Therefore, it is necessary to present robust recognition methods for dealing with reasonable noise or distortions of the speech signal.

2.2 Hidden Markov Model

The most flexible and successful approach to speech recognition so far has been Hidden Markov Models (HMMs). In this section we will present the basic concepts of HMMs, describe the algorithms for training and using them, discuss some common variations, and review the problems associated with HMMs.

2.2.1 Basic Concepts

A hidden Markov Model is a collection of states connected by transitions, as illustrated in Figure 2.3. It begins in a designated initial state. In each discrete time step, a transition is taken into a new state, and then one output symbol is generated in that state. The choice of transition and output symbol are both random, governed by probability distribution. The HMM can be though of as a black box, where the sequence of output symbols generated over time observable, but the sequence of states visited over time is hidden from view. This is way it's called a Hidden Markov Model.
HMMs have a variety of applications. When a HMM is applied to speech recognition, the states are interpreted as acoustic models, indicating what sounds are likely to be heard during their corresponding segments of speech; while the transitions provide temporal constraints, indicating how the states may follow each other in sequence. Because speech always goes forward in time, transitions in a speech application always go forward (or make a self-loop, allowing a state to have arbitrary duration). Figure 2.4 illustrates how states and transitions in an HMM can be structured hierarchically, in order to represent phonemes, words, and sentences.

Figure 2.4 a hierarchically structure HMM
Formally, a HMM consists of the following elements:

\[ \{s\} = \text{A set of states} \]

\[ \{a_{ij}\} = \text{A set of transition probabilities, where } a_{ij} \text{ is the probability of taking the} \]

transition from state \( i \) to state \( j \).

\[ \{b_i(u)\} = \text{A set of emission probabilities, where } b_i \text{ is the probability distribution} \]

over the acoustic space describing the likelihood of emitting each possible sound \( u \) while

in state \( i \).

Since \( a \) and \( b \) are both probabilities, they must satisfy the following properties:

\[ a_{ij} \geq 0, \ b_i(u) \geq 0, \ \forall_{i,j,u} \]

\[ \sum_j a_{ij} = 1, \ \forall_i \]

\[ \sum_u b_i(u) = 1, \ \forall_i \]

In using this notation we implicitly confine our attention to First-Order HMMs, in which

\( a \) and \( b \) depend only on the current state, independent of the previous history of the state

sequence. This assumption, almost universally observed, limits the number of trainable

parameters and makes the training and testing algorithms very efficient, rendering HMMs

useful for speech recognition.

2.2.2 Algorithm

There are three basic algorithms associated with Hidden Markov Models:

- The forward algorithm, useful for isolated word recognition;
- The Viterbi algorithm, useful for continuous speech recognition; and
- The forward-backward algorithm, useful for training an HMM.
In this section we will focus on the first two algorithms used in this thesis.

*The Forward Algorithm*

In order to perform isolated word recognition, we must be able to evaluate the probability that a given HMM word model produced a given observation sequence, so that we can compare the scores for each word model and choose the one with the highest score. More formally: given an HMM model $M$, consisting of $\{s\}$, $\{a_i\}$, and $\{b_i(u)\}$, we must compute the probability that it generated the output sequence $y_1^T = (y_1, y_2, y_3, \ldots, y_T)$. Because every state $i$ can generate each output symbol $u$ with probability $b_i(u)$, every state sequence of length $T$ contributes something to the total probability. A brute force algorithm would simply list all possible state sequences of length $T$, and accumulate their probabilities of generating $y_1^T$; but this is clearly an exponential algorithm, and is not practical.

A much more efficient solution is the *Forward Algorithm*, which is an instance of the class of algorithms known as *dynamic programming*, requiring computation and storage that are only linear in $T$. First, we define $\alpha_j(t)$ as the probability of generating the partial sequence $y_1^t$, ending up in state $j$ at time $t$. $\alpha_j(t = 0)$ is initialized to 1.0 in the initial state, and 0.0 in all other states. If we have already computed $\alpha_j(t-1)$ for all $i$ in the previous time frame $t-1$, then $\alpha_j(t)$ can be computed recursively in terms of the incremental probability of entering state $j$ from each $i$ while generating the output symbol $y_t$ (Figure 2.5):

$$a_j(t) = \sum_i a_i(t-1)a_y b_j(y_t)$$
If $F$ is the final state, then by induction we see that $\alpha_F(T)$ is the probability that the HMM generated the complete output sequence $y_1^T$.

![Diagram](image)

Figure 2.5 the forward pass recursion

**The Viterbi Algorithm**

While the Forward Algorithm is useful for isolated word recognition, it cannot be applied to continuous speech recognition, because it is impractical to have a separated HMM for each possible sentence. In order to perform continuous speech recognition, we should instead infer the actual sequence of states that generated the given observation sequence; from the state sequence we can easily recover the word sequence. Unfortunately the actual state sequence is hidden (by definition), and cannot be uniquely identified; after all, any path could have produced this output sequence, with some small probability. The best we can do is to find the one state sequence that was most likely to have generated the observation sequence. As before, we could do this by evaluating all possible state sequences and reporting the one with the highest probability, but this would be an exponential and hence infeasible algorithm.
A much more efficient solution is the Viterbi Algorithm, which is again based on dynamic programming. It is very similar to the Forward Algorithm, the main different being that instead of evaluating a summation at each cell, we evaluate the maximum:

\[ v_j(t) = \text{MAX}[v_i(t-1)a_{ij}b_j(y_i)] \]

This implicitly identified the single best predecessor state for each cell in the matrix. If we explicitly identify that best predecessor state, saving a single backpointer in each cell in the matrix, then by the time we have evaluated \( v_F(T) \) at the final time frame, we can retrace those backpointers from the final cell to reconstruct the whole state sequence. Once we have the state sequence, we can trivially recover the word sequence.

2.2.3 Limitation of HMMs

Despite their state-of-the-art performace, HMMs are handicapped by several well-known weaknesses, namely:

- The First-Order Assumption – which says that all probabilities depend solely on the current state – is false for speech applications. One consequence is that HMMs have difficulty modeling coarticulation, because acoustic distributions are in fact strongly affected by recent state history. Another consequence is that duration are modeled inaccurately by an exponentially decaying distribution, rather than by a more accurate Poisson or other bell-shaped distribution.

- The Independence Assumption – which says that there is no correlation between adjacent input frames – is also false for speech application. In accordance with this assumption, HMMs examine only one frame of speech at a time. In order to
benefit from the context of neighboring frames, HMMs must absorb those frames into the current frame (e.g., by introduction multiple streams of data in order to exploit delta coefficients, or using LDA to transform these streams into a single stream.

- The HMM probability density models (discrete, continuous, and semi-continuous) have suboptimal modeling accuracy. Specifically, discrete density HMMs suffer from quantization errors, while continuous or semi-continuous density HMMs suffer from model mismatch, i.e., a poor match between their a priori choice of statistical model (e.g., a mixture of $K$ Gaussians) and the true density of acoustic space.

- The Maximum Likelihood training criterion leads to poor discrimination between the acoustic models (given limited training data and correspondingly limited models).

Because HMMs suffer from all these weaknesses, they can obtain good performance only by relying on context dependent phone models, which have so many parameters that they must be extensively shared --- and this, in turn, calls for elaborate mechanisms such as senones and decision trees.
Chapter 3

Speech Signal Feature Extraction and Representation

3.1 Introduction to Speech Sounds

3.1.1. *Speech Production*

Speech sound is produced by a set of well-controlled movement of various speech apparatus. Figure 3.1 shows a schematic cross-section through the vocal tract of the apparatus.

![Diagram of the human speech apparatus]

*Figure 3.1 Schematic view of the human speech apparatus*
The vocal tract is a primary acoustic tube, which is the region of the mouth cavity bounded by the vocal cords and the lips. As air is expelled from the lungs, the vocal cords are tensed and then caused to vibrate by the airflow. The frequency of oscillation is called the fundamental frequency, and it depends on the length, tension and mass of the vocal cords. During this process, the shape of the vocal tube is changed by different positions of the velum, tongue, jaw and lips. The average length of the vocal tract for an adult male is about 17cm, and its cross-section area can vary in its outer section from 0 to about 20cm. Therefore, the vocal tract, as an acoustic resonator, will determine variable resonant frequencies by adjusting the shape and size of the vocal tract. The resonant frequency is called the formant frequency or simply formant. The nasal tract is an auxiliary acoustic tube that can be acoustically cooperated with vocal tract to produce nasal sounds.

Various speech sounds are produced not only by adjusting the shape of the vocal tract, but also the type of excitation. Besides the airflow from the lung, the excitation could come from some other sources: the fricative excitation, plosive excitation and whispered excitation [13].

3.1.2. Speech Perception

As the vocal system can produce speech sounds, the auditory system is capable of detecting the change in air pressure of audible sounds [14]. Figure 3.2 shows a cross-section diagram of human ear. The ear consists of three parts: the outer ear, the middle ear, and the inner ear. The outer ear collects the sound waves and passes the air pressure variation to the eardrum. The middle ear is an air-filled cavity, which serves as a
mechanical amplifier and transforms vibration of the eardrum into oscillations of the fluid filled inner ear. The inner ear then converts the mechanical vibrations into electrical potentials that go to the auditory nerve and the cortex.

The human ear is most sensitive to frequencies of the range from 1000 to 4000Hz. Most speech information is covered within these frequencies. It is shown by experiments that human ears are largely phase insensitive. The basilar membrane is only deformed when the stapes pushes on the oval window [15], thus very little information is available for the brain to determine the waveform's phase. This fact could be applied to speech recognition to reduce the amount of data in the encoded waveform.

![Figure 3.2 Cross-section of human ear](image)

3.1.3 *Speech Features*

The speech recognition can be divided into two processes: feature extraction and pattern recognition. Feature extraction is responsible for searching the speech characteristics and storing them for the second process: pattern recognition. In order to identify the speech
characteristics accurately and efficiently, it is necessary to investigate the features and classification of speech sounds.

Any natural language, including English, is based on a set of distinguishable and mutually exclusive primary units, which are called phonemes. All the phonemes are related to different articulatory gestures of a language.

There are several ways to classify speech sounds [14][15]. According to the type of excitation source of phonemes, speech sounds can be classified into the following categories:

- Voiced sounds (/a/, /d/) occur when air pressure pushes the vocal cords open and causes them to vibrate. The vibrating cords modulate the air stream from the lungs at a rate that could be as low as 60 times per second for some males to 500 times per second for children. The peak amplitude of voiced sound is much higher than that of the unvoiced sound.

- Nasal sounds such as /m/, /n/ are also voiced. However, the nasal cavity is involved together with the vocal cavity during the utterance. Part of the airflow is diverted into the nasal tract by opening the velum.

- Fricatives are generated by exciting the vocal tract with turbulent flow created by airflow through a narrow constriction. For example, the sound /f/, /s/ and /sh/ are fricatives.
• Voiced fricatives occur when the vocal tract is excited simultaneously by both turbulence flow and vocal vibration. The sounds /z/, /zh/ and /v/ belong to this category.

• Plosives are produced by exciting the vocal tract with a rapid release of pressure by the constriction of lips or teeth. The plosives /t/, /k/ are voiceless, while /b/, /d/ are voiced.

• Affricative sounds are produced by gradually releasing a completely closed and pressurized vocal tract.

• Whispered sounds are excited by airflow rushing through a small triangular opening between the arytenoids cartilages at the rear of the nearly closed vocal folds.

3.2. Speech Feature Extraction

3.2.1 Introduction

During feature extraction, the input speech samples are transformed into a domain where information of importance for recognition is preserved and redundant information is discarded. This signal processing stage is usually quite computationally intensive.

To give some idea of the variability of speech signals, Figure 3.3 shows time-domain speech waveforms for three speakers saying the word ‘zero’. These waveforms are of similar duration but with no obvious common pattern.
Patterns in the speech signal become much more evident when the signal is transformed into the frequency domain. Figure 3.4 illustrates corresponding spectrograms – the vertical axis denotes frequency and the horizontal axis denotes time. The power of the frequency components is proportionally represented by darkness. The resonant frequencies of the vocal tract (formants) appear clearly as the dark bands flowing across each spectrogram.
Figure 3.4 Spectrograms for three talkers saying word ‘zero’

The typical process of frequency analysis of the sampled speech signal is illustrated in Figure 3.5. The samples are first assembled and windowed into overlapping frames. The overlap period is usually chosen to be in the range of 10 to 20ms during which speech is assumed to be quasistationary. A fast Fourier transform (FFT) is used to calculate the power spectrum.
3.2.2 *Cepstral Coefficients (MFCC)*

The power spectrum is then organized into frequency bands according to a series of Mel scale filters as shown in Figure 3.6. These filters are spaced linearly to 1 kHz and then logarithmically up to the maximum frequency. The spacing of these bands is based on measurements of the sensitivity of the human ear to changes in frequency. In this way the power spectrum can be represented by about 20 Mel scale filter outputs – considerably reducing the data rate.

The dynamic range of the power spectrum is quite large and hence the logarithm is taken of the Mel filter outputs. This accord with human perception of sound intensity which is thought to vary with the logarithm of intensity.
Finally, a discrete cosine transform (DCT) is performed on the logged Mel filter outputs. This is given as:

\[ C(k) = \sum_{i=0}^{N-1} f(i) \cos \left( \frac{2\pi ki}{N} + 0.5 \right) \quad k \in [0, M] \]

Where \( C(k) \) is the \( k \)th DCT output and \( f(i) \) is the \( i \)th of \( N \) log filter bank outputs. The DCT output is known as the cepstrum. Two important functions are served by this transform. First, it acts as a data reduction stage. The power spectrum envelop varies slowly over the frequency range and so \( M \) is usually much less than \( N \) in above equation. Secondly, the
DCT outputs are relatively uncorrelated so that each output value can be assumed to be independent of every output value. This is ideal for most types of pattern matching, as the vector operations for training and recognition can be greatly simplified.

Each feature vector contains a subset of the cepstral coefficients. In addition, the time derivative of the cepstral coefficients computed over successive, non-overlapping, frame is often included. Similarly, the different logarithm of frame energies is also included. The final feature vector may consist of:

\[
\begin{bmatrix}
C0 \\
..... \\
CM \\
\Delta C0 \\
..... \\
\Delta CM \\
\Delta \log \text{energy}
\end{bmatrix}
\]
Chapter 4

Neural Network Technologies for Speech Recognition Research

4.1 Introduction

Neural networks are, in essence, biologically inspired networks since they are based on the current understanding of the biological nervous system. In essence they are comprised of a network of densely interconnected simple processing elements which perform in a manner analogous to the most elementary functions of a biological neuron.

A brief history of the development of neural networks and a basic introduction to their theory is outlined in this chapter, and Self-Organizing Maps and its application are discussed and the Fuzzy Neural Network is described in detail.

4.1.1 The Perceptron

The idea of the simple neuron model first emerged in the 1940s with the work of McCulloch and Pitts [16]. The cybernetics movement which ensued attempted to combine biology, psychology, engineering and mathematics resulting in architectures for neurons which would perform a number of tasks. In 1949, Hebb's book [17] put forward the theory of neural network developing "internal representations" related to experience.
In the 1950s, research continued initially into the development of networks to perform specific tasks but this changed and the goal became to develop machines that could learn. By the end of the decade there had been a lack of significant developments and work in this field diminished considerably.

In the 1960s, interest was revived with the publication of a book by Rosenblatt [18] where he defined the concept of the perceptrons and laid down many theories about them. It was proved theoretically that a perceptron could learn to perform any task as long as it was possible to program it to do so.

4.1.2 The Multi-Layer Perceptron

Minsky and Papert had proposed a solution to the problem posed by functions such as the ex-or. They suggested that an extra layer of nodes with non-linear activation functions could be introduced. The output would now be a non-linear combination of the inputs so more complicated decision surfaces could be represented. The problem that remained was that no training algorithm was available to train such a network of perceptrons at the time.

During the 1970s more research turned towards the representation of knowledge and away from learning and many new ideas were developed. Then, in the 1980s, there was a resurgence of interest in neural networks and it was during this time that an effective
algorithm, called back propagation, for the training of multi-layer perceptron (MLP) structures was developed [19].

4.1.3 The Error Back Propagation Algorithm

Error back propagation is a gradient descent algorithm where weights and biases are adjusted to minimize a cost function equal to the mean square error in the network.

For a 3-layer neural network with $N$ input nodes and $M$ output nodes, the network’s weights are initially set to small random values. An input/output vector pair $p$ is presented to the network with input vector $x_{p0}, x_{p1}, ..., x_{pN-1}$, and target output vector $t_0, t_1, ..., t_{M-1}$. From this input vector an output vector is produced by the network which can then be compared to the target output vector. If there is no difference between the produced and target output vectors no learning takes place. Otherwise the weights are changed to reduce the difference. The weights are adapted using a recursive algorithm which starts at the output nodes and works back to the hidden layer.

4.1.4 The Fully Connected Neural Network

The most common form of neural network is the 3-layer, fully connected, feed forward MLP, the nodes are arranged in 3 layers; an input layer, a hidden layer and an output layer with inputs flowing in the forward direction from input layer to output layer through the hidden layer except during training. In this type of network, the inputs of every node in the hidden layer are connected to the outputs of every node in the input layer and the
inputs of every node in the output layer are connected to the outputs of every node in the hidden layer. The nodes in the input layer are used to monitor the external signals input to the neural network and the neurons in the output layer are used to make the final decision and transmit the signal produced to the outside world.

4.2 Self-Organizing Map

4.2.1 Background

The network architectures and signal processes used to model nervous systems can roughly be divided into three categories, each based on a different philosophy, Feedforward [19] networks transform sets of input signals into sets of output signals. The desired input-output transformation is usually determined by external, supervised adjustment of the system parameters. In feedback network [20], the input information defines the initial activity state of a feedback system, and after state transitions the asymptotic final state is identified as the outcome of the computation. In the third category, neighboring cells in a neural network compete in their activities by means of mutual lateral interactions, and develop adaptively into specific detectors of different detectors of different signal patterns. In this category learning is called competitive, unsupervised, or self-organizing.

The Self-Organizing Map discussed in this chapter belongs to the last category. It is a sheet-like artificial neural network, the cells of which becomes specifically tuned to various input signal patterns or classes of patterns through an unsupervised learning process. In the basic version, only one cell or local group of cells at a time gives the
active response to the current input. The location responses tend to become ordered as if some meaningful coordinate system for different input features were being created over the network. The spatial location or coordinates of a cell in the network then correspond to a particular domain of input signal patterns. Each cell or local cell group acts like a separate decoder for the same input. It is thus the presence or absence of an active response at that location, and not so much the exact input-output signal transformation or magnitude of the response, that provides an interpretation of the input information.

After a large number of observations, a fairly detailed organizational view of the brain has evolved [21]. Especially in higher animals, the various cortices in the cell mass seem to contain many kinds of "map" [22], such that a particular location of the neural response in the map often directly corresponds to a specific modality and quality of sensory signal. The field of vision is mapped "quasiconformally" onto the primary visual cortex. Some of the maps, especially those in the primary visual sensory areas, are ordered according to some feature dimensions of the sensory areas, are ordered according to some feature dimensions of the sensory signals; for instance, in the visual areas, there are line orientation and color maps, and in the auditory cortex there are the so-called tonotopic maps, which represent pitches of tones in terms of the cortical distance, or other auditory maps.

It seems as if the internal representations of information in the brain are generally organized spatially. Although there is only partial biological evidence for this, enough data are already available to justify further theoretical studies of this principle. Artificial
self-organizing maps and brain maps thus have many features in common, and what is even more intriguing, we now fully understand the processes by which such artificial maps can be formed adaptively and completely automatically.

4.2.2 Principle and Learning Algorithm

Several years ago, one of the authors (Kohonen) developed a model of neural adaptation that is capable of unsupervised formation of spatial maps for many different kinds of data. This section first summarizes the model equations and then explains how a structure-preserving map of hierarchically related data is generated by them. More detailed description of the process and its background can be found in the following original publications and some recent developments [21][22][23].

The model assumes a set of laterally interacting adaptive neurons, usually arranged as a two-dimensional sheet. The neurons are connected to a common bundle of input fibers. Any activity pattern on the input fibers gives rise to excitation of some local group of neurons. After learning, the spatial positions of the excited groups specified a mapping of the input patterns on to the two-dimensional sheet, the latter having the property of the a topographic map, i.e. it represents distance relation of the high-dimensional space of the input signals approximately as distance relationships on the two-dimensional neural sheet. This remarkable property follows from the assumed lateral interactions and a very simple, biologically justifiable adaptation law.
In fact, it seems that the main requirements for such self organization are that (1) the neurons are exposed to a sufficient number of different input, (2) for each input, the synaptic input connections to the excited group are only affected, (3) similar updating is imposed on many adjacent neurons, and (4) the resulting adjustment is such that it enhances the same responses to a subsequent, sufficiently similar input.

Mathematically, the activity pattern at the input is described by an n-dimensional real input vector \( x \), where \( n \) is the number of input lines. The responsiveness of neuron \( r \) is specified by an n-dimensional vector \( W_r \), eventually corresponding to the vector of synaptic efficacies, and it is measured by the dot product \( X \cdot W_r \).

The neurons are arranged in a two-dimensional lattice, and each neuron is labeled by its two-dimensional lattice position \( r \). The group of excited neurons is taken to be centered at the neuron \( s \) for which \( X \cdot W_s \) is maximal. Its extent and shape are described by a function \( h_{rs} \), whose value is the excitation of neuron \( r \), if the group center is at \( s \). This function may be constant for all \( r \) in a "neighborhood zone" around \( s \) and zero elsewhere, or bell-shaped, like in the present simulations that are supposed to describe a more natural mapping. In this case \( h_{rs} \) will be largest at \( r = s \) and decline to zero with increasing distance \( \| r - s \| \). A rather realistic modeling choice for \( h_{rs} \) is

\[
h_{rs} = \exp \left( -\frac{\| r - s \|^2}{\delta^2} \right)
\]

The learning algorithm for SOM can be described as follows:

1. Search the output unit whose incoming connection weights are the closest to the input pattern in term of Euclidian distance:

\[
d_j(t) = \sum_{i=0}^{n-1} (x_i(t) - w_{ij}(t))^2
\]
(2) Adjust connection weights of the winning unit and the adjacent output units in close proximity of the neighborhood of the winning, and get moved closer to input pattern.

\[ w_y(t + 1) = w_y(t) + h_{rn}(t)(x_i(t) - w_y(t)) \]

(3) Repeat above two steps during learning time.

(4) As learning progresses, the size of the neighborhood around the winning unit decreases. Initially, large number of output units will updated, and as the learning proceeds smaller and smaller numbers are updated, until at the end of the learning only the winning unit is adjusted. Similarly, the learning rate will decrease as learning progresses.

It may suffice to assert that the resulting maps are nonlinear projections of the input space onto this surface with the following two main properties: (1) the distance relationships between the source data are preserved by their images in the map as faithfully as possible. However, a mapping from a high dimensional space to a lower-dimensional one will usually distort most distances and only preserve the most important neighborhood relationships between the data items. (2) If different input vectors appear with different frequencies, the more frequent one will be mapped to larger domains at the expense of the less frequent ones. This results in a very economic allocation of memory resources to data items, and complies with physiological findings.
4.2.3 Learning Vector Quantization

If the Self-Organizing Map is to be used as a pattern classification in which the cells or their responses are grouped into subsets, each of which corresponds to a discrete class of patterns, then the problem becomes a decision process and must be handled differently. The original Map, like any classical Vector Quantization (VQ) method is mainly intended to approximate input signal values, or their probability density function, by quantized "codebook" vectors that are localized in the input space to minimize a quantization error functional.

On the other hand, if the signal sets are to be classified into a finite number of categories, then several codebook vectors are usually made to represent each class, and their identity within the classes is no longer important. In fact, only decision made at class borders count. It is then possible, as shown below, to define effective values for the codebook vectors such that they directly define near-optimal decision borders between the classes, even in the sense of classical Bayesian decision theory. These strategies and learning algorithms were introduced in [24], and called learning vector quantization (LVQ).

*Type one learning vector quantization (LVQ1)*

If several codebook vectors \( m_i \) are assigned to each class, and each of them is labeled with the corresponding class symbol, the class regions in the \( X \) space are defined by simple nearest-neighbor comparison of \( X \) with the \( m_i \); the label of the closest \( m_i \) defines the classification of \( X \).
To define the optimal placement of $m_i$ in an iterative learning process, initial values for them must first be set using any classical VQ method or by the Self-Organizing Map algorithm. The initial values in both cases roughly correspond to the overall statistical density function $P(x)$ of the input. The next phase is to determine the labels of the codebook vectors, by presenting a number of input vectors with known classification, and assigning the cells to different classes by majority voting, according to the frequency with which each $m_i$ is closest to the calibration vectors of a particular class.

The classification accuracy is improved if the $m_i$ is updated according to the following algorithm [24]. The idea is to pull codebook vectors away from the decision surfaces to demarcate the class borders more accurately. Let $m_c$ be the codebook vector closest to $X$ in the Euclidean metric; this then also defines the classification of $X$. Apply training vectors $X$ the classification of which is known. Update the $m_i = m_i(t)$ as follows:

$$m_c(t + 1) = m_c(t) + \alpha(t)[x(t) - m_c(t)]$$

if $X$ is classified correctly

$$m_c(t + 1) = m_c(t) - \alpha(t)[x(t) - m_c(t)]$$

if the classification of $X$ is incorrect,

$$m_i(t + 1) = m_i(t) \quad for \ i \neq c$$

Here $\alpha(t)$ is a scalar gain $(0 < \alpha(t) < 1)$, which is decreasing monotonically in time, as in earlier formulas. Since this is a fine-tuning method, one should start with a fairly small value, say $\alpha(0) = 0.01$ or $0.02$ and let it decrease to zero.
After training, the \( m_i \) will have acquired values such that classification using the "nearest neighbor" principle, by comparing of \( X \) with the \( m_i \), already rather coincides with that of the Bayes classifier. Figure 4.1 represents an illustrative example in which \( X \) is two-dimensional, and the probability density functions of the classes substantially overlap. The decision surface defined by this classifier seems to be near-optimal, although piecewise linear, and the classification accuracy in this rather difficult example in within a fraction of a percent of that achieved with the Bayes classifier.

![Figure 4.1 An illustrative example of the probability density functions of the classes substantially overlap](image)

### 4.3 Fuzzy Neural Network

#### 4.3.1 Introduction

A Neural Network (NN) has a massively parallel structure which is composed of many processing elements to each other through weights [25] [26] [27]. Neural Networks are built after biological neural systems. A NN stores patterns with distributed coding and is a trainable nonlinear dynamic system. A NN has a faster response and a higher performance than those of a sequential digital computer in emulating the capabilities of...
the human brain. Recently, NN have been used in pattern recognition problems, especially where input patterns are shifted in position and scale-changed.

Some works have been carried out on fuzzy neural network systems for pattern recognition. Kosko [28] has proposed a Fuzzy Associate Memory (FAM) which defined mappings between fuzzy sets. FAM used fuzzy matrices instead of fuzzy neurons to represent fuzzy associations. Yamakawa and Tomoda [29] have described a simple fuzzy neuron model and used in a neural network for application in character recognition problems. However, they did not describe the specific learning algorithm for this network. Takagi [30] have constructed a structured neural network using the structure of fuzzy inference rules. This structured NN has better performance than ordinary NN’s when used in pattern recognition problems. However, it is complicated to train this NN as it is composed of many small NN’s. Kwan [31] has presented a new kind of fuzzy neural network to combine the features of fuzzy systems (with an ability to process fuzzy information using fuzzy algorithms) and the features of neural networks (with a learning ability and a high-speed parallel structure) to form a fuzzy neural network which can learn from environments. This FNN defines a fuzzy neuron (FN), introduce four types of fuzzy neurons (FN’s) and use them to construct a four-layer feedforward fuzzy neural network which can be applied to pattern recognition. This FNN also propose a self-organizing learning algorithm for the four-layer feedforward FNN. The structure of this FNN is simple and its learning and recall speeds are fast. The third and fourth layers of the FNN are self-organized during learning.
4.3.2 *Fuzzy Neurons*

A typical nonfuzzy neuron has \( N \) weighted inputs and one output. The neuron sums these input \( x_i \) (for \( i = 1 \) to \( N \)) through the corresponding weights \( w_i \) (\( i = 1 \) to \( N \)) and transfers the result to a nonlinear activation function \( f(\cdot) \). The output of such a neuron can be expressed as:

\[
y = f\left(\sum_{i=1}^{N} w_i x_i - T\right)
\]

Where \( T \) is the internal threshold of the neuron.

*Definition of Fuzzy Neuron*

A FN has \( N \) weighted inputs, \( x_i \) for \( i = 1 \) to \( N \), with \( w_i \) (\( i = 1 \) to \( N \)) as the weights, and \( M \) outputs, \( y_j \) for \( j = 1 \) to \( M \). All the inputs and weights are real values and the outputs are real values in interval \([0,1]\). Each output could be associated with the membership value of a fuzzy concept, i.e. it expresses to what degree the pattern with the inputs \( \{x_1, x_2, ..., x_N\} \) belongs to a fuzzy set. Moreover, we have:

\[
z = h[\!w_1 x_1, w_2 x_2, ..., w_N x_N]\]

\[
s = f(z - T)\]

\[
y_j = g_j[s] \text{ for } j = 1 \text{ to } M
\]

Where \( z \) is the net input of the FN; \( h[] \) is the aggregation function; \( s \) is the state of the FN; \( f[] \) is the activation function; \( T \) is the activation threshold; and \( \{g_j[], j=1,2,...,M\} \) are the \( M \) output functions of the FN which represent the membership functions of the input pattern \( \{x_1, x_2, ..., x_N\} \) in all the \( M \) fuzzy sets. Consequently, FN’s can express and process fuzzy information.
B. Input-FN

If a FN is used in the input layer of a FNN and it has only one input $x$ such that

$$z = x$$

Then this FN is called an INPUT-FN.

C. Maximum-FN (Min-FN)

If a maximum function is used as the aggregation function of a FN such that

$$z = \max_{i=1}^{N}(w_i x_i)$$

Then this FN is called a MAXIMUM-FN or MAX-FN

D. Minimum-FN (Min-FN)

If a minimum function is used as the aggregation function of a FN such that

$$z = \min_{i=1}^{N}(w_i x_i)$$

Then this FN is called a MINIMUM-FN or MIN-FN

E. Competition-FN (Comp-FN)

If a FN has a variable threshold $T$ and only one output such that

$$y = g(s - T) = \begin{cases} 0 & \text{if } s < T \\ 1 & \text{if } s \geq T \end{cases}$$

$$T = t[c_1, c_2, ..., c_k]$$
Where $s$ is the state of the FN; $t[]$ is the threshold function; and $c_k$ ($k=1$ to $K$) are competitive variables of the FN. This FN is called a COMPETITIVE-FN or COMP-FN.

4.3.3 Structure of the FNN

The proposed FNN is a four-layer feedforward FNN as shown in Figure 4.2. The first layer is the input layer which accepts patterns into the network. We use INPUT-FN’s in this layer. Each INPUT-FN in this layer corresponds to one pixel of an input pattern. The INPUT-FN’s are displayed and indexed in two-dimensional and the number of FN’s in this layer is equal to the total number of pixels of an input pattern. Assuming each input pattern has $N_1 \times N_2$ INPUT-FN’s. The algorithm of the $(i,j)$th INPUT-FN in the first layer is:

$$s_y^{[1]} = z_y^{[1]} = x_{ij}, \text{ for } i = 1 \text{ to } N_1, j = 1 \text{ to } N_2$$

$$y_y^{[1]} = s_y^{[1]} / P_{\text{max}}, \text{ for } i = 1 \text{ to } N_1, j = 1 \text{ to } N_2$$

Where $x_{ij}$ is the $(i,j)$th pixel value of an input pattern ($x_{ij} > 0$) and $P_{\text{max}}$ is the maximum pixel value among all input patterns.

The second layer is also displayed in two-dimensional and consisted of $N_1 \times N_2$ MAX-FN’s. The purpose of this layer is to fuzzify input patterns through a weight function $w[m,n]$. The state of the $(p, q)$th MAX-FN in this layer is:

$$s_{pq}^{[2]} = \max_{j=1}^{N_2} (\max_{i=1}^{N_1} (w[p-i, q-j]y_{ij}^{[1]}))) \text{ For } p=1 \text{ to } N_1, q=1 \text{ to } N_2$$
Where \( w[p-i, q-j] \) is the weight connecting the \((i,j)\)th INPUT-FN in the first layer to the \((p, q)\)th MAX-FN in the second layer which is defined by:

\[
w[m, n] = \exp(-\beta^2 (m^2 + n^2))
\]

for \( m = (N_1-1) \) to \((N_1-1)\), \( n = (N_2-1) \) to \((N_2-1)\)

A plot of \( w[m, n] \) for \( \beta=0.3 \) is shown in Figure 4.3. By using this weight function, each FN in the second layer is just like a lens so that each FN focuses on one pixel of an input pattern but it also can see the surrounding pixels.
Each MAX-FN in this layer has $M$ different outputs ($M$ is the number of FN's in the third layer), one for each FN in the third layer. The outputs of the $(p, q)$th MAX-FN in the layer are:

$$y^{[2]}_{pqm} = g_{pqm}[s^{[2]}_{pq}]$$

For $p=1$ to $N_1$, $q=1$ to $N_2$, $m=1$ to $M$

Where $y^{[2]}_{pqm}$ is the $m$th output of the $(p, q)$th MAX-FN which is to be connected to the $m$th MIN-FN in the third layer. The output function $g_{pqm}[s^{[2]}_{pq}]$ is determined by the learning algorithm and is defined as follows:

$$y^{[2]}_{pqm} = g_{pqm}[s^{[2]}_{pq}] = \begin{cases} 1 - 2\left|s^{[2]}_{pq} - \Theta_{pqm}\right|/\alpha & \text{if } \alpha / 2 \geq \left|s^{[2]}_{pq} - \Theta_{pqm}\right| \\ 0 & \text{if otherwise} \end{cases}$$

for $\alpha \geq 0$, $p=1$ to $N_1$, $q=1$ to $N_2$, $m=1$ to $M$

Where $\Theta_{pqm}$ is the central point of the base of the function $g_{pqm}[s^{[2]}_{pq}]$. What have to be determined by the learning algorithm are the corresponding $\alpha$ and $\Theta_{pqm}$ for every set of $p$, $q$ and $m$.

We use MIN-FN's in the third layer. Each MIN-FN in the third layer represents one learned pattern. Hence, the number of MIN-FN's in the third layer, $M$, could be determined only after the learning procedure is finished. The output of the $m$th MIN-FN in the third layer is:

$$y^{[3]}_m = s^{[3]}_m = \min_{p=1}^{N_1} \left( \min_{q=1}^{N_2} \left( y^{[2]}_{pqm} \right) \right) \text{ for } m = 1 \text{ to } M$$

Where $s^{[3]}_m$ represents the state of the $m$th MIN-FN in the third layer.
The fourth layer is the output layer. We use COMP-FN's in this layer, one for each of the \( M \) learned patterns, to provide nonfuzzy outputs. If an input pattern is most similar to the \( m \)th learned pattern, then the output of the \( m \)th COMP-FN in the fourth layer is 1 while other outputs are 0. The number of COMP-FN's in the output layer is equal to \( M \). The algorithm of the \( m \)th COMP-FN in the fourth layer is:

\[
s_m^{[4]} = z_m^{[4]} = y_m^{[3]} \quad \text{for } m = 1 \text{ to } M
\]

\[
y_m^{[4]} = g[s_m^{[4]} - T] = \begin{cases} 0 & \text{if } s_m^{[4]} < T \\ 1 & \text{if } s_m^{[4]} = T \end{cases} \quad \text{for } m = 1 \text{ to } M
\]

\[
T = \max_{m=1}^{M}(y_m^{[3]}) \quad \text{for } m = 1 \text{ to } M
\]

Where \( T \) is the activation threshold of all the COMP-FN's in the fourth layer.
4.3.4 Self-Organizing Learning Algorithm of the FNN

The following parameters must be determined by the learning procedure: the parameters of the output functions of the second layer, $\alpha$ and $\Theta_{pqM}$, the parameter of the fuzzification function,$\beta$; and the number of neurons in each of the third and fourth layers, $M$. We define $T_f$ as the fault tolerance of the FNN and $K$ as the total number of training patterns.

♦ Step 1:

Create $N1 \times N2$ first layer and second layer, choose a value for $\alpha$ and $\beta$.

♦ Step 2:

Set $M=0$ and $k=1$;

♦ Step 3:

Set $M=M+1$. Create the $M$th neuron in the third and fourth layer. Set:

$$\Theta_{pqM} = \delta^{[2]}_{pqM} = \max_{i=1}^{N_1} \max_{j=1}^{N_2} (w[p-i, q-j]x_{jk})$$

♦ Step 4:

Set $k=k+1$. If $k>K$, then the learning procedure is finished. Otherwise, input the $k$th training pattern to the network and compute the output of the current FNN. Set:

$$\delta = 1 - \max_{j=1}^{M} (y_{jk}^{[3]})$$

Where $y$ is the output of the third layer for the $k$th training pattern $X_k$. If $\delta<T_f$, go to step 4. If $\delta>T_f$, go to step 3.
4.3.5 Analysis of the FNN

Like Neural Networks, this FNN is a parallel system which possesses all the pixels of an input pattern simultaneously. The proposed FNN is adaptively organized or constructed during the learning procedure. The proposed FNN is consisted of four types of fuzzy neurons, which can express and process fuzzy information and uses fuzzy algorithm to solve pattern recognition problems.

The first layer of the network accepts the data of an input pattern into the network. The INPUT-FN’s in this layer transform the pixel values of an input pattern into normalized values within interval [0,1].

The second layer of the network fuzzifies the input pattern. Each MAX-FN in the second layer is connected to all the INPUT-FN’s of the first layer by the weight function $w[m,n]$ and takes the maximum value of all the weighted inputs as its state. The result of using MAX-FN’s and using fuzzification weights is that one dark pixel in the input pattern will affect the states of several FN’s in the second layer. Consequently, the dark pixels of the input pattern are fuzzified by the second layer of the FNN.

The FN’s in the third layer give the similarities of the input pattern to all of the learned patterns. As we use MIN-FN’s in the third layer, the similarity of the input pattern $X=\{x_j\}$ to the $m$th learned pattern is computed by the FNN as the output of the $m$th MIN-FN in the third layer. When an input pattern $X$ is one of the learned patterns, there will be
one similarity $y_m^{[3]}$ which equals to 1. When the input pattern is not any of the learned patterns, all of the $M$ similarities are less than 1.

The output layer of the FNN is used to do defuzzification and give nonfuzzy outputs. It chooses the maximum similarity as the activation threshold of all the COMP-FN’s in the fourth layer. If $y_m^{[3]}$ is the maximum among all the outputs of the FN’s in the third layer, then the output of the $m$th COMP-FN in the fourth layer is 1 and the outputs of the other COMP-FN’s in the fourth layer are 0. The recognition procedure of the proposed FNN is finished in four steps: input data(layer 1); fuzzification(layer 2); fuzzy deduction (layer 3); and defuzzification (layer 4).

Using the self-organizing learning algorithm proposed in this paper, the third and fourth layers of the FNN are constructed during the learning procedure. Additional training patterns could be learned at any time by restarting the learning algorithm from the step 2 with $M = M_0$ ($M_0$ is the number of previous learned patterns). As a result, new additional FN’s in the third and fourth layers will be added when distinct patterns are used in the additional training patterns. If a learned pattern or a pattern similar to one of the learned patterns is fed to the FNN, the FNN will treat this pattern as a previous learned pattern without relearning it. Whether an additional pattern will be treated as a distinct pattern or as a learned pattern is determined by the similarities of the additional pattern with all the learned patterns, and by the learning parameters $\alpha, \beta$ and $T_r$. 

*Phoneme Based Speech Recognition Using Self-Organizing Map* - 50 -
The proposed FNN is used to process 2-D patterns. The FN's in its first two layers are displayed and indexed in two-dimension. Since the 2-D structure of the first two layers only affect the indexes of the connection weights between the first and the second layers, the FNN can be implemented in hardware as if it were a 1-D structure by using the corresponding 1-D indexes of the connection weights. Besides being a fuzzy system, this FNN also possesses the advantages of high-speed parallelism, and the learning ability of neural networks.

This FNN has been used successfully by Kwan for pattern recognition of 26 English alphabets and the 10 Arabic numerals.
Chapter 5

Hybrid Speech Recognizer by SOM and FNN

5.1 Speech Recognition Application by SOM

A growing trend in speech recognition application is to combine HMMs with ANNs in a hybrid system. This allows the temporal modeling capability of HMMs and the discriminative nature of ANNs to be exploited.

Self-Organizing Map as one type of artificial neural network, has the special property that the internal representation of information are generally organized spatially, just like the topographically organized maps found in the brain. There are lots of researches for many application fields by SOM, in which speech recognition is one of them.

T. Kohonen[32] has used the SOM for the Finnish phonemes recognition, the simplest type of speech maps formed by self-organizing is the static phoneme map. There are 21 phonemes in Finnish. For their representation he used short-time spectra as the input pattern $x(t)$. The spectra were evaluated every 9.83ms. They were computed by the 256-point FFT, from which a 15-component spectral vector was formed by grouping of the channels. In the present study all the spectral samples, even those from the transitory regions, were employed and presented to the algorithm in the natural order of their utterance. During learning, the spectra were not segmented or labeled in any way: any
features present in the speech waveform contributed to the self-organized map. After adaptation, the map was calibrated using known stationary phonemes by LVQ (Figure 5.1). The map resembles the well known formant maps used in phonetics; the main different is that in the maps complete spectra, not just two of their resonant frequencies as in formant maps, are used to define the mapping.

Recognition of discrete phonemes is a decision-making process in which the final accuracy only depends on the rate of misclassification errors. It is therefore necessary to try to minimize them using a decision-controlled (supervised) learning scheme, using a training set of speech spectra with known classification.

Figure 5.1 an example of a phoneme map
In practice, for a new speaker, it will be sufficient to dictate 200 to 300 words which are then analyzed by an automatic segmentation method. The latter picks up the training algorithm spectra that is applied in the supervised learning algorithm. The finite set of training spectra must be repeated in the algorithm either cyclically or in a random permutation. LVQ1, LVQ2, or LVQ3 can be used as the fine tuning algorithm. A map created for a typical speaker can then be modified for a new speaker very quickly, using 100 more dictated words, and LVQ fine tuning only.

Because of coarticulation effects, i.e., transformation of the speech due to neighboring phonemes, systematic errors appear in phonemic transcriptions. For instance, the finnish word “hauki” (meaning pike) is almost invariably recognized as the phoneme string /haouki/ by the acoustic processor. It may then be suggested that if a transformation rule /aou/ ---/au/ is introduced, this error will be corrected. It might also be imagined that it is possible to list and take into account all such variations. However, there may be hundreds of different frames or contexts of neighboring phonemes in which a particular phoneme may occur, and in many cases such empirical rules are contradictory; there are only statistically correct. The frames may also be erroneous. In order to find an optimal and very large system of rules, the Dynamically Expanding Context grammar was applied [33]. Its rules or productions can be used to transform erroneous symbol strings into correct ones, and even into orthographic text.

Because the correction rules are made accessible in memory using a software content-addressing method, they can be applied very quickly, such that the overall operation of
the grammar, even with 15000 rules, is almost in real time. This algorithm is able to correct up to 70% of the errors left by the phoneme map recognizer.

The speech recognition system developed at Helsinki University of Technology employs the Self-Organizing Feature Maps (SOFMs) as vector quantizers. The SOFMs are two-dimensional artificial neural network, the cells of which become tuned to various input signal patterns through an unsupervised learning process. In order to distinguish phonemic classes in the best possible way, the SOFMs have been fine-tuned by Learning Vector Quantization (LVQ). The aim of vector quantization here is not to represent speech with the smallest possible distortion, but with the best possible separation between phoneme classes. The LVQ accomplishes this by modifying the codevectors adaptively so that the borders between phoneme classes approximate Bayes’ decision surfaces. As a result, the speech signal will be vector-quantization into a string of phoneme labels a centisecond apart, which is called the quasiphoneme string.

The feature vectors used in the system are 15-component approximations of the short-time power spectra of the speech signal. These are based on 256-point FFT computed every centisecond. A so-called stationary SOFM is first used to vector-quantize the feature vectors derived from the entire speech signal. This SOFM cannot make distinctions between unvoiced plosives, but it can detect their presence. Extra transient SOFMs are then used to classify the plosive bursts more accurately. A single classification decision is made from the basis of a feature vector constructed from the
burst of a plosive. These transient SOFMs have been trained using only data picked from corresponding phonemes, in this case from the unvoiced plosives.

The quasiphoneme strings obtained by using the stationary SOFM have been transformed into phonemic transcription using simple context-free transformation rules that take into account only the durations of individual phonemes, not the effect of their contexts on them. The language used in this system is Finnish, compared to the English language, the number of phonemes in Finnish can be seen to be quite small. Furthermore, a large proportion of the phonemes are quasistationary, i.e., they often have a stationary part and thus they are relatively easy to recognize. The choice of phonemes as the basic recognition unit is therefore natural. The methods described are not limited to Finnish only, however, their applicability to Japanese has already been demonstrated.

In [34], Mikko Kokkonen has proposed a new approach (Figure 5.2) to construct transcription of spoken utterances. The Self-Organizing Feature Maps by Kohonen are first applied to vector-quantize speech into a sequence of phoneme labels a centisecond apart. This code sequence is converted into a phoneme string using a multi-layered feed-forward network trained with error back propagation. The trained network acts as a filter removing undesired transitional and coarticulatory effects from the code sequence. This makes it almost a trivial task to convert the code sequence into a phoneme sequence. The need to any statistical speech models, such as Hidden Markov Models, is thus eliminated. The new approach is compared to an existing one being used in a speech recognition system, in which simple durational rules are used for the same transformation task.
Figure 5.2 a simplified configuration of the system
The accuracy of produced phonemic transcriptions is 4.8 percent units better using the proposed multi-layered network approach (88.4% opposed to 83.6%).

Mikko Kokkonen did not get good results for English phoneme transcription because the number of English phonemes is larger than that of Finnish phonemes and a large proportion of the phonemes are not quasistationary, and not easy to recognize them. In this thesis, we will deal with the digit speech recognition task, so only 20 phonemes (including silence) which are the fundamental components to form the digit speeches are needed to train and form the phoneme maps.

Most of speech recognition applications by SOM are focused on the phoneme transcription and recognition. There are few application examples by SOM for word and sentence recognition because there are lack of effective ways to deal with 2D spatial response information of words and sentences on SOM.

In this thesis, a novel fuzzy neural network (FNN) [31] is proposed to apply to the phoneme and words recognition based on the phoneme maps. This FNN will use the 2D spatial information of phoneme maps which are the spatial responses of speech signal and broaden the research perspective of speech recognition by SOM.
5.2 Hybrid Digit Speech Recognizer

5.2.1 Structure

This hybrid speech recognizer is composed of two layer structures. The first layer is Self-Organizing Phoneme Map (SOPM) (Figure 5.3), it's an unsupervised neural network, and the neighboring cells in output layer compete in their activities by means of mutual lateral interactions, and develop adaptively into specific detectors of phoneme signal patterns. The second layer is Fuzzy Neural Network (FNN) (Figure 5.4); it can be constructed into supervised neural network, and be used for the two-dimensional pattern matching problems. FNN can memorize the new training patterns of signal by fuzzifying the input patterns, which can improve the matching result of similar signal patterns; then the well-trained FNN can recognize the testing patterns of signal by comparing them to the learning patterns stored in the FNN.

In this hybrid recognizer, SOPM is first trained by MFCC feature vectors of speech frames cutting from digit voice samples, then the known phoneme samples are used to LVQ the SOPM for the optimization of labeling of neurons in the output map. The tuned SOPM so can be used to obtain the two-dimensional response patterns (Figure 5.3) of digit voice samples, which are further input to FNN for training and recognition of digit voice samples.
Figure 5.3 Self-Organizing Phoneme Map and Training
Figure 5.4 Structure of Fuzzy Neural Network
So, the digit speech recognition problem can be transferred to the 2-dimensional pattern matching problem, after SOM well learns the feature vectors of phonemes from different speakers, it can be used to create the 2-dimensional patterns from the digit voice utterances. We can select some digit utterances from different speakers and their corresponding response pattern on SOPM to form the training corpus for FNN (Figure 5.3), and FNN will memorize these training patterns and be used to recognize the response patterns coming from testing digit utterances from similar speakers as training corpus or totally different speaker by comparing the testing response patterns with memorized training patterns. Figure 5.4 is the structure of Fuzzy Neural Network and its pattern matching.
5.2.2 Strategy of Speech Phoneme Feature Extraction

The speech recognition can be divided into two processes: feature extraction and pattern recognition. Feature extraction is responsible for searching the speech characteristics and storing them for the second process: pattern recognition. In order to identify the speech characteristics accurately and efficiently, it is necessary to investigate the features and classifications of speech sounds.

Any natural language, including English, is based on a set of distinguishable and mutually exclusive primary units, which are called phonemes. All the phonemes are related to different articulator gestures of a language. A speech signal can be broken into several small components: phonemes, diphones, syllables or words, where a phoneme is a minimal unit of speech sound. Most recognition systems are based on the phonemes.

Feature extraction over the phonemes is a challenging work because some phonemes are short duration while others are long duration and having dynamic nature, and same phonemes uttered by different persons may seems very different, moreover, phonemes sometimes have a great influence on neighboring phonemes, this is so-called coarticulatory effects.

Most speech recognition systems extract mel-frequency cepstral coefficients (MFCC coefficients) from phoneme or speech signal to represent them, please refer to chapter 3 for details of MFCC feature extraction. Some systems [35] extract three continuous frames MFCC feature vectors to represent the phonemes and speech signals, but
some phonemes are long duration and we have to take into consideration the dynamic nature of these phoneme speech: the identify of a phoneme will often depend not only on the spectral features at one point in time, but it will also depend on how the features change over time.

Figure 5.5(a) waveform of the phoneme /t/

Figure 5.5(b) 14 MFCC curve of five frames extracted from phoneme /t/
In Figure 5.5(a), we can see the phoneme /t/ has short duration and only five frames can be extracted (Figure 5.5(b)), and the MFCC feature change over time is relatively small; while in Figure 5.6(a), phoneme /ou/ has long duration and up to 16 frames can be extracted (Figure 5.6(b)), the MFCC feature change over time is prominent.

In order to capture the internal dynamic feature of phonemes with long duration and prominent feature change over time, we need to take a wide context window of features; but this wide context window can not be appropriate for the phonemes with short duration. In [36], the speech recognition system with neural network at the OGIS use broad contexts to define 576 categories for all American English phonemes as a compensate for the conflicts of long and short phoneme for feature extraction, but this large number of phonemes will increase the computation and efficiency burden for speech recognition system. In this thesis, we will divide the phonemes of the digit utterances into short and long phonemes according to phoneme duration as well as practical experience. For short phonemes, we can take 3 continuous frames MFCC features to represent them, and for long phonemes, we will take 5 separated frames MFCC features to obtain the feature vectors of these phonemes (Figure 5.7).
Figure 5.6(a) Waveform of the phoneme /ou/

Figure 5.6(b) 14 MFCC curve of 16 frames extracted from phoneme /ou/
Figure 5.7 3 continuous and 5 separated frames feature extraction of short and long phonemes
There are totally 20 phonemes (including silence) that form the fundamental components of these 10 digit-voices. The speech data for the examples and experiments in this thesis is available from the Otago Speech Corpus on New Zealand [38]. These fundamental components of these 10 digit-voices are showed as following table (Table 5.1). There are two kinds of phoneme components for ‘two’ because of the variety of utterances in New Zealand English. According to the phoneme utterance duration and dynamic feature, 13 phonemes are classified into short phoneme while other 7 phonemes are classified into long phonemes (Table 5.2).

<table>
<thead>
<tr>
<th>Phoneme Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
</tr>
<tr>
<td>One</td>
</tr>
<tr>
<td>Two</td>
</tr>
<tr>
<td>Three</td>
</tr>
<tr>
<td>Four</td>
</tr>
<tr>
<td>Five</td>
</tr>
<tr>
<td>Six</td>
</tr>
<tr>
<td>Seven</td>
</tr>
<tr>
<td>Eight</td>
</tr>
<tr>
<td>Nine</td>
</tr>
</tbody>
</table>

Table 5.1 phoneme component for all the 10 digit voices

<table>
<thead>
<tr>
<th>Phoneme components of all 10 digit voices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short phonemes</td>
</tr>
<tr>
<td>/l/, /k/, /ʃ/, /v/, /θ/, /s/, /z/, /n/, /r/, /w/, /iː/, /e/, /l/</td>
</tr>
<tr>
<td>Long phonemes</td>
</tr>
<tr>
<td>/iː/, /oː:/, /uː:/, /eɪ/, /æː/, /ɒː/, /silence/</td>
</tr>
</tbody>
</table>

Table 5.2 13 short phonemes and 7 long phonemes
5.2.3 Learning Process of Phoneme Feature Maps

The feature vectors of short and long phonemes will be inputted as learning of phoneme feature maps, however, the size of short and long phoneme feature vectors is different while the size of input layer of SOM are fixed, so we need two SOM to deal with short and long phonemes separately and combine these two short and long phoneme maps into one phoneme map for the learning and recognition of FNN for speech pattern matching.

We can use the digit speech utterances and extract 3 continuous frames MFCC along the speech signal to train the short phoneme SOM, after enough iteration the weight well memorizes the input patterns and then the known short phonemes and their corresponding 3 continuous frames MFCC feature vectors are used to do learning vector quantization (LVQ) to make optimal boundary on the maps for phoneme labeling. The similar steps for long phoneme map learning process, instead 5 separated frames MFCC feature vectors are used to train and LVQ SOM to form long phoneme self-organizing map (Figure 5.8).
In Figure 5.9, we can see each neuron in the short and long phoneme maps can be labeled by one particular phoneme and neurons responsive to the same phoneme form domains, which are grouped to form the spatial order of internal representations of the phonemes. The aim of vector quantization (VQ) here is not to represent speech with the smallest possible distortion, but with the best possible separation between phoneme classes.

Figure 5.9 Fine-tuned self-organizing short and long phoneme maps (only main phoneme domains show)
5.2.4 *Optimal Response Patterns of Speech Signal on SOM by Viterbi Algorithm*

The short and long fine-tuned phoneme maps are combined to form one tuned phoneme map that is later used as 2D pattern recognition by FNN. For a speech voice input, the short phoneme vector and long phoneme vector are extracted at the same time for central frame along input voice; these two vectors go into short and long phoneme maps separately to have the strongest response point for each phoneme domain on the maps (Figure 5.10). The response value on the short and long maps will divided by the frame number of input layer of SOM to compare the maximum response on the maps.

![Diagram showing phoneme maps and speech signal response](image)

Figure 5.10 speech signal response on phoneme maps
\[ \| W_n - P_i \| = \frac{1}{N} \min_{n_i \in M} \left( \| W_{n_i} - P_i \| \right) \quad (N = 3, 5) \]

Where \( N \) is the frame number of the context window, for short phoneme map, \( N = 3 \), for long phoneme map, \( N = 5 \).

How to form the 2D phoneme response sequence of speech signal on the maps is the key factor for the word recognition by FNN. There are lots of variations for each digit voice sample, even for the same digit samples by same persons; the phoneme response sequences on the maps still have somewhat differences which make them difficult for FNN to recognize them as same digit patterns.

The initial method I choose to get 2D response curve is at first to find maximum response point (compare points on short and long phoneme maps) for each central frame along voice input, and then filter some noise points by choosing the large response points, at last connect them by response sequence to form 2D phoneme response curve. But the experiment result is not so good, only recognition rate at 76% for 240 samples. The reason I think is there is much diversity for speaker voice even the same speaker, the 2D response curves on the maps are still much different from each other for the same words from same person, the following two response curves (Figure 5.11) are ‘zero’ response from same person, but they could not be classified into same pattern by FNN because some noises exist for phoneme sequence although these two phoneme sequences look like very similar. Figure 5.12 are two samples of ‘six’ by sample speaker, we can see the difference of phoneme response sequences make it happened to recognize them as different digits. So maximum response based method cannot achieve good result.
(a) The phoneme response sequence of ‘zero’ on the phoneme map
(ттттфлссссссссссссссссссссссссссссUUUUdудудудудудудудудудудудудудudу)

(b) The phoneme response sequence of ‘zero’ on the phoneme map
(0000ттфлсссссссссссссссссссссссссссссUUUUdудудудудудудудудудудудудудуду)

Figure 5.11 two maximum response patterns of ‘zero’ by same person
(a) The phoneme response sequence of ‘six’ on the phoneme map
(fft000ssssfiiiiiiiidi ou oiuutttkkkkkkttsssssst0000fftt)

(b) The phoneme response sequence of ‘six’ on the phoneme map
(fff0ftssssttttffiiiiitiittkkkkkkfssssssssssstkkffft)

Figure 5.12 two maximum response patterns of ‘six’ by same person
From Figure 5.11, Figure 5.12, the phoneme response sequences on the phoneme maps are not necessary to be similar even for the same digit voices from same speaker because of the variation and complexity of speech signal, but if we look into the phoneme sequences of digit voice, we can find the sequence of what we expect for that digit voice hidden in the response phoneme sequences. Figure 5.13, Figure 5.14 show us the phoneme sequences hidden in the response phoneme sequences of digit voices on the phoneme maps.

\[
\begin{array}{cccc}
0000tfsssszzzzeeeeeeertrrrrrreeeAA\Lambda\Lambda & A & A & A \\
\end{array}
\]

Figure 5.13 the phoneme sequence of ‘zero’ hidden in the response sequence.

\[
\begin{array}{cccc}
fff000ssssfiiiiiiiiiiiu uu uuuuttttkkkkkkttsssssssst00000fft \\
\end{array}
\]

Figure 5.14 the phoneme sequence of ‘six’ hidden in the response sequence.

These hidden phonemes have strong phoneme response on the maps, if we can extract these hidden phoneme sequence from response sequence, the noise phonemes will be filtered and the recognition result by FNN will improved dramatically.
An alternate and effective way is to use Viterbi search algorithm to find the hidden phoneme sequences and their corresponding optimum 2D phoneme response curve on the map, once we obtain the fine-tuned short and long phoneme maps, each phoneme can form domain, and for the central voice frames along time, we extract 3 continuous and 5 separated frames MFCC vectors and then find the maximum response point in each phoneme domain on short and long phoneme maps, its response value, 2D position, and form trellis (Figure 5.15) along input frames (skip silence).

How can we find the most possible phoneme sequence uttered by the speaker along trellis, and thus the corresponding 2D phoneme curve in the map? The Viterbi search algorithm can deal well with it! Suppose we already know the maximum response value for each phoneme domain along frame sequence, this is so called posterior possibility, now we have to define transition value between each phoneme (state). To make things simple, we just define the transition value from each phoneme to itself and transition value between the phonemes that appear next to each other in the under-recognition word speeches are 1; other transition values are 0. (Figure 5.16) For example, in the digit voice '0ri:', we define the transition from each phoneme to itself is 1, so from /θ/ to /θ/ is 1, from /r/ to /r/ is 1, and from /i:/ to /i:/ is 1; also from /θ/ to /r/ is 1, and from /r/ to /i:/ is 1.

The definition of these transition value just sets up the constrains between phonemes in digit voices, some phoneme combinations will never happen in our digit voice recognition task such as from /z/ to /k/, from /t/ to /θ/, or from /u:/ to /i:/.
transition value will be zero, while the transition of some other phoneme combinations which will happen in our digit voices will be one.

![19 Phonemes Trellis](image)

**Figure 5.15** 19 phoneme trellis along voice frames sequence

| t  | k  | f  | v  | θ  | s  | z  | n  | r  | w  | i  | e  | Ai: | o:u: | ei | ai | āu |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| t  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 0  |
| k  | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| f  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  |
| v  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  |
| θ  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  |
| s  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  |
| z  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  |
| n  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 1  |
| r  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 1  | 0  | 0  |
| w  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| i  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | 1  | 0  | 0  | 0  | 0  |
| e  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | 0  |
| Λ  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  |
| i: | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  |
| o: | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| u: | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  |
| ei | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  |
| ai | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 1  |
| āu | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

**Figure 5.16** Transition value defined between phonemes
Now we have the response values \( q(y_i \mid s_t, s_{t-1}) \) and the transition value \( p(s_i \mid s_{i-1}) \), we can form phoneme-time trellis along input frames \( S=1, 2, \ldots, 19; \) corresponding to 19 phoneme states. \( Y_i(i=0, \ldots, M) \) corresponding to \( i \) th frame speech input. For each frame speech input, we can get the maximum response values of each phoneme domain on the phoneme; so there are 19 response values for each frame speech input (see Figure 5.15, each column corresponds to each frame speech input and 19 phoneme response values), the transition value \( P(s_i \mid s_{i-1}) \) is just for the calculation of maximum probability path. The maximum probability path can be calculated recursively by following formulas:

\[
    r_i(s_i) = \max_{s_{i-1}} p(s_i, s_{2}, \ldots, s_{t}, y_1, y_2, \ldots, y_i \mid s_0) \\
    = \max_{s_i} p(y_i, s_i \mid s_{i-1}) r_{i-1}(s_{i-1}) \\
    = \max_{s_i} q(y_i \mid s_i, s_{i-1}) p(s_i \mid s_{i-1}) r_{i-1}(s_{i-1})
\]

The Viterbi search algorithm can find the most possible phoneme sequence directly along phoneme-time trellis, so the optimal phoneme sequences of word speeches are formed and thus the corresponding 2D phoneme curves by connecting the phoneme response sequence points in the maps (see Figure 5.17, Figure 5.18), which are used for 2D pattern recognition of fuzzy neural network.
(a) phoneme sequence

'zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz'

(b) phoneme sequence

'zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz'

Figure 5.17 two optimal response patterns of 'zero'
in Figure 5.11 after Viterbi algorithm
(a) phoneme sequence
'sssssssssssssssssiiiiiiiiiiiiiiiiiiiiiiikkkkkkkkkkkkkkkkssssssssssssssss'

(b) phoneme sequence
'sssssssssssssssssiiiiiiiiiiiiiiiiiiiiiiikkkkkkkkkkkkkkkkssssssssssssssss'

Figure 5.18 two optimal response patterns of 'six'
in Figure 5.12 after Viterbi algorithm
5.2.5 Training and Recognition Process by FNN

Once we obtain the optimal phoneme response patterns of digit voices on the phoneme maps by Viterbi search algorithm, these patterns are binary and suitable for the learning and recognition by Fuzzy Neural Network.

The first layer of FNN is to accept the input patterns for learning and recognition, the second layer is to fuzzify the input patterns and make the FNN have the ability to recognize some shifted and distorted patterns. The third and fourth layers of the FNN are constructed during the learning procedure. Additional training patterns could be learned at any time by restarting the learning algorithm. As a result, new additional FN's in the third and fourth layers will be added when distinct patterns are used in the additional training patterns. If a learned pattern or a pattern similar to one of the learned patterns is fed to the FNN, the FNN will treat this pattern as a previous learned pattern without relearning it.

Whether an additional pattern is determined by the similarities of the additional pattern with all the learned patterns, and by the learning parameters $\alpha$, $\beta$ and $T_f$. $\alpha$ and $\beta$ affect the computation of similarities. $T_f$ is the fault tolerance of the FNN. If one of the similarities of an input pattern is larger than or equal to $1 - T_f$, then this input pattern is treated as a previously learned pattern. Otherwise, it is treated as a new distinct pattern. We can see that the learning of this FNN can be a continual process. In other words, the FNN needs
not to be completely trained before being used. This is somewhat like the learning ability of the human brain.

This proposed FNN is used to process 2-D patterns. The FN's in its first two layers are displayed and indexed in two-dimension. Since the 2-D structure of the first two layers only affect the indexes of the connection weights between the first and second layers, the FNN can be implemented in hardware as if it were a 1-D structure by using the corresponding 1-D indexes of the connection weights. Besides being a fuzzy system, this FNN also possesses the advantages of high-speed parallelism, and the learning ability of neural networks.

5.3 Speech Database

The speech database used for the recognition experiment is Otago speech corpus created by Department of Information Science at University of Otago in New Zealand [38]. This database includes all English phoneme wave files and some training and test word utterances from lots of speakers. All the phoneme and word samples are in WAVE format, and sampled in Mono at 22.050 kHz, with a 16 bit resolution.

Because this thesis is only concern with the digit speech recognition research, we select those phoneme samples which are the fundamental elements of 0—9 digit samples for the SOM learning and LVQ; and select the various digit voice samples from different speakers for FNN training and speech recognition.
5.4 Simulation and Results

We use 50 digit utterances (0-9 samples by 5 different speakers including 3 males, 2 females) to extract 3 continuous frame MFCC vectors and 5 separated frame MFCC vectors to train the short and long phoneme maps separately. After 8000 iterations, short and long phoneme maps well learn the MFCC feature vectors of speech signal. As SOM training progresses, the size of the neighborhood around the winning unit decreases, initially, large number of output units will be update, and as the learning proceeds smaller and smaller numbers are updated, until at the end of the learning only the winning unit is adjusted. Similarly, the learning rate will decrease for more and more accurate learning of input patterns as learning progresses.

After training SOM, we adapt type one learning vector quantization (LVQ1) to fine-tune the short and long phoneme maps. To define the optimal placement of weights in an iterative learning process, initial values for them is already obtained from training SOM by speech signal; the next phase is to determine the labels of the neurons on the phoneme maps, by LVQ1, the boundary classification accuracy on the phoneme maps is well improved.

we use 130 short phoneme samples (13 short phonemes by 10 different speaker including 2 females) and their corresponding 42 MFCC vectors to LVQ the learned SOM, after also 8000 iterations, we can obtain fine-tuned and optimal boundary short phoneme map; the
similar for LVQ for long phoneme map, instead we use 61 long phoneme samples (6 long phonemes by 10 speakers and one silence sample).

Thus we take altogether 80 training samples that are the 10 digital number utterances by 8 persons (5 male and 3 female) to obtain the 2D optimal phoneme curves on map by Viterbi search algorithm, and use these 80 2D patterns to train FNN. After that we use another 240 utterance test samples from same 8 persons (each person have 3 samples for each of the ten digits) and their corresponding 2D optimal phoneme response sequence patterns to test the FNN. Table.5.3 is the recognition results of these 2D optimal phoneme response sequences by FNN with different parameter values. We also can easily obtain the VQ-code phoneme sequences and their corresponding 2D phoneme sequences of 80 training samples and 240 test samples on the phoneme maps without Viterbi search algorithm, Table.5.4 is the recognition results of these 2D VQ-code phoneme response sequences by FNN with same set of parameters. We can see the Viterbi search algorithm can improve the recognition rate of digit voice test samples significantly.

<table>
<thead>
<tr>
<th>α</th>
<th>β</th>
<th>Number of FNs in 3rd &amp; 4th Layers</th>
<th>Recognition Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.1</td>
<td>76</td>
<td>78</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
<td>80</td>
<td>74</td>
</tr>
<tr>
<td>2.0</td>
<td>0.1</td>
<td>65</td>
<td>85.4</td>
</tr>
<tr>
<td>2.0</td>
<td>0.3</td>
<td>80</td>
<td>91.3</td>
</tr>
<tr>
<td>2.0</td>
<td>0.6</td>
<td>80</td>
<td>90</td>
</tr>
<tr>
<td>3.5</td>
<td>0.1</td>
<td>38</td>
<td>82</td>
</tr>
<tr>
<td>3.5</td>
<td>0.3</td>
<td>68</td>
<td>82.9</td>
</tr>
</tbody>
</table>

Table.5.3. Dependent Recognition Rate of FNN with different α and β (Tmin = 0.4) by 240 2D response patterns by Viterbi algorithm
<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>Number of FNs in 3rd &amp; 4th Layers</th>
<th>Recognition Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.1</td>
<td>80</td>
<td>65.0</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>2.0</td>
<td>0.1</td>
<td>78</td>
<td>75</td>
</tr>
<tr>
<td>2.0</td>
<td>0.3</td>
<td>80</td>
<td>78.7</td>
</tr>
<tr>
<td>2.0</td>
<td>0.6</td>
<td>80</td>
<td>76.2</td>
</tr>
<tr>
<td>3.5</td>
<td>0.1</td>
<td>74</td>
<td>77.5</td>
</tr>
<tr>
<td>3.5</td>
<td>0.3</td>
<td>80</td>
<td>73.7</td>
</tr>
</tbody>
</table>

Table 5.4: Dependent Recognition Rate of FNN with different $\alpha$ and $\beta$ ($T_f = 0.4$) by 240 2Dresponse patterns without Viterbi algorithm

We can see when the parameters of FNN $\alpha = 2.0$, $\beta = 0.3$, $T_f = 0.4$, the recognition rate of FNN can achieve the highest, and the number of FNs in 3rd & 4th Layers is the total number of learning patterns, and thus the FNN have to memorize all the learning patterns for dependent and independent speech recognition of digit voices.

To look into the details of digit speech dependent recognition, we examine the recognition rate of each digit of ten digits (0—9) of all the 240 digit samples, each digit has 24 utterances by 8 different speakers, Figure 5.19 shows the recognition rate of each digit in this simulation and we choose the learning parameters of FNN $\alpha = 2.0$; $\beta = 0.3$ at its best recognition rate.
Figure 5.19 Dependent Recognition rate of each digit of 240 digit samples
($\alpha=2.0; \beta=0.3$ for FNN)

Table 5.3 and Table 5.4 is the recognition result of dependent digit utterances, that means the 80 digit training samples and 240 digit testing samples are from same 8 persons. We will try independent digit speech recognition by collecting another 240 digit testing samples from totally different 8 persons (including 5 male, 3 female). Table 5.5 shows this independent recognition result of different learning parameters of FNN. Figure 5.20 shows the recognition rate of each digit in this simulation and we choose the learning parameters of FNN $\alpha=2.0; \beta=0.3$ at its best recognition rate.
<table>
<thead>
<tr>
<th>α</th>
<th>β</th>
<th>Number of FNs in 3rd &amp; 4th Layers</th>
<th>Recognition Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.1</td>
<td>80</td>
<td>62.5</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
<td>72</td>
<td>56.2</td>
</tr>
<tr>
<td>2.0</td>
<td>0.1</td>
<td>76</td>
<td>77.9</td>
</tr>
<tr>
<td>2.0</td>
<td>0.3</td>
<td>80</td>
<td>78.7</td>
</tr>
<tr>
<td>2.0</td>
<td>0.6</td>
<td>80</td>
<td>77</td>
</tr>
<tr>
<td>3.5</td>
<td>0.1</td>
<td>65</td>
<td>76.2</td>
</tr>
<tr>
<td>3.5</td>
<td>0.3</td>
<td>80</td>
<td>78.3</td>
</tr>
</tbody>
</table>

Table 5.5. Independent Recognition Rate of FNN with different α and β (T_r = 0.4) by 240 2D response patterns by Viterbi algorithm

Figure 5.20 Independent Recognition rate of each digit of 240 digit samples (α=2.0; β=0.3 for FNN)
5.5 Analysis of Experiment Results

In this section, we will just analyze the result and reason about the misclassifications of the dependent and independent recognition, and sum up several types of reason which cause the low independent recognition rate and provide some suggestions and directions for future research.

There are totally 21 misclassifications for the digit utterances in all the 240 dependent digit samples and 55 misclassifications for all 240 independent digit samples. There are several reasons for these misclassifications, the first type of misclassification is Wrong Phoneme Sequence (WPS), and the optimal phoneme sequence obtained by Viterbi search algorithm is not what we expect for that particular digit utterance. The reason why we can’t get the optimal phoneme sequence we expect is the variation of speech signal, some independent digit utterances could not obtain strong response on the phoneme maps. The following Figure 5.21 is the examples of this type of misclassification.
(a) Testing pattern of 'two'

(b) Training pattern of 'zero'
‘zzzzzzzzzzzzeeeeeeeeeeertrrrrrrrrrrrrrououououououou’

Figure 5.21 the wrong phoneme sequence (WPS) for testing pattern 'two' (digit 'two' (a) is misclassified into digit 'zero' (b), the wrong phoneme sequence of (a) is ‘ttttttttteeeeeeeertrrrrrrouou ouououououou’

Phoneme Based Speech Recognition Using Self-Organizing Map - 89 -
Another type of misclassification is the Pattern Mixing Up (PMU) caused by FNN. In following Figure 5.22, we can see the response pattern (a) of digit ‘three’ is misclassified into response pattern (b) of digit ‘zero’, although it has the similar pattern as the response pattern (c) of another digit ‘three’ in training samples. Figure 5.23 is another example for this misclassification.

(a) Testing pattern of ‘three’
Figure 5.22 Pattern Mixing Up (PMU) for testing pattern ‘three’
digit ‘three’ (a) is misclassified into digit ‘zero’ (b) although it has similar pattern as (c) of digit ‘three’,
phoneme sequence of (a) is ‘θθθθθθθθmmrrrrri:i:i:i:i:i:i:i:i:i:i’ Pattern (c) is the most similar training
pattern of all digits ‘three’ training patterns as pattern (a))
(a) Testing pattern of 'six'

(b) Training pattern of 'seven'
'sssssssssssseeeeeeheevevvvvviiiiiiiiiiiiiiiiinnnnnnn'
(c) Training pattern of 'six'
'ssssssssssssssssssssssssssssssssssssssssssss'

Figure 5.23 Pattern Mixing Up (PMU) for testing pattern 'six'
digit 'six' (a) is classified into digit 'seven' (b)
although it has similar pattern as (c) of digit 'six',
phoneme sequence of (a) is ' sssssssssssssssssssssssssssssssssssssssssssssssssssssss'. Pattern (c) is the most similar
training pattern of all digits 'six' training patterns
as pattern (a)

We do the same research for all the misclassifications of dependent and independent
speech recognition, and classify all the misclassifications into above two types of errors,
Table 5.6 and Table 5.7 show us some information about this classification.

<table>
<thead>
<tr>
<th></th>
<th>zero</th>
<th>one</th>
<th>two</th>
<th>three</th>
<th>four</th>
<th>five</th>
<th>six</th>
<th>seven</th>
<th>eight</th>
<th>nine</th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>WPS</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>PMU</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>SUM</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 5.6 the two types of misclassifications over 10 digits for all the 21 misclassifications
in dependent digit speech recognition by FNN
<table>
<thead>
<tr>
<th></th>
<th>zero</th>
<th>one</th>
<th>two</th>
<th>three</th>
<th>four</th>
<th>five</th>
<th>six</th>
<th>seven</th>
<th>eight</th>
<th>nine</th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>WPS</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>10</td>
<td>33</td>
</tr>
<tr>
<td>PMU</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>SUM</td>
<td>2</td>
<td>3</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>10</td>
<td>55</td>
</tr>
</tbody>
</table>

Table 5.7 the two types of misclassification over 10 digits for all the 55 misclassifications in independent speech recognition by FNN

5.6 Comparison with Hidden Markov Model based on SOM

The most flexible and successful approach to speech recognition so far has been Hidden Markov Models (HMMs). Figure 5.24 is the Markov general model of word ‘zero’, this HMM consists of the following elements:

\( \{s\} = \text{A set of states} /z/, /e/, /t/, /ow/ \).

\( a_{ij} = \text{A set of transition probabilities, where } a_{ij} \text{ is the probability of taking the transition from state } s_i \text{ to state } s_j. \)

\( \{b_i(O)\} = \text{A set of emission probabilities, where } b_i \text{ is the probability distribution over the acoustic space describing the likelihood of emitting each possible sound } O \text{ while in state } i. \)
In Figure 5.24, we can see the state $s_1$ and $s_6$ are corresponding to the silence of speech signal, and $O_i$ is corresponding to the frame of the speech signal. $a_{ij}$ and $b_i(O_i)$ are the transition probability and output probability of the model ‘zero’, these probabilities are the parameters of the model and normally are estimated by the learning from a lot of ‘zero’ samples.

A growing research trend in speech recognition is to combine HMM with ANN in a hybrid system. This allows the temporal modeling capability of HMM and the discriminative nature of ANN to be exploited [36] [37]. Alternatively, the standard HMM parameters can be estimated by discriminative training of ANN.
In this thesis, we can use SOM to estimate the output probabilities. On the fine-tuned short and long phoneme maps, there are totally 20 phoneme categories and each neuron is labeled by one of the 20 phonemes and each phoneme approximately occupies each domain on the maps. So, for the central voice frames along speech signal, we extract 3 continuous and 5 separated frames MFCC vectors and then find the maximum response value in each phoneme domain on short and long phoneme maps, these 20 maximum response values are the estimates of output probabilities of 20 phoneme categories for this central voice frame. The transition probability $a_{ij}$ is defined to be 1, other transition probabilities are defined to be 0.5 because each state has two possibilities to transit to the next process, this state itself or the next state; we can assume these two transitions have same probabilities.

For the digit 0-9, we can construct each model for each digit word, the output probabilities of each model can be calculated from the response values of the signal frames on the phoneme maps, the transition probabilities of each model are defined as the same as those of the model of ‘zero’. For the isolated word recognition, model $M_i$ is selected for which $P(O \mid M_i)$ is a maximum (Figure 5.25). The likelihood, $P(O \mid M_i)$, is approximately by finding the most likely state sequence $X$,

$$\hat{P}(O \mid M_i) = \max_x [P(O, X \mid M_i)]$$

This can be computed very easily using the Viterbi Algorithm, defined in Section 5.2.4.
Unknown O =

\[ P(O | M_1) \quad P(O | M_2) \quad \ldots \quad P(O | M_{10}) \]

Choose Max

Figure 5.25  Isolated word recognition

Suppose the \( M_i \) model is corresponding to the digit word 'zero', the followings are the general steps of how to construct the model and compute the probability of unknown speech frames to this model.

1. **Construct the model based on the phonemes of the word**

   The word 'zero' has four phoneme categories, /z/, /e/, /x/, /ə/. These four phonemes is preceded and followed by silence state. (See Figure 5.24). The phoneme component of other digit word can be seen in Table 5.1.

2. **Estimation of parameters of model**

   Transition probability \( a_{ij} \) --- The transition probability \( a_{12} \) is defined to be 1, other transition probabilities are defined to be 0.5 because each state has two possibilities to transit to the next process, this state itself or the next state; we can assume these two transitions have same probabilities.

   Output probability \( b_i(O_j) \) --- for the central voice frames along speech signal, we extract 3 continuous and 5 separated frames MFCC vectors and then find the maximum response value in each phoneme domain on short and long.

---

**Phoneme Based Speech Recognition Using Self-Organizing Map** - 97 -
phoneme maps, these 20 maximum response values are the estimates of output probabilities of 20 phoneme categories for this central voice frame. we can form the trellis along the frames of unknown speech signal. (Figure 5.15)

(3) **Probability Calculation by Viterbi Algorithm**

For the model 'zero', we can construct the trellis along the input frames to calculate the probability by Viterbi Algorithm. The following figures are the step by step calculation.

![Diagram of Viterbi calculation](image)

Figure 5.26 (a) Viterbi calculation for first speech frame

For the input frame $O_1$, we can get output probabilities $b_i(O_1)$ from the SOM output and calculate the probability from silence to each state $i$ (Figure 5.26 (a)):

$$ r_i(s_i | o_i) = a_{ii} b_i(o_i), \text{ for } i=2, 3, 4, 5 $$

For the input frame $O_2$, we can get $b_i(O_2)$ and probabilities at each state $s_i$ for the frame $O_2$ (Figure 5.26 (b)):

$$ r_2(s_i | o_2) = \max_j (r_j(s_j | o_1) a_{ji} b_i(o_2)) $$
Figure 5.26 (b) Viterbi calculation for second speech frame

Figure 5.26 (c) Viterbi calculation for Nth speech frame

For the input frame $O_N$, we can get $b_l(O_N)$ and probabilities at each state $s_t$ for the frame $O_N$ (Figure 5.26 (c)):

$$r_N(s_t | o_N) = \max_j (r_{N-1}(s_j | o_{N-1})a_{ji}b_l(o_N))$$
Figure 5.26 (d) Viterbi calculation for whole speech frames

The maximum probability for the input frame \( O = [O_1, O_2, \ldots, O_N] \) belong to the ‘zero’ model is calculated as (Figure 5.26 (d)), there is a path \( S \) which is corresponding to this maximum probability.

\[
P(O \mid M_i) = \max_j (r_N(s_j \mid o_N) \alpha_{16})
\]

In the example shown (Figure 5.27) here, we have a simple search path that can recognize ‘zero’ or ‘nine’, both of which have to be preceded and followed by silence. In Viterbi searching, when we look at a new frame, we transition to a new state if the probability of the new category is greater than the probability of the current category. At the end of the search, we have a score for the most likely category sequence and the path through the categories that was used to generate the best score. We can take this path and easily determine the corresponding word. This word has the best fit to the input data, and it is therefore the word that was most likely uttered.
Figure 5.27  Viterbi search path for ‘zero’ and ‘nine’

The above figure traces the search paths for “zero” and “nine” through a hypothetical matrix of probabilities. For a digit utterance, we can search paths for all the 10 digit models by Viterbi search algorithm and find the maximum probability model; this is the recognition result for this digit utterance.

In order to compare with the pattern matching of FNN, we use the same 240 dependent digit samples and another 240 independent digit samples used by SOM & FNN for the speech recognition by our HMM & SOM. The following tables (Table 5.8, 5.9) are the recognition result by HMM based on SOM.
<table>
<thead>
<tr>
<th></th>
<th>Number of samples</th>
<th>Recognition Rate%</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM&amp;HMM</td>
<td>240</td>
<td>97.5</td>
</tr>
<tr>
<td>SOM&amp;FNN</td>
<td>240</td>
<td>91.3(α=2.0, β=0.3)</td>
</tr>
</tbody>
</table>

Table 5.8 Recognition rate of 240 dependent digit samples by FNN and HMM

<table>
<thead>
<tr>
<th></th>
<th>Number of samples</th>
<th>Recognition Rate%</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM&amp;HMM</td>
<td>240</td>
<td>95.8</td>
</tr>
<tr>
<td>SOM&amp;FNN</td>
<td>240</td>
<td>78.7(α=2.0, β=0.3)</td>
</tr>
</tbody>
</table>

Table 5.9 Recognition rate of 240 independent digit samples by FNN and HMM
Chapter 6

Conclusions and Suggestions for Future Works

6.1 Conclusions

This thesis explored the issues involved in designing a phoneme based hybrid digit speech recognizer, which is composed of two layer structures. The first layer is Self-Organizing Phoneme Map (SOPM), it's an unsupervised neural network and the neighboring cells in output layer compete in their activities by means of mutual lateral interactions, and develop adaptively into specific detectors of phoneme signal patterns. The second layer is Fuzzy Neural Network (FNN), it can be constructed into a supervised neural network, and be used for the two-dimensional pattern matching problems. FNN can memorize the new training patterns of signal by fuzzifying the input patterns, which can improve the matching result of similar signal patterns; then a well-trained FNN can recognize the testing patterns of signal by comparing them to the learning patterns stored in the FNN.

Self-Organizing Map as one type of artificial neural network, has the special property that the internal representation of information are generally organized spatially, just like the topographically organized maps found in the brain. There are lots of researches for many applications by SOM. For speech recognition application, researches include phoneme
transcription and recognition. There are few such application examples by SOM for word and sentence recognition because there is a lack of effective ways to deal with 2D spatial response information of words and sentence on SOM.

In this thesis, the fuzzy neural network (FNN) is applied to the phoneme and word recognition based on the phoneme maps. This FNN uses the 2D spatial information of phoneme maps that are the spatial responses of speech signal and broadens the research perspective of speech recognition by SOM.

The analysis and results of this hybrid speech recognizer reveals that the Viterbi search algorithm can obtain the optimal response patterns of speech samples on SOM, and can dramatically improve the recognition rate of digit utterances.

The results also show this hybrid speech recognizer can cope well with the digit-word dependent speech recognition, but for the digit-word independent speech recognition, this recognizer can not work well. This thesis analyzes two kinds of misclassifications of independent speech recognition, and gives a few suggestions for future research.

6.2 Suggestions for Future Work

The aim of this thesis is to explore and discuss the application of SOM for speech recognition, a fuzzy neural network (FNN) is used for the 2D pattern matching of optimal response pattern of speech signal on SOM. So the problem of speech recognition is transformed into a problem of 2D pattern matching.
From the analysis of the experiment results of this hybrid digit speech recognizer, we can see this transformation from speech recognition to pattern matching is a challenging task, I think there are two aspects of the challenge.

The first challenge is we have to obtain the optimal phoneme response pattern of speech signal. The variations of a speech signal make the FNN difficult to recognize the response pattern if no Viterbi algorithm is applied to extract the hidden phoneme sequence of speech utterance, but the Viterbi algorithm itself would obtain the wrong phoneme sequence if some phonemes of a speech utterance have no strong response on the SOM. So we have to include all possible phoneme responses on SOM while have to filter some noisy phoneme responses that will cause misclassifications by the FNN. Another challenge is the limitations of FNN. This FNN has a shortcoming that it could not deal well with noisy patterns while phoneme response patterns have lots of noises. In future research, some update is needed to make this FNN more adaptive for recognition of speech response patterns.
REFERENCES


[38] http://divcom.otago.ac.nz:800/COM/INFOSCI/KEL/speech.htm
Vita Auctoris

Mr. Xiaofeng Dong was born on 22 Feb, 1970, in TongShan, Hubei, P.R. China to Mr. Guangxiang Dong and Mrs. Lijuan Jiang. He has one older brother, Junmin Dong (b.1965).

He graduated and received his Bachelor of Science and Master of Science in Electrical Engineering in July 1992 and July 1995 respectively. In August 1999, he came to Canada to begin his graduate study at University of Windsor.