SPEECH RECOGNITION USING NEURAL NETWORKS

by

Pablo Zegers

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements

For the degree of

MASTER OF SCIENCE

WITH A MAJOR IN ELECTRICAL ENGINEERING

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 9 8

STATEMENT BY AUTHOR

This Thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the copyright holder.

SIGNED: ________________________________

APPROVAL BY THESIS DIRECTOR

This Thesis has been approved on the date shown below:

________________________________	________________________________
Malur K. Sundareshan Professor of Electrical and Computer Engineering	Date

DEDICATION

To God, “for thou hast made us for thyself and restless is our heart until it comes to rest in thee.”

Saint Augustine, *Confessions*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Although speech recognition products are already available in the market at present, their development is mainly based on statistical techniques which work under very specific assumptions. The work presented in this thesis investigates the feasibility of alternative approaches for solving the problem more efficiently. A speech recognizer system comprised of two distinct blocks, a Feature Extractor and a Recognizer, is presented. The Feature Extractor block uses a standard LPC Cepstrum coder, which translates the incoming speech into a trajectory in the LPC Cepstrum feature space, followed by a Self Organizing Map, which tailors the outcome of the coder in order to produce optimal trajectory representations of words in reduced dimension feature spaces. Designs of the Recognizer blocks based on three different approaches are compared. The performance of Templates, Multi-Layer Perceptrons, and Recurrent Neural Networks based recognizers is tested on a small isolated speaker dependent word recognition problem. Experimental results indicate that trajectories on such reduced dimension spaces can provide reliable representations of spoken words, while reducing the training complexity and the operation of the Recognizer. The comparison between the different approaches to the design of the Recognizers conducted here gives a better understanding of the problem and its possible solutions. A new learning procedure that optimizes the usage of the training set is also presented. Optimal tailoring of trajectories, new

insights into the use of neural networks in this class of problems, and a new training paradigm for designing learning systems are the main contributions of this work.

1. INTRODUCTION

1.1 Languages

1.1.1 Introduction

Language is what allows us to communicate, *i.e.* to convey information from one person to another. It is this ability that permits groups of individuals to transform into an information sharing community with formidable powers. Thanks to the existence of language, anyone can benefit from the knowledge of others, even if this knowledge was acquired in a different place or at different times.

Language is a complex phenomenon, and it can appear in very different forms, such as body positions, facial expressions, spoken languages, etc. Not only that, these different types can be used separately or in combination, rendering really complex and powerful methods to communicate information. For example, dance is a method of communication that emphasizes the use of body position and facial expressions. Like dancing, a face to face conversation relies on body gestures and facial expressions, but it also relies on the use of sounds and spoken language. Summing up, when we are speaking about a language, we should not restrict ourselves to think about it as a collection of sounds ordered by some grammar. Any language is far more than that.

Modern language theories state that what we call languages are surface manifestations, or exterior representations, of a common inner core, not directly seen from the outside, where thought processes occur (see Figure 1-1). In other words, they state that we do not think using what we would normally call a language, but internal mental representations that condense all sorts of cognitive processes [1]. Under this point of view, the different languages that a person uses are nothing more than a set of different interfaces between the processing results of that inner core and the community.

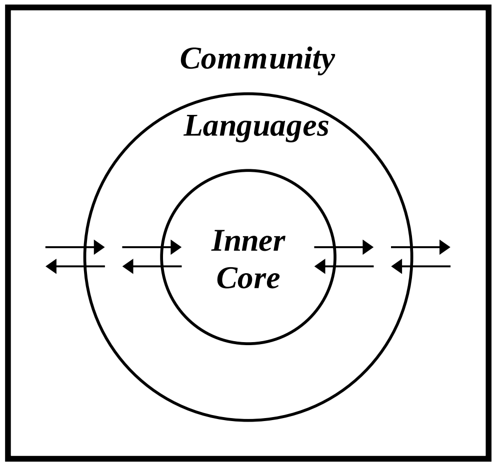

Figure 1-1. Relationship between Languages and Inner Core Thought processes.

Even though under normal conditions we normally use all types of languages at the same time, it can be said that spoken language is the one that conveys the most of the information, while the others normally enhance or back up the meanings conveyed by spoken language.

An amazing discovery done by language researchers is the fact that irrespective of the civilization stage of a culture, its language is as complex as the one developed by any other culture [1]. For example, the languages currently used by certain stone age tribes in New Zealand are as powerful and complex as English, Spanish or any other culture. The same is true with all the languages currently used in the world. This is even true for sign languages: their description power and complexity matches those of spoken language. Quoting Pinker [1], “language is a complex, specialized skill, which develops in the child spontaneously, without conscious effort or formal instruction, is deployed without awareness of its underlying logic, is qualitatively the same in every individual, and is distinct from more general abilities to process information or behave intelligently." Current research in cognitive sciences states that language corresponds more to an instinct [1] than to a socially developed skill, implying that is something inherently human. More than this, it seems to be inherently biological: the capacity of speaking a language is a consequence of human biology, while its manifestation, *i.e.* the actual way we speak, a consequence of the culture where persons live.

Even more, it seems that only humans possess this skill. All studies done in animals, even the more advanced primates, indicate orders of magnitude of difference between human languages and the animal ones. So big is this difference that if language is defined as something similar to human languages, it can be safely considered that animals do not have the ability to communicate using a language.

1.1.2 Spoken Language

The basic building block of any language is a set of sounds named phonemes. As an example, a condensed list of American English phonemes, in their ARPABET representation, is given in Table 1-1 [2].

ARPABET	Example	ARPABET	Example
IY	beat	NX	sing
IH	bit	P	pet
EY	bait	T	10
EH	bet	K	kit
AE	bat	B	bet
AA	Bob	D	debt
AH	but	H	get
AO	bought	HH	hat
OW	boat	F	fat
UH	book	TH	thing
UW	boot	S	sat
AX	about	SH	shut
IX	roses	V	vat
ER	bird	DH	that
AXR	butter	Z	zoo
AW	down	ZH	azure
AY	buy	CH	church
OY	boy	JH	judge
Y	you	WH	which
W	wit	EL	battle
R	rent	EM	bottom
L	let	EN	button
M	met	DX	batter
N	net	Q	glottal stop

Table 1-1. American English ARPABET phonemes and examples.

The phonemes of Table 1-1 comprise all the building blocks needed to produce what is called standard American English. Other versions of English use slightly different sets of phonemes, but similar enough to allow understanding by any English speaker. These differences explain the different accents existing in modern English. As it is in English, all languages follow this structure: they are built upon a basic set of phonemes. It is interesting to note that the number of phonemes in a set of basic phonemes never exceeds seventy. This fact might indicate a limit on the number of sounds a human can produce or distinguish in order to communicate efficiently.

The second level of building blocks of a spoken language consist of the words. These building blocks are built from sequentially concatenated phonemes extracted from the basic set of phonemes of the language. An utterance of the word ‘ZERO’ is shown in Figure 1-2. It can be seen in the figure that there are roughly four distinct zones in the utterance, each of them directly related to one of the four phonemes that comprise the utterance of that word.

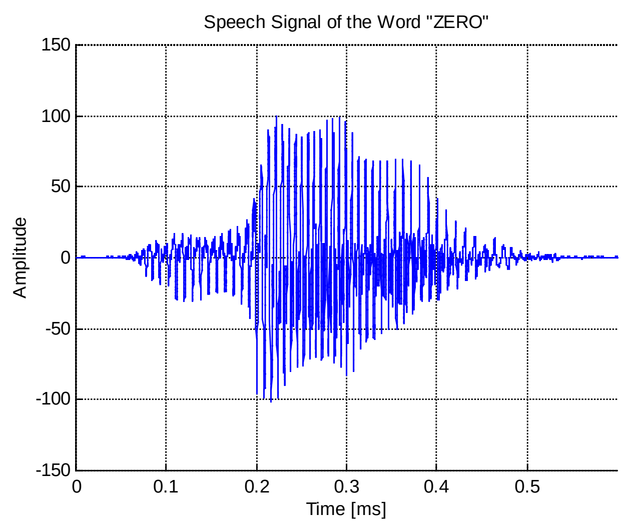

Figure 1-2. Speech Signal for the word ‘ZERO’.

All the words in a language constitute a set called vocabulary. It is generally agreed among experts that a normal person understands and uses an average of 60,000 words [1] of his native language.

The next building block assembles the words into phrases and sentences. They are made up from successively braided words, according to the language's vocabulary, grammar, and the information to be conveyed. Let us consider the following sentences:

- 'OLD HAPPY NEW TABLE POWERLESS'
- 'THE ELEPHANT IRONED THE WRINKLES OF HIS WINGS'
- 'I RECEIVED A PHONE CALL FROM YOUR MOTHER'

The first one is formed by using words from an existing vocabulary, but does not respect a grammar or a context. The second respects a vocabulary and a grammar, but it is not true within a normal context. The third one respects all three aspects.

The topmost level of a language building blocks reflects the way in which the phrases and sentences are combined. Within this level there is an infinite set of possibilities, mainly ranging between two extremes: a lecture mode which does not have interruptions, like a politician's speech, and a conversational mode, which exhibits an enormous amount of interaction, like an argument between any two persons. While the first one does not rely on interaction and tries to convey all the information through spoken language, the other is highly interactive and normally relies in other languages as well.

Phonemes combine into words, words combine into phrases and sentences, phrases and sentences combine to form spoken language. This seems to be all, but it is not. Normally, not only what is said conveys information, but also how it is said. The manner in which expressions are uttered is described by its prosody: the characteristic stresses, rhythms, and intonations of a sentence. Each word has its own pattern of stresses that must be precisely kept, *i.e.* the meaning of the word ‘CONTENT’ depends on which vowel is stressed. A good example of spoken English’s rhythm is provided by the following lines of Tennyson:

‘BREAK, BREAK, BREAK,

ON THY COLD GRAY STONES, O SEA!”

where the time spanned by the utterances between the underlined vowels is normally kept constant, no matter how many phonemes are between them [3]. This rhythm is normally used by all native speakers of the English language in everything they say, and it is one of the aspects that other languages like Spanish or Japanese completely lack. In the case of intonation, the meaning of the sentence “YOU ARE INTELLIGENT” completely changes if a statement-like intonation is used instead of a question-like one.

It is only under ideal conditions that the actual utterance of a word exactly follows the sequences of phonemes that define that word. Normally, the phonemes are slurred, the vowels which are more difficult to pronounce are replaced by easier-to-pronounce ones, and the words are merged such that the ending phoneme of a word is merged with the starting phoneme of the following word, a phenomenon called coarticulation. This occurs because all utterances are produced by a physical system which is always trying to produce the desired output while minimizing the effort necessary to do that. This trade-off normally sacrifices some of the enunciation quality in order to diminish the spent energy. That the speech producing system minimizes energy is seen in the fact that the most used sound in a language normally corresponds to positions where almost no energy is needed to produce the sound. As an example, the vowel sound in the English word

‘BUT’ is generated with all the speech organs in a relaxed position. This is the most used sound in the English language.

A final aspect that explains the variability of an utterance is noise. Noise can be originated by speech glitches, like mouth pops, coughing, false starts, incomplete sentences, stuttering, laughter, lip smacks, heavy breathing, and the environment.

Summing up, the enormous number of possible utterances within a given language is explained by all possible combinations of phonemes, words, phrases and sentences, degree of interaction with other people, possible manifestations of the stresses, rhythms, and intonations, the continuous effort to optimize performed by the speech organs, and noise. It is important to note that there is a set of possible utterances for a given word. It is not necessarily valid to model all of them as deviations from a “correct” one. On the contrary, each of them is equally valid. Within the language context, utterance variability does not necessarily mean utterance deviation. Utterance variability cannot be explained as an ideal template distorted by some error. The concept of error is only useful to explain the always present noise.

1.1.3 Learning Spoken Language

The following description of an infant’s language skills development is based on the description given by Pinker in [1].

Spoken language acquisition is an unconscious process. During their first year of existence, children learn the basic set of phonemes. During the first two months children produce all kinds of sounds to express themselves. After that, they start playing with them and babbling in syllables like ‘BA-BA-BA’. When they are 10 months old, they tune their phoneme hearing capacity such that they start to react more frequently to the sounds spoken by their parents. This transition is done before they understand words, implying that they learn to identify phonemes before they start understanding words.

At the end of the first year they start to understand syllable sequences that resemble speech. By listening to their babbling they receive a feedback that helps them to finish the development of their vocal tract and to learn how to control it.

Around the first year they start understanding words and produce words. The one word stage can last from two months to a year. Around the world, scientists have proven that the content of this vocabulary is similar: half of them are used to designate objects (‘JUICE’, ‘EYE’, ‘DIAPER’, ‘CAR’, ‘DOLL’, ‘BOTTLE’, ‘DOG’, etc.), the rest are for basic actions (‘EAT’, ‘OPEN’, ‘UP’, etc.), modifiers (‘MORE’, ‘HOT’, ‘DIRTY’, etc.), and social interaction routines (‘LOOK AT THAT’, ‘WHAT IS THAT’, ‘HI’, ‘BYE-BYE’, etc.).

Language starts to develop at an astonishing rate at eighteen months. They already understand sentences according to their grammar. They start to produce two and three word utterances and sentences that are correctly ordered despite some missing words.

After late twos, the language employed by children develops so fast that in the words of Pinker [1]: “it overwhelms the researchers who study it, and no one has worked out the exact sequence.” Sentence length and grammar complexity increases exponentially.

Why does it take almost the very first three years of our lives to master a language? Before birth, virtually all nerve cells are formed and are in their proper places, but it is after birth that head size, brain weight, thickness of the cerebral cortex, long distance connections, myelin insulators and synapses grow. It is during these first years that the qualitative aspects of our behavior are developed by adding and chipping away material from our brains. Later, the growth is more devoted to quantity rather than quality.

1.1.4 Written Language

Written language is what permits us to communicate through physical tokens in a completely time independent manner. While speech uses configurations of matter that change through space and time to convey information, written language only uses permanent configurations of matter that do not change over reasonable changes of space and time. It seems to be a small difference, but it turns out to be of transcendental importance. Thanks to this characteristic it is possible to communicate no matter the distance in space and time. As an example, it is due to this that it is possible to read Homero's Odyssey, which was written in Greece thousands of years ago.

Written language is a cultural phenomenon. Not every culture developed it. Moreover, impressive cultures have existed without needing a written language at all. Good examples are all the cultures that flourished in the Americas before the arrival of Christopher Columbus.

Written language correlates uttered words to symbolic expressions. While languages like Chinese correlate complete word utterances to one symbol, languages like Spanish correlate phonemes to symbols. While spelling is meaningless in Chinese, it is straightforward in Spanish. Languages like English are in between these two: while some utterances follow definite and precise rules, others are spelled in a manner unrelated to their enunciation. George Bernard Shaw clearly stated this lack of consistency with the following example: the word ‘FISH’ can be spelled like ‘GHOTI’ using ‘GH’ as in ‘TOUGH’, ‘O’ as in ‘WOMEN’, and ‘TI’ as in ‘NATION’ [1].

1.1.5 Learning Written Language

Written language acquisition is done from learning the correlation between utterances and their symbolic representations. The character and type of this correlation depends on the language. It usually involves learning how to write. This process, as everything related to written language, is a cultural process. While spoken language is normally learned in an unconscious way during the early infancy, written language is an excellent example of a conscious process.

1.2 Speech Recognition

1.2.1 Some Basics

An observer is a system that assigns labels to events occurring in the environment. If the labels belong to sets without a metric distance it is said that the result of the observation is a classification and the labels belong to one of several sets. If, on the contrary, the sets are related by a metric, it is said that the result is an estimation and the labels belong to a metric space. According to these definitions, the goal of this work is to devise an observer that describes air pressure waves using the labels contained by some written language. Because these labels are not related by a metric, the desired process is a classification.

Why does the speech recognition problem attract researchers and funding? If an efficient speech recognizer is produced, a very natural human-machine interface would be obtained. By natural one means something that is intuitive and easy to use by a person, a method that does not require special tools or machines but only the natural capabilities that every human possesses. Such a system could be used by any person able to speak and will allow an even broader use of machines, specifically computers. This potentiality promises huge economical rewards to those who learn to master the techniques needed to solve the problem, and explains the surge of interest in the field during the last 10 years.

If an efficient speech recognition machine is enhanced by natural language systems and speech producing techniques, it would be possible to produce computational applications that do not require a keyboard and a screen. This would allow incredible miniaturization of known systems facilitating the creation of small intelligent devices that can interact with a user through the use of speech [4]. An example of this type of machines is the Carnegie Mellon University JANUS system [13] that does real time speech recognition and language translation between English, Japanese and German. A perfected version of this system could be commercially deployed to allow future customers of different countries to interact without worrying about their language differences. The economical consequences of such a device would be gigantic.

Phonemes and written words follow cultural conventions. The speech recognizer does not create its own classifications and has to follow the cultural rules that define the target language. This implies that a speech recognizer must be taught to follow those cultural conventions. The speech recognizer cannot fully self organize. It has to be raised in a society!

The complexity of the speech recognition problem is defined by the following aspects [2]:

- Vocabulary size, *i.e.* the bigger the vocabulary the more difficult the task is. This is explained by the appearance of similar words that start to generate recognition conflicts, *i.e.* ‘WHOLE’ and ‘HOLE’.
- Grammar complexity.
- Segmented or continuous speech, *i.e.* segmented streams of speech are easier to recognize than continuous ones. In the latter, words are affected by the coarticulation phenomenon.
- Number of speakers, *i.e.* the greater the numbers of speakers whose voice needs to be recognized, the more difficult the problem is.
- Environmental noise.

A speech recognition system, sampling a stream of speech at 8 kHz with 8 bit precision, receives a stream of information at 64 Kbits per second as input. After processing this stream, written words come out at a rate of more or less 60 bits per second. This implies an enormous reduction in the amount of information while preserving almost all of the relevant information. A speech recognizer has to be very efficient in order to achieve this compression rate (more than 1000:1).

In order to improve its efficiency, a recognizer must use as much *a priori* knowledge as possible. It is important to understand that there are different levels of *a priori* knowledge. The topmost level is constituted by *a priori* knowledge that holds true at any instant of time. The lowermost extreme is formed by *a priori* knowledge that only holds valid within specific contexts. In the specific case of a speech recognizer, the physical properties of the human vocal tract remain the same no matter the utterance, and *a priori* knowledge derived from these properties is always valid. On the other extreme, all the *a priori* knowledge collected about the manner in which a specific person utters words is only valid when analyzing the utterances of that person. Based on this fact, speech recognizers are normally divided into two stages, as shown by the schematic diagram in Figure 1-3. The Feature Extractor (FE) block shown in this figure generates a sequence of feature vectors, a trajectory in some feature space, that represents the input speech signal. The FE block is the one designed to use the human vocal tract knowledge to compress the information contained by the utterance. Since it is based on *a priori* knowledge that is always true, it does not change with time. The next stage, the Recognizer, performs the trajectory recognition and generates the correct output word. Since this stage uses information about the specific ways a user produce utterances, it must adapt to the user.

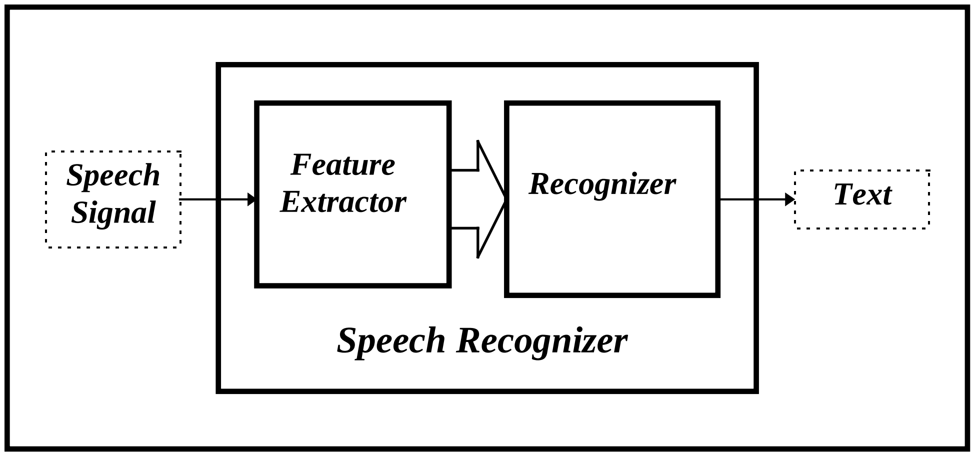

Figure 1-3. Basic building blocks of a Speech Recognizer.

The FE block can be modeled after the stages evidenced in the human biology and development. This is a block that transforms the incoming sound into an internal representation such that it is possible to reconstruct the original signal from it. This stage can be modeled after the hearing organs, which first transduces the incoming air pressure waves into a fluid pressure wave and then converts them into a specific neuronal firing pattern. After the first stage, comes the one that analyzes the incoming information and classifies it into the phonemes of the corresponding language. This Recognizer block is modeled after the functionality acquired by a child during his first six months of existence, where he adapts his hearing organs to specially recognize the voice of his parents.

Once the FE block completes its work, its output is classified by the Recognizer module. It integrates the sequences of phonemes into words. This module sees the world as if it where only composed of words and classifies each of the incoming trajectories into one word of a specific vocabulary.

The process of correlating utterances to their symbolic expressions, translating spoken language into written language, is called speech recognition. It is important to understand that it is not the same problem as speech understanding, a much broader and powerful concept that involves giving meaning to the received information.

1.2.2 A Brief History of Speech Recognition Research

The next paragraphs present a brief summary of the efforts done in the speech recognition field. This summary is based on [2].

Researchers have worked in automatic speech recognition for almost four decades. The earliest attempts were made in the 50's. In 1952, at Bell Laboratories, Davis, Biddulph and Balashek built a system for isolated digit recognition for a single speaker. In 1956, at RCA Laboratories, Olson and Belar developed a system designed to recognize 10 distinct syllables of a single speaker. In 1959, at University College in England, Fried and Dener demostrated a system designed to recognize four vowels and nine consonants. The same year, at MIT's Lincoln Laboratories, Forgie and Forgie built a system to recognize 10 vowels in a speaker independent manner. All of these systems used spectral information to extract voice features.

In the 60's, Japanese laboratories appeared in the arena. Suzuki and Nakata, from the Radio Research Laboratories in Tokyo, developed a hardware vowel recognizer in 1961. Sakai and Doshita, from Kyoto University, presented a phoneme recognizer in 1962. Nagata and coworkers, from NEC Laboratories, presented a digit recognizer in 1963. Meanwhile, in the late 60's, at RCA Laboratories, Martin and his colleagues worked on the non-uniformity of time scales in speech events. In the Soviet Union, Vintsyuk proposed dynamic programming methods for time aligning a pair of speech utterances. This work remained unknown in the West until the early 80's. A final achievement of the 60's was the pioneering research of Reddy in continuous speech recognition by dynamic tracking of phonemes. This research spawned the speech recognition program at Carnegie Mellon University, which, to this day remains a world leader in continuous speech recognition systems.

In the 70's, researchers achieved a number of significant milestones, mainly focusing on isolated word recognition. This effort made isolated word recognition a viable and usable technology. Itakura's research in USA showed how to use linear predictive coding in speech recognition tasks. Sakoe and Chiba in Japan showed how to apply dynamic programming. Velichko and Zagoruyko in Russia helped in the use of pattern recognition techniques in speech understanding. Important also were IBM contributions to the area of large vocabulary recognition. Also, researchers at ATT Bell Labs began a series of experiments aimed at making speech recognition systems truly speaker independent.

In the 80's, the topic was connected word recognition. Speech recognition research was characterized by a shift in technology from template-based approaches to statistical modeling methods, especially Hidden Markov Models (HMM). Thanks to the widespread publication of the theory and methods of this technique in the mid 80's, the approach of employing HMMs has now become widely applied in virtually every speech recognition laboratory of the world. Another idea that appeared in the arena was the use of neural nets in speech recognition problems. The impetus given by DARPA to solve the large vocabulary, continuous speech recognition problem for defense applications was decisive in terms of increasing the research in the area.

Today's research focuses on a broader definition of speech recognition. It is not only concerned with recognizing the word content but also prosody and personal signature. It also recognizes that other languages are used together with speech, taking a multimodal approach that also tries to extract information from gestures and facial expressions.

Despite all of the advances in the speech recognition area, the problem is far from being completely solved. A number of excellent commercial products, which are getting closer and closer to the final goal, are currently sold in the commercial market. Products that recognize the voice of a person within the scope of a credit card phone system, command recognizers that permit voice control of different types of machines, "electronic typewriters" that can recognize continuous voice and manage several tens of thousands word vocabularies, and so on. However, although these applications may seem impressive, they are still computationally intensive, and in order to make their usage widespread more efficient algorithms must be developed. Summing up, there is still room for a lot of improvement and of course, research.

1.2.3 State of the Art

An overview of some of the popular methods for speech recognition is presented in this section following the schematic diagram that outlines the constituent blocks as in

Figure 1-3. The functionality of the individual blocks is also described in order to precisely state the contributions that stem from the work outlined in this thesis.

1.2.3.1 Feature Extractors

The objective of the FE block is to use *a priori* knowledge to transform an input in the signal space to an output in a feature space to achieve some desired criteria. Because *a priori* knowledge is used, the FE block is usually a static module that once designed will not appreciably change. The criteria to be used depend on the problem to be solved. For example, if a noisy signal is received, the objective is to produce a signal with less noise; if image segmentation is required, the objective is to produce the original image plus border maps and texture zones; if lots of clusters in a high dimensional space must be classified, the objective is to transform that space such that classifying becomes easier, etc.

The FE block used in speech recognition should aim towards reducing the complexity of the problem before later stages start to work with the data. Furthermore, existing relevant relationships between sequences of points in the input space have to be preserved in the sequence of points in the output space. The rate at which points in the signal space are processed by the FE block does not have to be the same rate at which points in the feature space are produced. This implies that time in the output feature space could occur at a different rate than time in the input signal space.

A priori knowledge concerning which are the relevant features that should be used in a speech recognition problem comes from very different sources. Results from biological facts, such as the EIH model (which is based on the inner workings of the human hearing system [2]), descriptive methods (like banks of pass band filters [2]), data reduction techniques (such as PCA [5]), speech coding techniques (such as LPC Cepstrum [7]), and neural networks (such as SOM [8]), have been combined and utilized in the design of speech recognition feature extractors. An important result obtained by Jankowski et al [5], which summarizes all of the above mentioned *a priori* knowledge, suggests that under relatively low noise conditions all systems behave in similar ways when tested with the same classifier. This explains why the speech recognition community has adopted the LPC Cepstrum, which is very efficient in terms of computational requirements, as the method of choice.

1.2.3.2 Recognizers

Recognizers deal with speech variability and account for learning the relationship between specific utterances and the corresponding word or words. There are several types of classifiers but only two are mentioned: the Template approach and the approach that employs Hidden Markov Models.

The first one, the Template Approach, is described by the following steps:

1. First, templates for each class to be recognized are defined. There are several methods for forming the templates. One of these consists in randomly collecting a fixed amount

of examples for each class in order to capture the acoustic variability of that specific class. The template of a class is defined as the set of collected examples for that class.

2. Select a method to deal with utterance time warpings. A commonly used technique is a procedure called Dynamic Time Warping [14]. It consists of an adaptation of dynamic programming optimization procedures for time warped utterances.
3. Select some distance measure for comparing an unknown example against the template of each class after time warping correction. This distance measure can be based on geometrical measures, like Euclidean distance, or perceptual ones, like Mel or Bark scales [2].
4. Compare unknown patterns against the collected templates using the selected time warping correction and the chosen distance measurement. The unknown pattern is classified according to the values obtained from the distance measurements.

One drawback of the template procedure is that nothing guarantees that the chosen examples would really capture the class variability. The main problem with this method is that several utterances must be collected in order to capture the word's variability. For small vocabularies this may not be a problem, but for large ones it is something unthinkable: no user would willingly utter thousands and thousands of examples.

Another approach is based on HMM, and solves the temporal and acoustic problems at once using statistical considerations. It is described by the following steps [9]:

1. Like the Template Approach, something that stores the knowledge about the possible variations of a class must be defined. The difference is that in this case it is an HMM

instead of a template. Consider a system that may be described at any time by means of a set of M distinct states, such as a word which can be represented by a collection of phonemes. At regularly spaced, discrete times, the system undergoes a change of state according to a set of probabilities associated with each state. Let us assume that each time the system moves to another state, an output selected from a set of N distinct outputs is produced. The outputs are selected according to a probability mass function. The states are hidden from the outside world, which only observes the outputs. In other words, a HMM is an embedded stochastic process with an underlying stochastic process that is not directly observable but can be observed only through another set of stochastic processes that produces the sequence of observations. In order to obtain the HMM that represents a class, the number of hidden states, state transition probabilities, and output generation probabilities must be defined. Usually, the number of states is defined by trial and error. The probabilities are instead obtained through iterative methods that work with sets of training examples. An example of an HMM is shown in Figure 1-4, where the solid circles represent the possible states, and the arrows represent the transitions between states as an utterance progresses. In the most basic level, a HMM is used to model a word and each state represents a phoneme. The number of states used to model a word depends on the number of phonemes needed to describe that word.

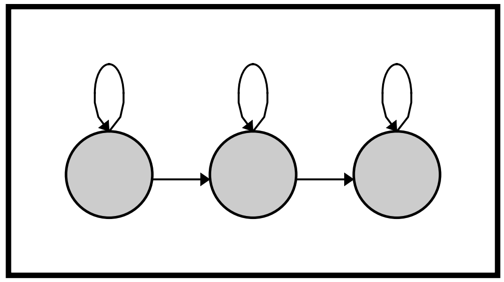

Figure 1-4. Hidden Markov Model example.

2. Once the HMMs have been determined, whenever an unknown utterance is presented to the system, each model evaluates the probability of producing the output sequence associated to that utterance. The word associated to the model with the highest probability is then assigned to the unknown utterance.

The HMM method does not have some of the drawbacks exhibited by the Template Approach. Because it models the utterances as stochastic processes, it can capture the variability within a class of words. In terms of scalability, it offers much more flexibility. Instead of defining one HMM for each word, a hierarchy of HMMs is built, where the bottommost layer of HMMs models phonemes, a second layer which models short sequences of connected phonemes and uses the output of the first layer as input, and a final layer which models each word and uses the output of the second layer as input. This hierarchical approach constitutes the most successful approach ever devised for speech recognition.

The limitations of the HMMs are that they rely on certain *a priori* assumptions that do not necessarily hold in a speech recognition problem [6] [15]:

- The First Order Assumption, that all probabilities depend solely on the current state, is false for speech applications. This is the reason why HMMs have strong problems modeling coarticulations, where utterances are in fact strongly affected by recent state history.
- The Independence Assumption, that there is no correlation between adjacent time frames, is also false. At a certain instant of time, both words can be described by the same state but the only thing that differentiates them is their behavior before that state, which is something completely ignored by the HMM model.

A characteristic common to all speech recognizers is that they are composed of layered hierarchies of sub-recognizers. Normally, the first layer is composed of a set of sub-recognizers whose output is integrated by the following layer of sub-recognizers, and so on, until the desired output is obtained. An example is the architecture used in a HMM based recognizer: the first layer is composed of a set of HMMs, each of them specialized in recognizing phonemes. The second layer, again composed of a set of HMMs, uses as input the output of the first layer and specializes in recognizing collections of phonemes. The third layer, based on HMMs too, uses the output of the second layer to recognize words. Finally, the fourth layer integrates the words into sentences using built-in knowledge about the grammar of the target language.

1.3 Contributions of this Thesis

The main contributions of this thesis are the following:

- The speech recognition problem is transformed into a simplified trajectory recognition problem. The trajectories generated by the FE block are simplified while preserving most of the useful information by means of a SOM. Then, word recognition turns into a problem of trajectory recognition in a smaller dimensional feature space.
- Another contribution is that the comparison between the different recognizers conducted here sheds new light on future directions of research in the field.
- Finally, a new paradigm that simplifies the training of a learning system is presented. The procedure allows a better usage of the training examples and reduces the computational costs normally associated with the design of a learning system.

It is important to understand that it is not the purpose of this work to develop a full-scale speech recognizer but only to test new techniques and explore their usefulness in providing more efficient solutions.

1.4 Organization of this Thesis

The major results of this research are presented following the conceptual division stated in Figure 1-3. The architecture used for the implementation of the FE block is detailed first. Then, a comparison between the different recognizers is presented. The thesis concludes by commenting on the results and proposing future steps for continuing this type of research.

1.5 Computational Resources

All the software used in this thesis to obtain the results was coded by the author in C++ and run in a Pentium class machine. All the plots were produced with the Matlab package.

2. NEURAL NETWORKS AND SPEECH RECOGNITION

2.1 Biological Neural Networks

The central idea behind most of the developments in neural sciences is that a great deal of all intelligent behavior is a consequence of the brain activity [16]. Even more, the different brain functions emerge as a result from the dynamic operation of millions and millions of interconnected cells called neurons. Roughly speaking, a typical neuron receives its inputs through thousands of dendrites, processes them in its soma, transmit the obtained output through its axon, and uses synaptic connections to carry the signal of the axon to thousands of other dendrites that belong to other neurons.

It is the manner in which this massively interconnected network is structured and the mechanisms by which neurons produce their individual outputs that determine most of the intelligent behavior. These two aspects are determined in living beings by evolutionary mechanisms and learning processes.

2.2 Artificial Neural Networks

During the last 30 years, a new computational paradigm based on the ideas from neural sciences has risen [17]. Its name, neural networks, comes from the fact that these techniques are based upon analogies drawn from the inner workings of the human nervous system. The main idea behind the neural network paradigm is to simulate the behavior of the brain using interconnected abstractions of the real neurons (see Figure 2-1), to obtain systems that exhibit complex behaviors, and hopefully intelligent ones.

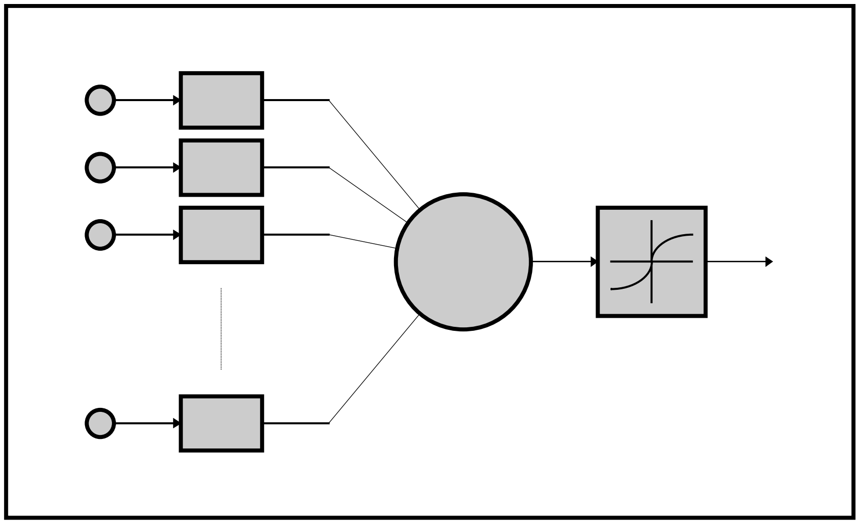

Figure 2-1. Artificial neuron.

In an artificial neuron, numerical values are used as inputs to the “dendrites.” Each input is multiplied by a value called weight, which simulates the response of a real dendrite. All the results from the “dendrites” are added and thresholded in the “soma.” Finally, the thresholded result is sent to the “dendrites” of other neurons through an “axon.” This sequence of events can be expressed in mathematical terms as:

$$y = f\left(\sum_{i=1}^{n} w_i x_i\right) \tag{2.1}$$

where x_i is the input received by a dendrite, w_i the weight associated to the i^{th} “dendrite,” $f(\ )$ a threshold function, y the output of the neuron.

Although the model suggested by Figure 2-1 is a simplistic reduction of a real neuron, its usefulness is not based on the individual capacities that each individual neuron exhibits but on the emergent behavior that arises when these simple neurons are interconnected to form a neural network. As in biological neurons, the interconnection architecture of the neurons and the manner in which neurons process their inputs determines the behavior exhibited by the neural network.

In contrast with biological neural networks, the architecture of a neural network is set by a design process, and the parameters that define the way its neurons process their respective inputs are obtained through a learning process. The learning process involves a search in a parameter space for determining the best parameter values according to some optimality criteria.

2.2.1 Self Organizing Map

The Self Organizing Map (SOM) is a neural network that acts like a transform which maps an m-dimensional input vector into a discretized n-dimensional space while locally preserving the topology of the input data [18]. The expression "locally preserving the topology" means that for certain volume size in the input space, points that are close together in the input space correspond to neurons that are close in the output space. This is the reason that explains why a SOM is called a feature map [19]: relevant features are extracted from the input space and presented in the output space in an ordered manner. It is always possible to reverse the mapping and restore the original set of data to the original m-dimensional space with a bounded error. The bound on this error is determined by the architecture of the network and the number of neurons. The SOM considers the data set as a collection of independent points and does not deal with the temporal characteristics of the data. It is a very special transform in the sense that it reexpresses the data in a space with a different number of dimensions while preserving some of the topology.

A useful training procedure for a SOM is as follows [20]:

1. The neurons are arranged in a n-dimensional lattice. Each neuron stores a point in an m-dimensional space.
2. A randomly chosen input vector is presented to the SOM. The neurons start to compete until the one that stores the closest point to the input vector prevails. Once the dynamics of the network converges, all the neurons but the prevailing one will be inactive. The output of the SOM is defined as the coordinates of the prevailing neuron in the lattice.
3. A neighborhood function is centered around the prevailing neuron of the lattice. The value of this function is one at the position of the active neuron, and decreases with the distance measured from the position of the winning neuron.
4. The points stored by all the neurons are moved towards the input vector in an amount proportional to the neighborhood function evaluated in the position of the lattice where the neuron being modified stands.
5. Return to 2, and repeat steps 2, 3, and 4 until the average error between the input vectors and the winning neurons reduces to a small value.

After the SOM is trained, the coordinates of the active neuron in the lattice are used as its outputs.

Figure 2-2 shows a mapping from the unitary square in a 2-dimensional space onto a lattice of neurons in a 1-dimensional space. The resulting mapping resembles a Peano space-filling curve [21].

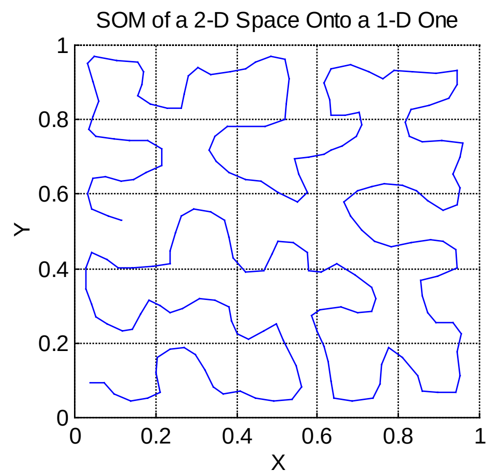

Figure 2-2. Unit Square mapped onto a 1-D SOM lattice.

2.2.2 Multi-Layer Perceptron

The functionality of the Multi-Layer Perceptron (MLP) can be best defined by the result proven by Kolmogorov, then rediscovered by Cybenko [22], Funahashi [23], and others. They proved that an MLP with one hidden layer with enough neurons is able to approximate any given continuous function. The MLP is a universal function approximator. As in the case of the SOM, the MLP does not deal with temporal behavior.

An MLP is comprised of several neurons arranged in different layers (see Figure 2-3). The layer that consists of the neurons which receive the external inputs is called the input layer. The layer that produces the outputs is called the output layer. All the layers between the input and the output layers are called hidden layers. Any input vector fed into the network is propagated from the input layer, passing through the hidden layers, towards the output layer. This behavior originated the other name for this class of neural networks, *viz.* feed-forward neural networks.

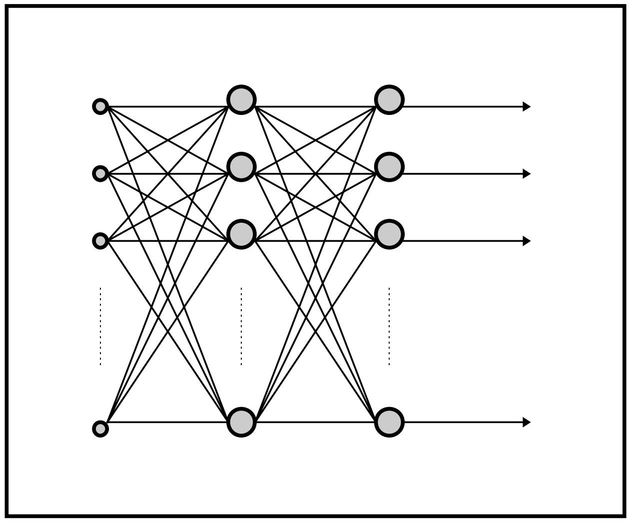

Figure 2-3. MLP with 1 hidden layer.

The parameters of the MLP are set in a training stage during which the data pairs formed by the input vectors and the corresponding output vectors are used to guide a parameter search algorithm. One approach consists in minimizing the errors caused by the training examples by means of a gradient descent algorithm. This procedure is called Error backpropagation method [17]. It is important to note that this is only one approach and many more are available, like reinforced learning, simulated annealing, genetic algorithms, etc.

2.2.3 Recurrent Neural Networks

A Recurrent Neural Network (RNN) is similar to a MLP but differs in that it also has feedback connections. Experimental results show that like a MLP, these networks also perform as universal function approximators. Experimental results also show that RNNs are able to approximate simple functions between temporal trajectories in the input space and temporal trajectories in the output space. Although they constitute a generalization of the MLP, nobody has yet proven that they are universal function approximators. So far, the only proof comes from Funahashi, who proved that a RNN, provided that is has enough neurons, can generate any finite time trajectory [24]. An example of a RNN architecture is shown in Figure 2-4. The main difference with the MLP consists of the existence of feedback connections between the neurons. This allows these type of neural networks to display all type of dynamical behaviors.

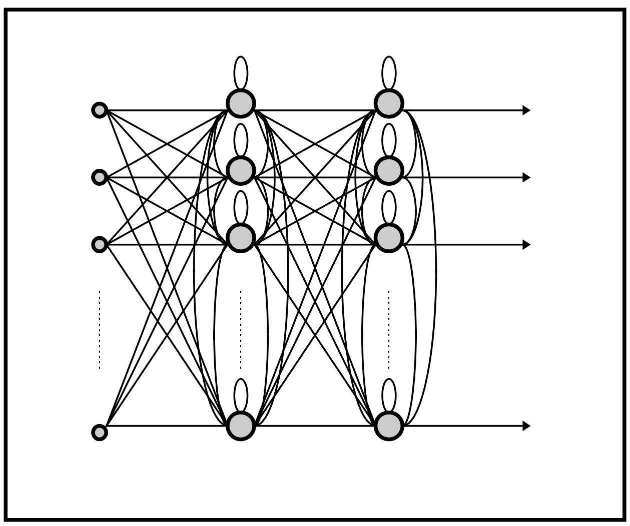

Figure 2-4. RNN architecture.

2.3 Neural Networks and Speech Recognition

Since the early eighties, researchers have been using neural networks in the speech recognition problem. One of the first attempts was Kohonen's electronic typewriter [25]. It uses the clustering and classification characteristics of the SOM to obtain an ordered map from a sequence of feature vectors. The training was divided into two stages, where the first of these was used to obtain the SOM. Speech feature vectors were fed into the SOM until it converged. The second training stage consisted in labeling the SOM, *i.e.* each neuron of the feature map was assigned a phoneme label. Once the labeling process was completed, the training process ended. Then, unclassified speech was fed into the system, which was then translated it into a sequence of labels. This way, the feature extractor plus the SOM behaved like a transducer, transforming a sequence of speech samples into a sequence of labels. Then, the sequence of labels was processed by some AI scheme in order to obtain words from it.

Another approach is Waibel's Time Delay Neural Network (TDNN) [26]. It used a modified MLP to capture the space deviations and time warpings in a sequence of features. One input layer, two hidden layers and, one output layer were used to classify the different phonemes produced by English native speakers. The weights that defined the TDNN were defined such that the system was somewhat invariant to time warpings in the speech signal. It only recognized speech at a phoneme level and it was not used to make decisions in longer time spans, *i.e.*, it was not directly used for word recognition.

As larger segments of speech are considered, approaches like the Electronic Typewriter or the TDNN become less useful. It is difficult for these approaches to deal with the time warpings, a problem which so far has impeded the neural networks to be successfully employed in the speech recognition problem. To integrate large time spans has become a critical problem and no technique using neural networks has yet been devised to solve this problem in a satisfactory manner.

In order to address the problem stated above, hybrid solutions have been used instead. Usually, after the phoneme recognition block, either HMM models [27] or Time Delay Warping (TDW) [28] measure procedures are used to manipulate the sequences of features produced by the feature extractors. In this thesis, alternate approaches, solely based on neural networks are developed.

3. DESIGN OF A FEATURE EXTRACTOR

As stated before, in a speech recognition problem the FE block has to process the incoming information, *viz.* the speech signal, such that its output eases the work of the classification stage. The approach used in this work to design the FE block divides it into two consecutive sub-blocks: the first is based on speech coding techniques, and the second uses a SOM for further optimization.

3.1 Speech Coding

Why do we employ speech coding techniques to design the FE block? Because it has been extensively proven that speech coding techniques produce compact representations of speech which can be used to obtain high quality reconstructions of the original signal. These techniques can provide a compression of the incoming data while preserving its content. Digital telephony is a field that heavily relies on these techniques and its results prove that these methods are quite effective.

3.1.1 Speech Samples

The context of the present work is speech recognition in a small vocabulary, segmented speech, and a single speaker. For developing experimental results, the set of English digits, recorded with a pause between them, and uttered by the author, was used.

All the examples were uttered in a relatively quiet room. No special efforts were made in order to diminish whatever noises were being produced at the moment of recording.

3.1.2 Speech Sampling

The speech was recorded and sampled using an off-the-shelf relatively inexpensive dynamic microphone and a standard PC sound card. The incoming signal was sampled at 11,025 Hertz with 8 bits of precision.

It might be argued that a higher sampling frequency, or more sampling precision, is needed for achieving a high recognition accuracy. However, if a normal digital phone, which samples speech at 8,000 Hertz with an 8 bit precision is able to preserve most of the information carried by the signal [2], it does not seem necessary to increase the sampling rate beyond 11,025 Hertz, or the sampling precision to something higher than 8 bits. Another reason behind these settings is that commercial speech recognizers typically use comparable parameter values and achieve impressive results.

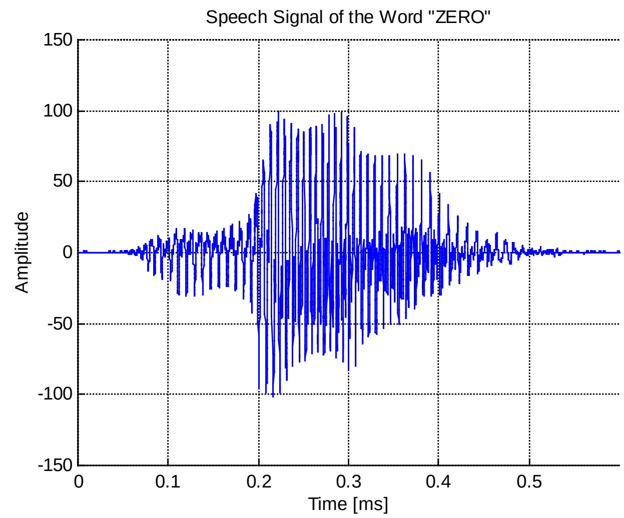

Figure 3-1. Speech signal for the word ‘ZERO’, sampled at 11,025 Hertz with a precision of 8 bits.

3.1.3 Word Isolation

Despite the fact that the sampled signal had pauses between the utterances, it was still needed to determine the starting and ending points of the word utterances in order to know exactly the signal that characterized each word [2]. To accomplish this, a moving average of the sampled values was used. After the end of a word was found, whenever the moving average exceeded a threshold, a word start point was defined, and whenever the moving average fell below the same threshold, a word stop point was defined. Both the temporal width of the moving average and the threshold were determined experimentally using randomly chosen utterances. Once those values were set, they were not changed. It must be noted that no correction for mouth pops, lip smacks or heavy breathing, which can change the starting and ending points of an utterance, was done.

3.1.4 Pre-emphasis Filter

As is common in speech recognizers, a pre-emphasis filter [2] was applied to the digitized speech to spectrally flatten the signal and diminish the effects of finite numeric precision in further calculations.

The transfer function of the pre-emphasis filter corresponds to a first order FIR filter defined by:

$$H(w) = 1 - ae^{-jw} \tag{3.1}$$

A value of $a = 0.95$, which leads to a commonly used filter [2], was used. This type of filter boosts the magnitude of the high frequency components, leaving relatively untouched the lower ones.

3.1.5 Speech Coding

After the signal was sampled, the utterances were isolated, and the spectrum was flattened, each signal was divided into a sequence of data blocks, each block spanning 30 ms, or 330 samples, and separated by 10 ms, or 110 samples [2]. Next, each block was multiplied by a Hamming window, which had the same width as that of the block, to lessen the leakage effects [2][28]. The Hamming window is defined by:

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi\ n}{N-1}\right) \quad , \quad n \in [0, N-1] \quad and \quad N = 330 \tag{3.2}$$

Then, a vector of 12 Linear Predicting Coding (LPC) Cepstrum coefficients was obtained from each data block using Durbin's method [2] and the recursive expressions developed by Furui [7]. The procedure is briefly outlined in the following:

1. First, the auto-correlation coefficients were obtained using:

$$r(m) = \sum_{n=0}^{N-1-m} x(n)x(n+m) \quad , \quad m = 12 \quad and \quad N = 330 \tag{3.3}$$

   where $x(i)$ corresponds to the speech sample located at the i^{th} position of the frame.

2. Then, the LPC values were obtained using the following recursion:

$$\begin{aligned}
l = 0 \quad &: \quad E(l) = r(l) \\
l = 1 \quad &: \quad k(l) = \frac{r(l)}{E(l-1)} \\
&\quad\quad E(l) = E(l-1)\left\{1 - k(l)^2\right\} \\
&\quad\quad a_l^l = k(l) \\
l > 1 \quad &: \quad k(l) = \frac{1}{E(l-1)}\left(r(l) - \sum_{i=1}^{l-1} a_i^{l-1} r(l-i)\right) \\
&\quad\quad E(l) = E(l-1)\left\{1 - k(l)^2\right\} \\
&\quad\quad a_m^l = a_m^{l-1} - k(l)a_{l-m}^{l-1} \\
&\quad\quad a_l^l = k(l)
\end{aligned} \tag{3.4}$$

   where $m \in [1, l]$, $\forall l \in [1, p]$, and $a_n = a_n^p$ is the n^{th} LPC coefficient. A value of $p = 12$ was used in this feature extractor [2].

3. Finally, the LPC Cepstrum coefficients were obtained using:

$$c_m = \begin{cases} a_m + \sum_{k=1}^{m-1} \dfrac{kc_k a_{m-k}}{m} & , \quad m \in [1, p] \\ \sum_{k=1}^{m-1} \dfrac{kc_k a_{m-k}}{m} & , \quad m > p \end{cases} \tag{3.5}$$

   where a value of $m = 12$ was used in the feature extractor, resulting in 12 LPC Cepstrum values per frame.

As a result of the LPC Cepstrum extraction procedure, each utterance was translated into a sequence of points, each belonging to the LPC Cepstrum feature space of dimension 12, and each 10 ms apart. In other words, this procedure translates an air pressure wave into a discretized trajectory in the LPC Cepstrum feature space.

Figure 3-2 and Figure 3-3 contain one example of the evolution of the LPC Cepstrum components for each of the digits. It is interesting to note in these plots that the length in time of the different utterances differs by important amounts. The shortest utterance lasts approximately 30 [ms], while the longest around 80 [ms]. It can also be appreciated that the values of the parameters also differ by important amounts. These differences between the utterances will be of great importance for the task of the Recognizer. Ideally speaking, the FE block should process the sampled speech such that the output clearly differs between different classes while the within class variance is kept as low as possible.

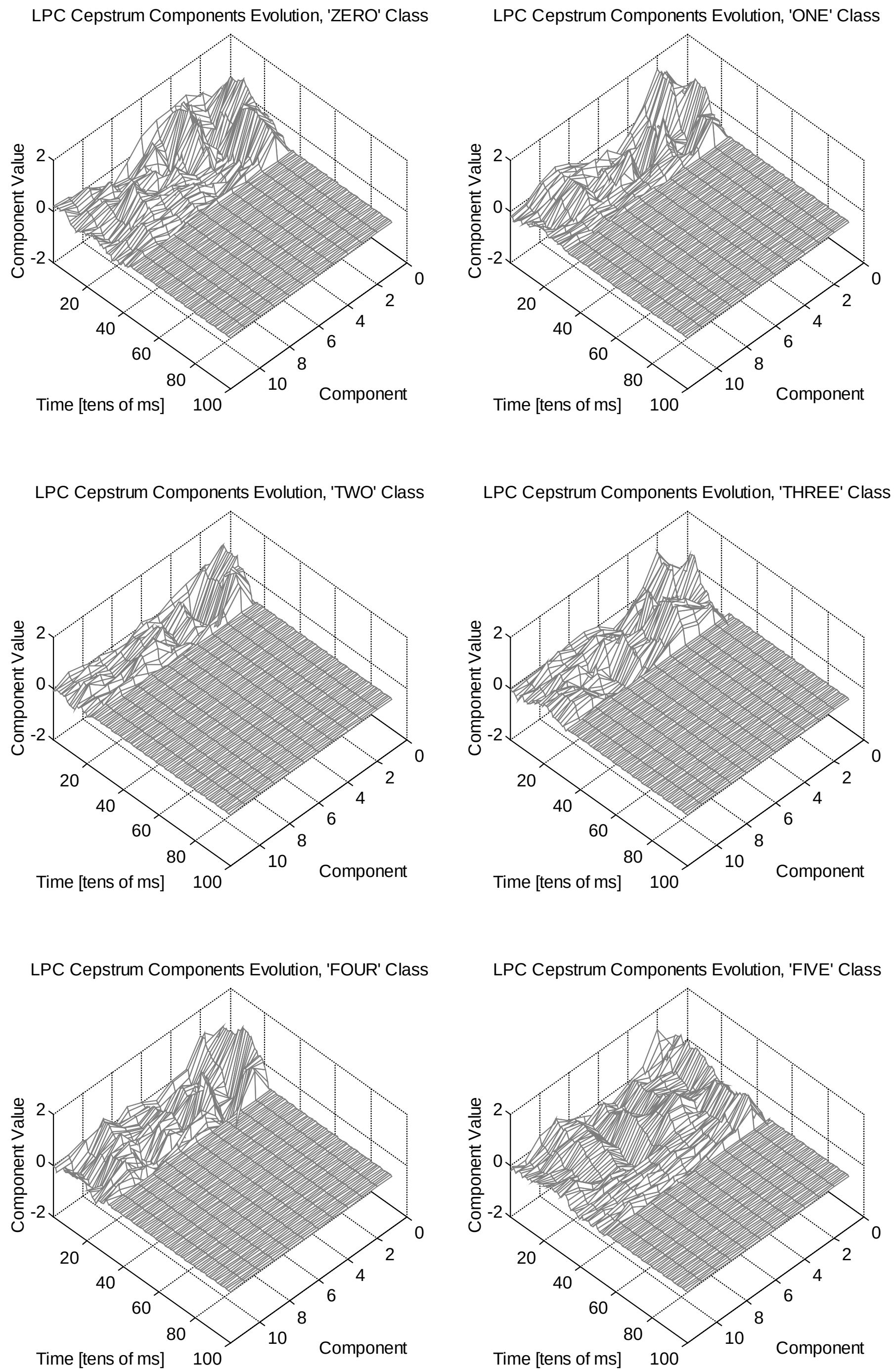

Figure 3-2. Examples of LPC Cepstrum evolution for all the digits.

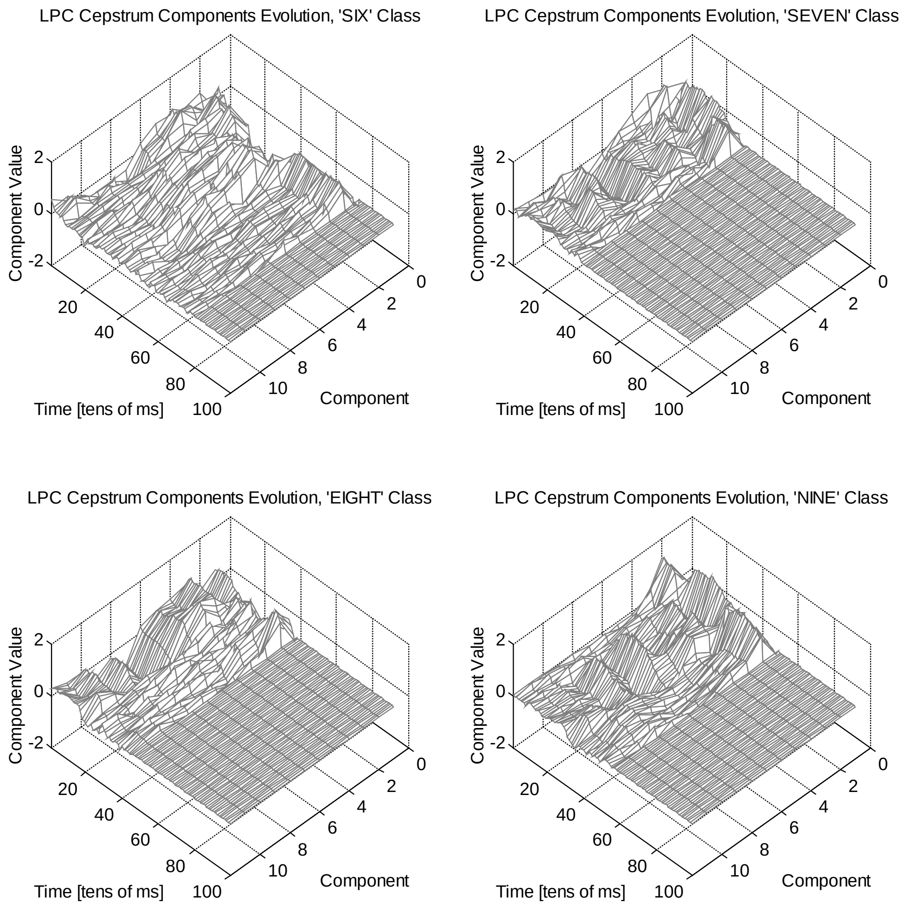

Figure 3-3. Examples of LPC Cepstrum evolution for all the digits. (continuation)

3.2 Dimensionality Reduction

Once the utterance is translated into a trajectory in some feature space, several options exist:

- The most common approach is to use the obtained trajectory to generate a sequence of labels, normally by means of a vector quantization (VQ) scheme [29]. This sequence is then used for recognition. As an example, Carnegie mellon's SPHINX speech recognition system [9] fed the output of the speech coding scheme into a VQ system which translated the incoming data into a sequence of phonemes. The SPHINX system then used an HMM approach to process the sequences of labels and recognize the words. Although it cannot be said that this approach is inferior because it has been successfully used in practical speech recognition systems, it is still throwing away information. It does not keep information about the topological relationships between the output labels. In other words, for any sequence of labels there is no indication concerning the closeness in the feature space of the vectors that generated these labels.
- Another approach that may be found in the literature is to directly use the obtained trajectory for recognition. This route was taken by Hasegawa et al [10], who recognized trajectories in a feature space with 20 dimensions. The problem with this approach is that processing trajectories in spaces with as many dimensions as 20 normally demands an excessive amount of computational resources.

One of the objectives of the present work is to convert the word recognition problem into a trajectory recognition problem, similar to the approach taken by Hasegawa et al [10], but with a modification, *i.e.* the dimensionality of the trajectories is reduced before feeding them into the recognizer block. In this manner, the trajectory classification is highly simplified.

Desire for achieving dimensionality reduction pervades several applications and several methods have been devised to accomplish it. This is specially true in the field of Automatic Control, where the characterization of highly nonlinear high order dynamic systems poses some major challenges for designing efficient control policies. Some examples [11] of the approaches used in this field to reduce the order of the system to be controlled are: Modal Aggregation, Continued Fraction Reduction, Moment Matching, and Pade Approximations. All of these methods require a previous identification of the system to be controlled. Once the system to be controlled is identified, the mathematical expression that relates the states of the system to its inputs and outputs is manipulated according to one of the procedures stated above in order to obtain a simplified version whose behavior resembles that of the original system. From the perspective of the speech recognition problem, the limitation of these approaches is that they simplify the model that generates the trajectories, not the trajectories themselves. The speech recognition problem is not a system identification problem, but a trajectory recognition one.

Even more, despite the fact that the utterances are produced by a biological system, a system that necessarily produces continuous outputs, different utterances can represent the same word. It is important to note that each of these alternate utterances is valid and none of them can be regarded as a deviation from some ideal way to enunciate that word or as an incorrect output. In other words, there is no one-to-one relationship between the set of possible utterances and the class to which they belong: one utterance necessarily implies only one class, but a class does not necessarily imply only one utterance. This aspect makes any analytical representation of the problem more complex than it could be expected.

Using the fact that the SOM is a VQ scheme that preserves some of the topology in the original space [8], the basic idea behind the approach employed in this work is to use the output of a SOM trained with the output of the LPC Cepstrum block to obtain reduced state space trajectories that preserve some of the behavior of the original trajectory. The problem is now reduced to find the correct number of neurons for constituting the SOM and their geometrical arrangement.

The SPHINX speech recognition system quantized the trajectories, which belonged to a space of 12 dimensions, using a VQ scheme with 256 vectors to generate a sequence of labels [9]. The SPHINX system achieved a high recognition accuracy in a much more difficult problem than the one being investigated in this work. It consisted of recognizing words from a 1000 word vocabulary, under continuous speech conditions, and in the presence of multiple speakers. Based on these results, since a SOM is a VQ scheme that preserves some of the topology of the original space, a SOM with as many neurons as vectors the SPHINX system had in its codebook set should be capable of an efficient quantization of the input space. In other words, it should be capable of at least retaining the amount of information needed to achieve the recognition accuracy reached by the SPHINX system.

Based on the ideas stated above, it was decided that a SOM with 256 neurons was enough to reduce the dimensionality of the trajectories while keeping enough information to achieve a high recognition accuracy. The SOM was arranged in a 2-D lattice. The coordinates of the lattice were used as the coordinates of the reduced space trajectory. The fact that the outcome is a 2-D trajectory which can be graphically represented for visual display was not particularly important for our decision about the number of dimensions of the output space.

Figure 3-4, Figure 3-5 and Figure 3-6 show the temporal evolution of the two components of the trajectories for each of the digits. It also depicts the same trajectory projected onto the reduced feature space, *i.e.* the trajectory was projected along the time axis onto the reduced feature space such that all of its temporal characteristics are not shown.

What remains to be demonstrated is whether the SOM mapped similar utterances onto similar trajectories. In other words, did the SOM preserve some of the original topology of the trajectories? The answer to this question will be given in the following chapter of this thesis.

It must be noted that a similar approach was used by Kohonen [8], but instead of reducing the dimensionality of the trajectory, he used the SOM to generate a sequence of labels, which was then used by a word classifier. Thus the present use of the SOM for order reduction in trajectory representations is a novel one, and this application can find appropriate use in other problems besides speech recognition where similar trajectories arise.

3.3 Optional Signal Scaling

The output of the SOM is defined by the coordinates of the neuron that was activated by the input. The output values may need to be scaled before they are fed into the Recognizer. All the approaches used in this work to design a recognizer, with the exception of the one that employed a RNN, did not require scaling of the outputs.

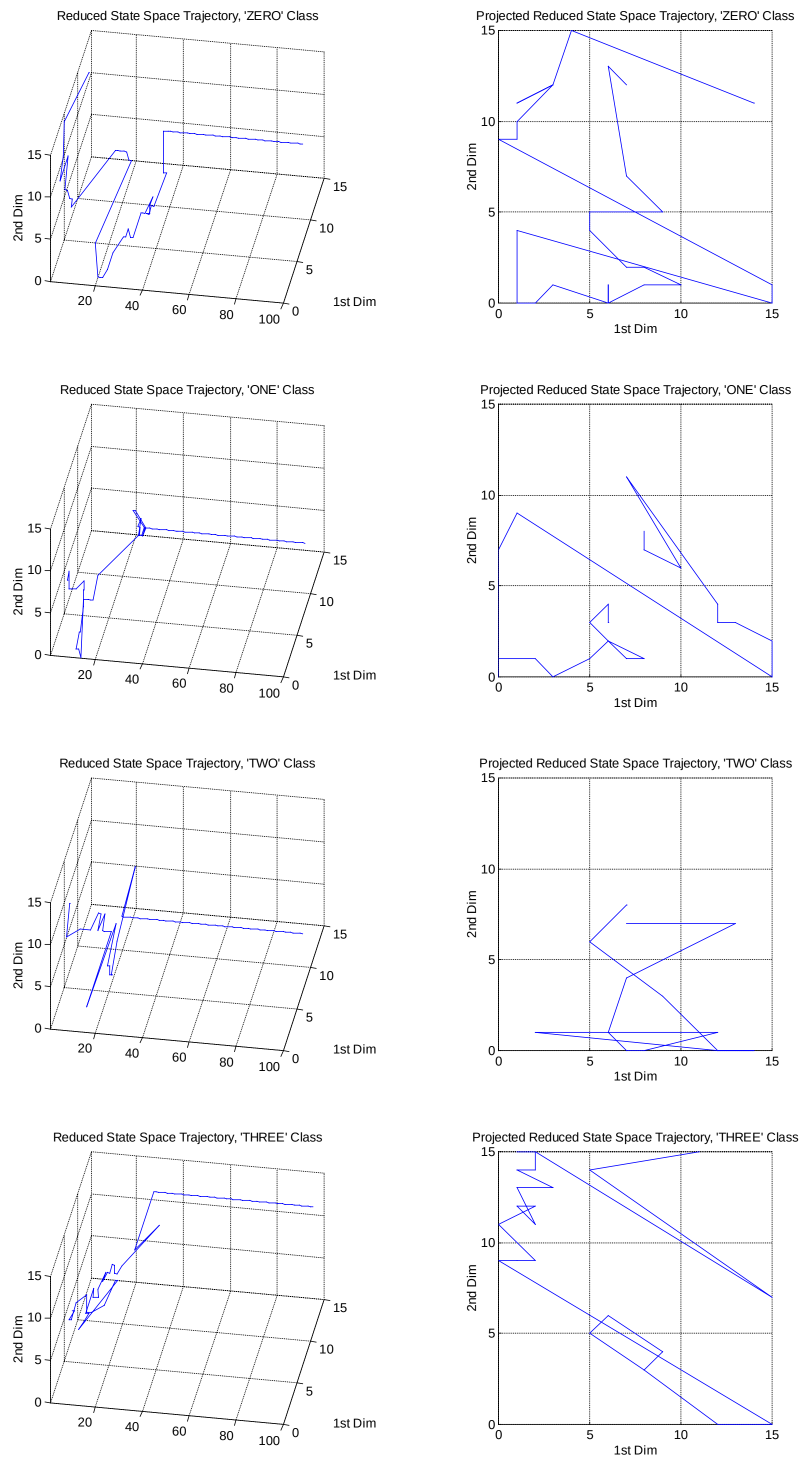

Figure 3-4. Reduced feature space trajectories, all digits.

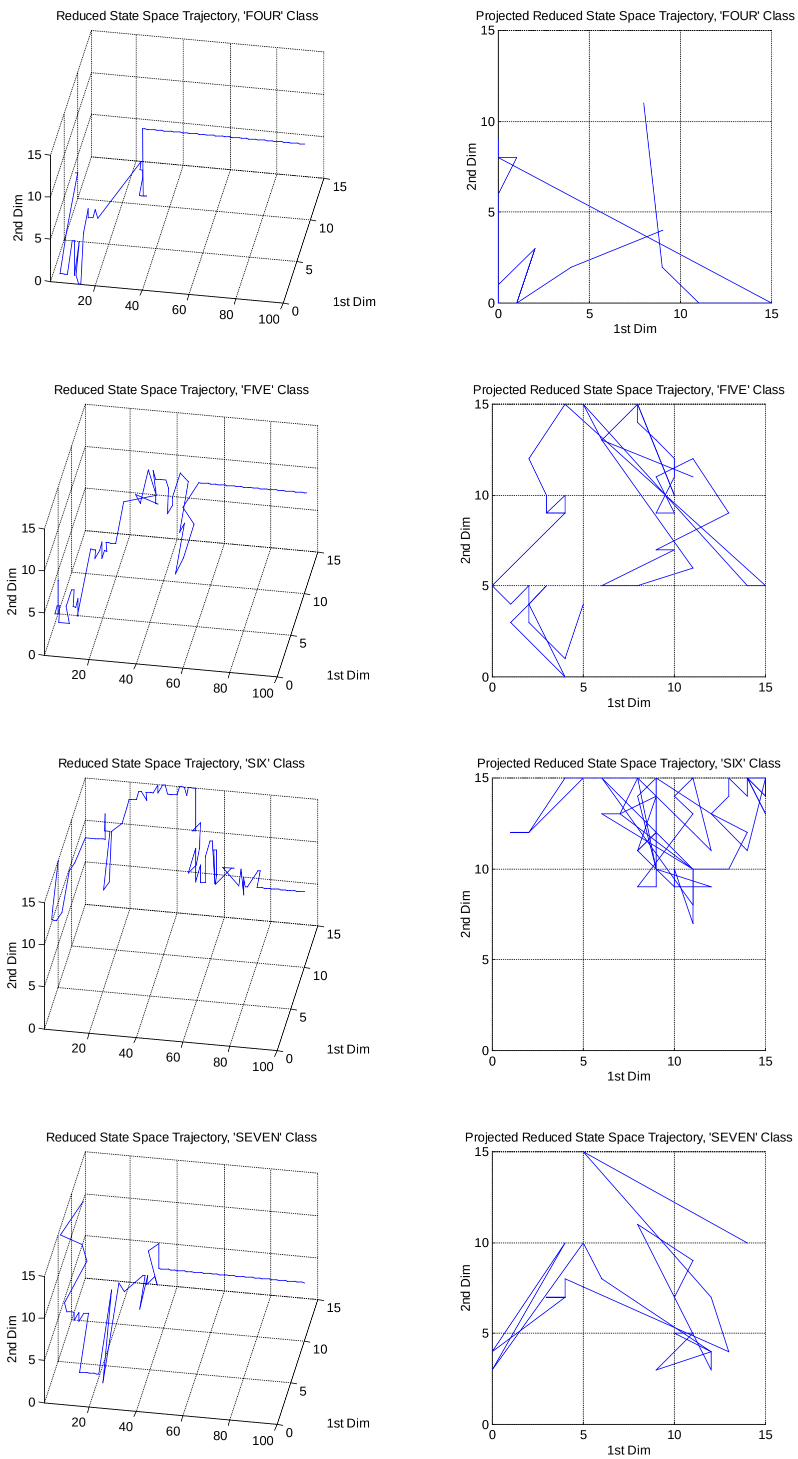

Figure 3-5. Reduced feature space trajectories, all digits. (Continuation)

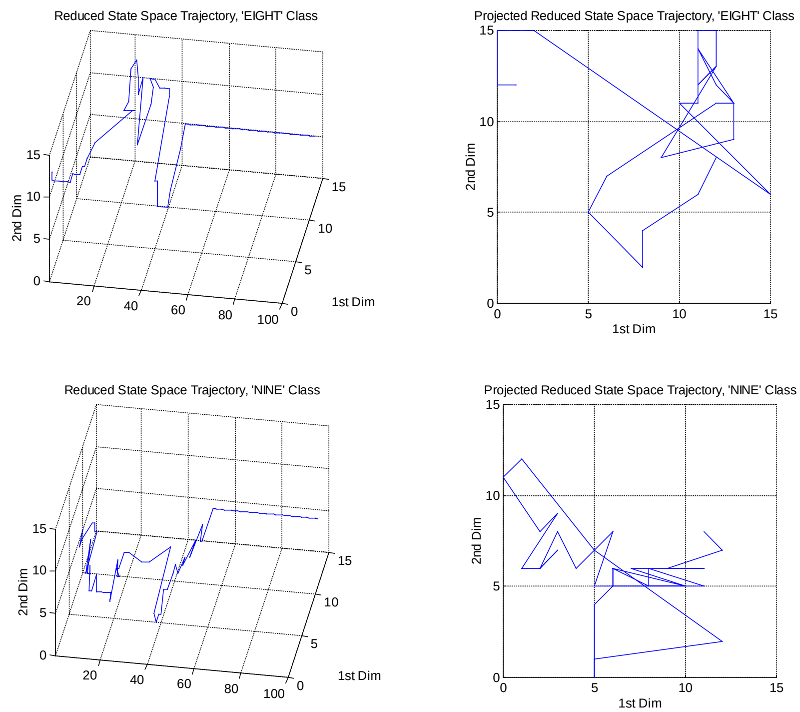

Figure 3-6. Reduced feature space trajectories, all digits. (Continuation)

3.4 Comments

As it was defined before, the Feature Extractor block works as a transducer that translates the information contained by an air pressure wave into a trajectory in some feature space.

Figure 3-7 sums up the processing done by the FE block. The different steps follow one another according to the following sequence:

1. The incoming pressure wave is reduced to a signal through a sampling process.
2. The starting and the ending points of the utterance embedded into the signal are obtained using the energy of the signal.
3. The spectrum of the extracted utterance is enhanced by means of a preemphasis filter which boosts the high frequency components.
4. Several data blocks are extracted from the enhanced signal.
5. The extracted data blocks are windowed to reduce leakage effects.
6. LPC components are extracted from the filtered blocks.
7. LPC Cepstrum components are then extracted from the LPC vectors.
8. The dimensionality of the LPC Cepstrum vectors is reduced using a SOM.
9. The resulting vectors are scaled if the Recognizer that is being used requires it.

Nothing can still be said about the overall effectiveness of the FE block, since it depends on the recognition accuracy of the overall system. As an example, if the complete system achieves low recognition percentages, that can be caused by the FE block or the Recognizer, but, if it achieves higher percentages that means that the FE block was at least able to produce data that allowed these recognition accuracies. In other words, in order to know the usefulness of the FE block, the Recognizer outputs must be obtained first.

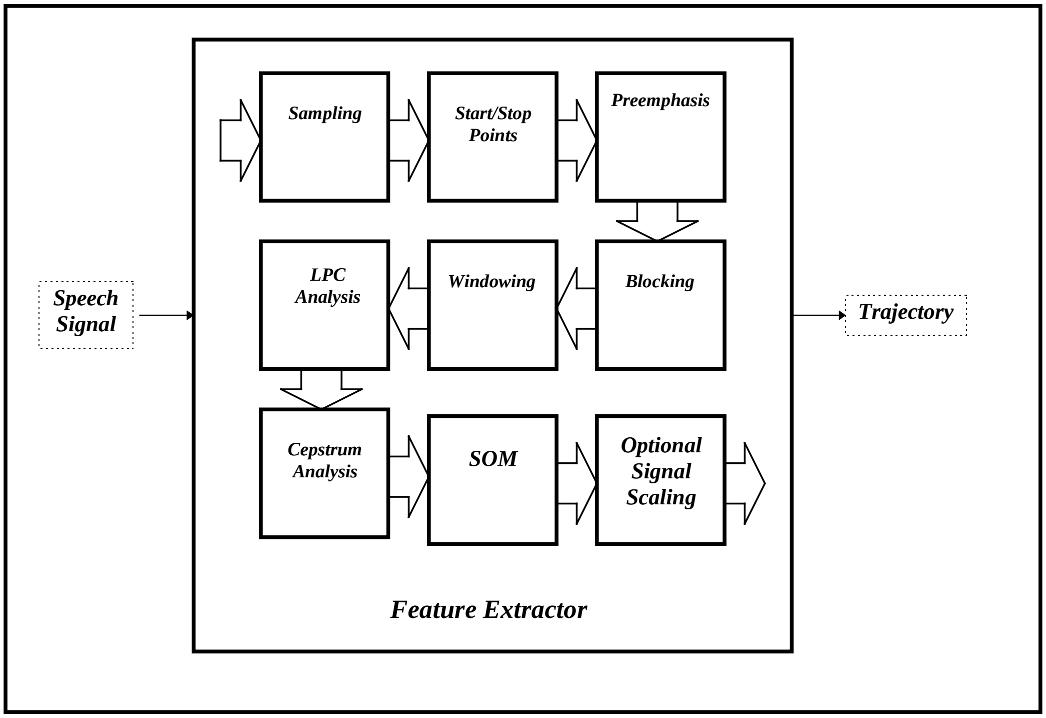

Figure 3-7. FE block schematic.

4. DESIGN OF A RECOGNIZER

The output of the FE block is "blind," *i.e.* it does not care about the word that is being represented by that trajectory. The FE block only transduces an incoming pressure wave into a trajectory in some feature space. It is the Recognizer block that discovers the relationships between the trajectories and the classes to which they belong, and adds the notion of class of utterance, *i.e.* type of word, into the system.

The Recognizer behaves like an observer. It fuses all the pieces of information provided by the FE block and produces an estimate. If this estimate belongs to a metric space, the Recognizer is called an estimator. On the other hand, if the estimate belongs to a set where the metric is ignored, or without a metric at all, the Recognizer is called a classifier. Since the output of a speech recognizer must belong to a vocabulary, a set of labels which does not have a metric associated to it, the speech recognizer must act like a classifier.

As mentioned before, it is not the purpose of this work to develop a full scale speech recognizer, but to test new techniques. Considering this goal, all the approaches developed in this chapter were tested on a simple problem: recognizing only one digit among the others.

All the experiments reported in this chapter used a training set comprised of 200 utterances, 20 for each digit, and a testing set composed of 100 utterances, 10 for each digit, different from the ones included in the training set. All the digits were uttered by the author.

The following sections describe two distinct treatments of the trajectories prior to feeding them into the Recognizer. The first translates each trajectory into a point that belongs to a higher dimensionality feature space. In this way, the trajectory recognition problem is converted into a Fixed Point recognition one, which explains why it is called the Fixed Point approach. The other treatment, which is called the Trajectory approach, directly works with the trajectories in time and analyzes them as they evolve.

4.1 Fixed Point Approach

This approach translates the time varying trajectory into a point in a space, reducing the trajectory classification problem into a point classification one.

Since the utterances exhibited a great deal of variation, not only in the feature space but also across time, they were normalized such that all time warping was canceled. All the trajectories in the training and the testing sets were normalized into trajectories of 100 points such that the Euclidean distance between adjacent points in the reduced feature space was always one hundredth of the length of the trajectory projected into the reduced feature space. This way, all possible time warpings were neutralized, leaving only the reduced feature space distortions [30] (see Figure 4-1, Figure 4-2, and Figure 4-3).

It is important to note that time warpings are not something that necessarily requires to be corrected. On the contrary, time warpings can convey important information which can be used to improve the accuracy of the recognition process.

Nevertheless, in order to simplify the task, the normalization approach was preferred to directly working with time warped trajectories.

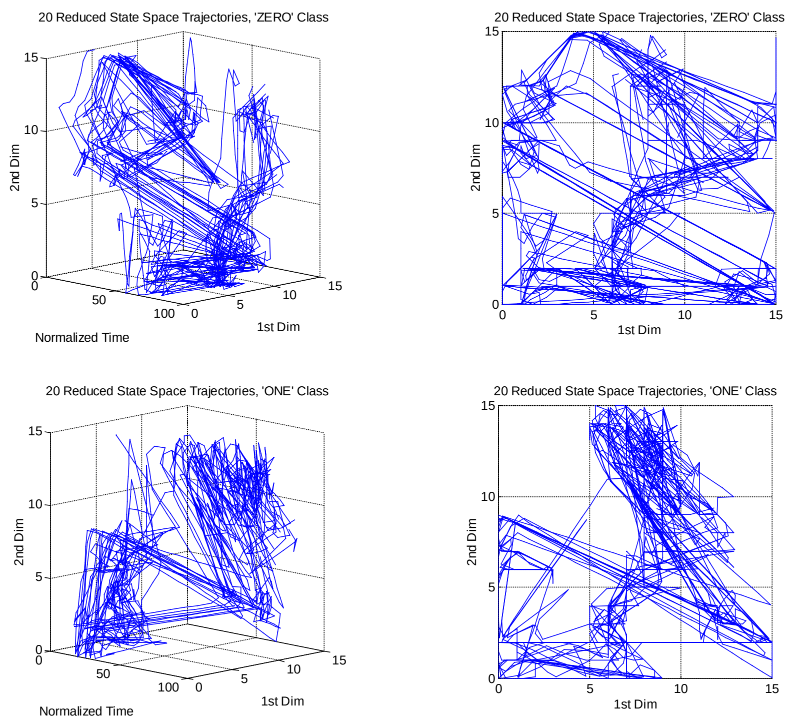

Figure 4-1. Normalized reduced feature space trajectories, 20 examples for each digit.

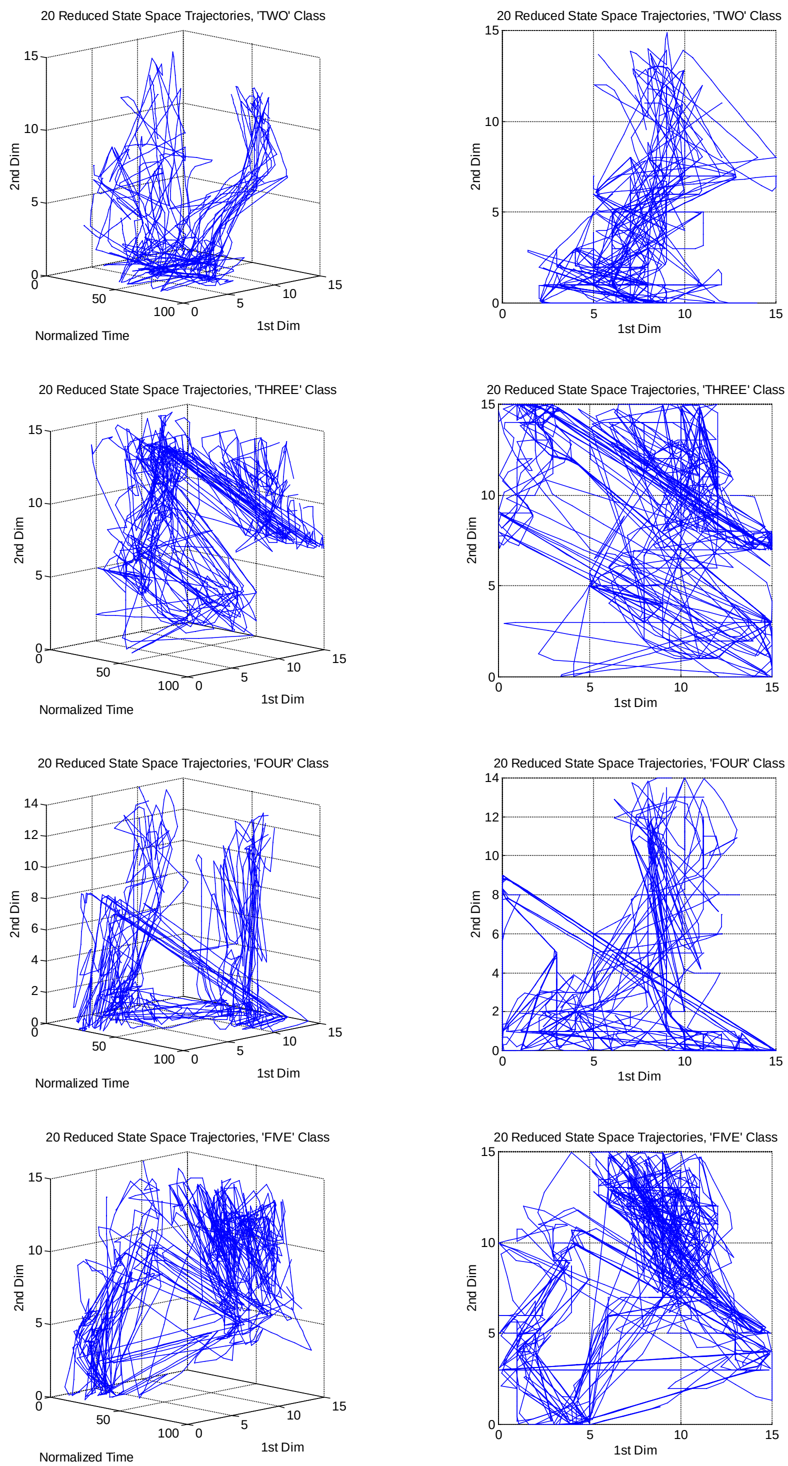

Figure 4-2. Normalized reduced feature space trajectories, 20 examples for each digit.

(continuation)

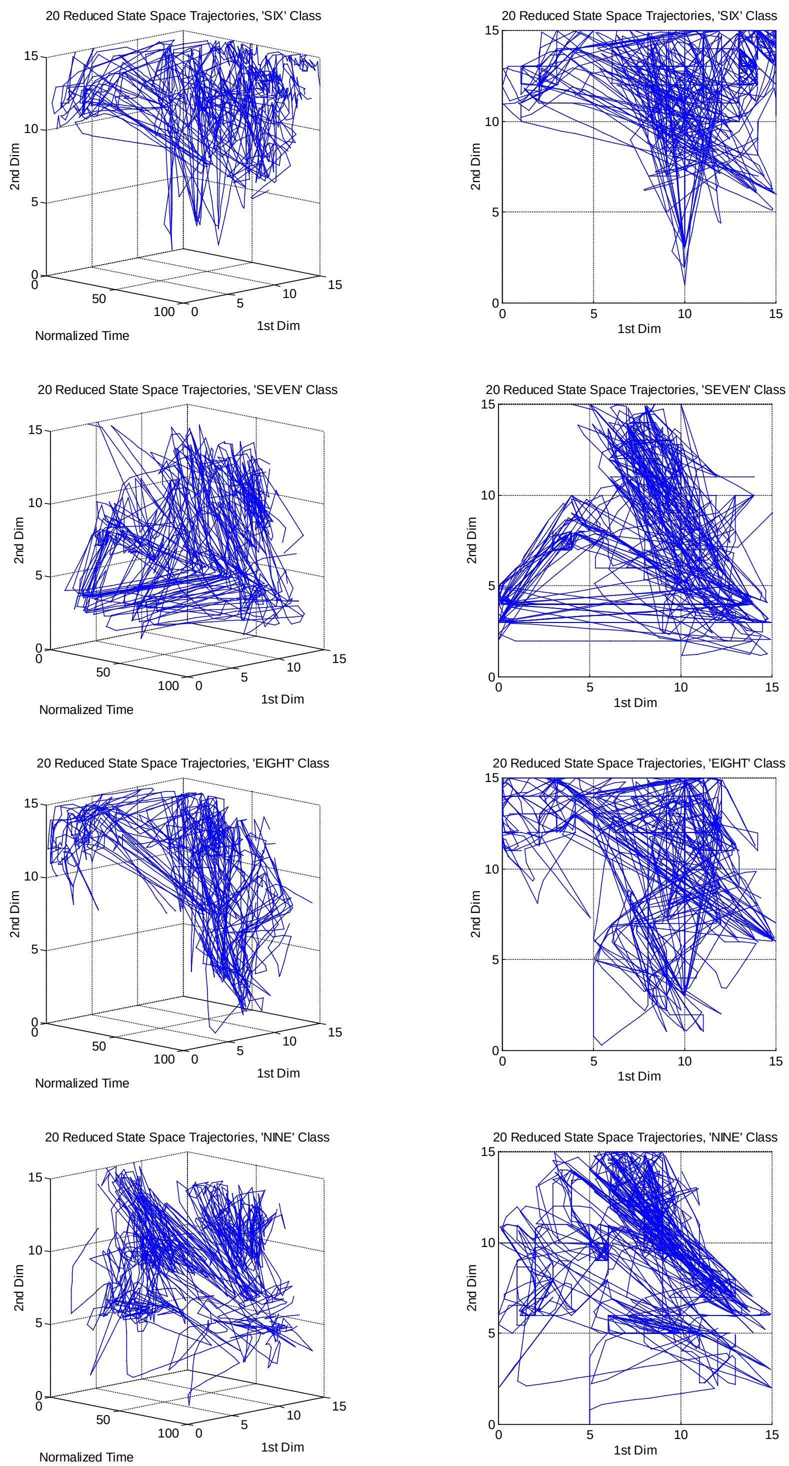

Figure 4-3. Normalized reduced feature space trajectories, 20 examples for each digit.

(continuation)

Then, each normalized trajectory was transformed into a Fixed Point in a higher dimensionality space. This was done by interpreting all the components of the vectors that defined a normalized trajectory, *viz.* 100 points, each of which belonged to a space of two dimensions, as a component of a point in a space of 200 dimensions. The components of the 100 vectors were stacked one after the other according to a lexicographic ordering until one vector with 200 components was obtained.

4.1.1 Template Approach

This approach divided the recognizer into two modules: the Template, a set that contained reference utterances of each of the digits, and a Comparator that compared an incoming trajectory against those stored in the Template.

The training set was divided into two subsets, each with an equal number of examples for each digit. The first of these subsets was used to build the Template, and the second to calibrate the Comparator.

The Template stored all the trajectories that belonged to a certain digit class, from now on let us say the digit ‘ZERO’, which were contained by the first of the smaller training subsets. Being all of them valid instances of that utterance, they were simply copied into the Template. This way, the intrinsic variability of the class was captured without the need for further data processing or system refinements.

The other module, the Comparator, worked as follows:

1. The mean Euclidean distance between the incoming trajectory and each of the trajectories stored in the Template, in this case 10 trajectories belonging to the class

‘ZERO’, was calculated. The expression that measures the mean Euclidean distance between the unknown trajectory and the i^{th} trajectory contained by the Template was defined by:

$$D_i = \sqrt{\frac{1}{100}\sum_{k=1}^{100}\left(t_i(k) - t_{unknown}(k)\right)^2} \tag{4.1}$$

2. Then, if $D_{\min} = \min D_i$, $\forall i \in [1,10]$, was below the T_{ZERO} threshold, it was said that the unknown trajectory belonged to the class ‘ZERO.’ If not, it was said that the unknown trajectory did not belong to the class ‘ZERO’.

To calculate T_{ZERO}, each of the ‘ZERO’ utterances contained in the second smaller subset obtained from the training set was compared against those stored in the Template. The ‘ZERO’ threshold for the digit ‘ZERO’ was defined as $T_{ZERO} = \max D_i$, $\forall i \in [1,10]$. This value of the ZERO’ threshold assured that at least all the ‘ZERO’ utterances contained in the training set were correctly identified.

4.1.1.1 Template Experiment

The procedure specified above was applied to all the digits. As expected, all the trajectories belonging to the training set were correctly classified. The results for the trajectories contained by the testing set are presented in Table 4-1.

Digit	**# of Errors**	**Recognition Accuracy**
0	11	89%
1	33	67%
2	22	78%
3	60	40%
4	15	85%
5	81	19%
6	22	78%
7	52	48%
8	11	89%
9	76	24%

Table 4-1. Testing set results for Template approach.

From Table 4-1 it can be appreciated that there are important variations between the results. While the recognizers for the digits ‘ZERO’ and ‘EIGHT’ achieved an 89% of accuracy, the recognizer for digit ‘FIVE’ only achieved 19%. This behavior can be explained by the fact that the recognition accuracy depends on the training examples stored in the template. If they do not capture the true variability of the utterance, the results will be poor. This could explain the poor results obtained for the ‘FIVE’ digit.

4.1.2 The Multi Layer Perceptron

In contrast with the previous approach, this approach employs a Recognizer composed of a MLP which simultaneously obtained the template and elaborated a way to compare the incoming points against the template. In an MLP, the final template and comparison method are encoded in distributed manner across the entire set of weights of the network.

The MLP had 200 input nodes, 5 hidden neurons, and 1 output neuron. The output of the neurons was inside the interval [-1,+1]. The hyperbolic tangent was used as the threshold function. Each neuron had an extra connection, whose input was kept constant and equal to one (the literature usually refers to this connection as bias or threshold). The weights were initialized with random values selected within the interval [-0.1,+0.1].

The MLP was trained using an approach that optimizes the usage of the training patterns and the Error backpropagation method. Details of this procedure are detailed in the following sections.

4.1.2.1 A New Learning Approach

Since the procedure outlined here is applicable to any general learning problem, the discussion in this section will be in a more general context than what is required for the specific problem of speech recognition.

A learning problem can be defined by the following assertions:

- The problem of interest is described by a set of infinite events.

- The events are indexed with the letter k.
- Each event is described by a pair of vectors, $(\vec{x}_k, \vec{y}_k)$, where $\vec{x}_k$ and $\vec{y}_k$ may belong to different metric spaces.
- All $\vec{x}_k$ and $\vec{y}_k$ are confined within limited regions of their respective metric spaces.
- All the data pairs are independent, *i.e.* $(\vec{x}_k, \vec{y}_k)$ is independent of $(\vec{x}_l, \vec{y}_l)$ for every $k \neq l$.
- All the data pairs are identically distributed.
- There is a system, dubbed the Learning Block (LB), which observes the events one at a time according to the order established by the indexing.
- The LB is a system composed of an arrangement of basic systems. The architecture of LB defines the way in which the basic systems are connected. Each of the basic systems is defined by an algorithm. This basic system is uniquely defined by an algorithm and a set of parameters. Summing up, an LB is completely described by an architecture, a set of algorithms, and a set of parameters. It will be assumed that the correct architecture and necessary basic algorithms has already been discovered through an evolutive process or set by design decisions. The only thing that is not defined about the LB is its set of parameters.
- One objective of the LB is to self configure its parameters using the observed data pairs such that when observing the k_*^{th} one for the first time, for every $\vec{x}_k$, $\forall k \geq k_*$, it

is able to generate an estimate $\hat{\vec{y}}_k$ of $\vec{y}_k$ such that $d_y\left(\vec{y}_k, \hat{\vec{y}}_k\right) < \varepsilon$. $d_y\left(\vec{y}_k, \hat{\vec{y}}_k\right)$ is a distance measure defined for the space that contains the $\vec{y}$ vectors.

- Another objective of the LB is to achieve the previous goal using as few observed data pairs as possible.
- Another objective of the LB is to achieve the previous goals with the smallest possible index k_*.
- After the LB observes an event $\left(\vec{x}_k, \vec{y}_k\right)$, before observing another event, it undergoes a training stage in which the parameters are adjusted in order to achieve the previously stated goals. Execution of training stage k is instantaneous but it incurs a cost C_k. The fourth objective of the LB is to achieve the previous goals while minimizing the total cost $C = \sum_{k=0}^{k_*} C_k$.

The traditional solution for this type of learning problems is defined by the following procedure:

1. LB does nothing until k_s events are observed.
2. At that moment, a training set $S_{training}$ is formed using m past event pairs. The function of $S_{training}$ is to store those data pairs selected to train the LB. Also, n past event pairs are selected to form a testing set $S_{testing}$. The function of $S_{testing}$ is to store those data pairs selected to test the LB. The selections are done such that $S_{training} \cap S_{testing} = \varnothing$ and $(m+n) \le k_s$.
3. The parameters of the LB are randomly initialized.

4. Before the event $k_s + 1$ is observed, the LB does its best to adapt its parameters using all the data pairs stored in $S_{training}$ in order to achieve its goals. The degree of fulfillment of the function approximation goal is measured using $S_{testing}$. After this training stage, the LB is never modified again.

The problem with this approach is that it is based on a chain of arbitrary decisions that are not optimized towards achieving the goals of LB. First, on what basis is the index k_s determined? The answer is none. The same applies to m and n. Furthermore, even if these numbers are optimal, who assures that the selected event pairs contain the desired information? In other words, there are no defined criteria referring to when the training has to start, or how many events, or which ones, should be included in the training and the testing sets.

This approach is equivalent to a teacher who has to teach calculus to a student, and whose teaching procedure consists of spending an undetermined amount of time to randomly gather information about the subject without ever opening the books that describe the topic. Once he arbitrarily decides that enough material has been collected, he hands all the gathered information over to the student. To make things worse, once the student receives the books, he studies them all, going through all the material, even if the material is repeated. It is easy to realize that there is a great inefficiency in this common training procedure. There must be a different approach that allows one to achieve the same goals in a more efficient manner. The next paragraphs will outline a new solution that seems to be better.

The new approach is based on the following fundamental assumption:

- If the LB, for every $\vec{x}_k$, $k < l$, produces an output $\hat{\vec{y}}_k$ such that $d_y\left(\vec{y}_k, \hat{\vec{y}}_k\right) < \varepsilon$, then, for some pairs $\left(\vec{x}_m, \vec{y}_m\right)$, $m \geq l$, such that $d_x\left(\vec{x}_k, \vec{x}_m\right) < \delta$, the relation $d_y\left(\vec{y}_m, \hat{\vec{y}}_m\right) < \varepsilon$ will be true. In other words, each time the LB learns the correct relation between a pair of vectors, it is learning to react to other vectors too. $d_x\left(\vec{x}_k, \vec{x}_m\right)$ and $d_y\left(\vec{y}_k, \hat{\vec{y}}_k\right)$ are distance measures defined for the spaces that contain the x and y vectors respectively.

The new learning procedure, based on an idea presented by the author in [12] is as follows:

1. The Controller is defined as a system that regulates the order in which the data pairs are observed by the LB.
2. Before training starts, the parameters of the LB are randomly initialized. Also, an empty training set $S_{training}$ is assigned to the LB. The function of $S_{training}$ is to store those data pairs selected to train the LB.
3. The Controller starts showing data pairs to the LB commencing $k = 0$ event pair.
4. When the LB observes the k^{th} pair, $k \in [0, \infty]$, if $d_y\left(\vec{y}_k, \hat{\vec{y}}_k\right) < \varepsilon$, the system ignores that pair, does not undergo a training stage, and waits until the Controller shows the next data pair. On the contrary, if $d_y\left(\vec{y}_k, \hat{\vec{y}}_k\right) \geq \varepsilon$, the pair is added to $S_{training}$ and the LB undergoes a training stage which uses the parameters found in the previous training stage as a starting point and $S_{training}$. If the system underwent a training stage, the

learning procedure returns to the previous state, where the Controller starts showing data pairs to the LB from index $k = 0$.

This approach is like a teacher that teaches the material lesson by lesson, such that when he realizes that the student already knows a new lesson, the teacher moves on to teach another.

According to the Fundamental Assumption, each learned pair implies that the system has learned to correctly react to a neighborhood of examples close to that pair. Hence, it becomes unnecessary to use all the observed data pairs. Only those data pairs outside those neighborhoods, which contain information unknown to the LB, should be used. With the selection of training examples in this manner, the training set is kept quite small throughout the training process.

Furthermore, after a pair is learned, the probability of finding a pair with new information reduces because the unlearned region diminishes. The expected number of redundant examples before finding a useful one is the expected number of trials $E[k] = \frac{(1-p)}{p}$ in a geometric distribution [31], where p is the probability of finding a useful example. In other words, as the system learns more and more, the probability of finding a pair that contains information unknown for the LB decreases.

In the parameter space, the search for the correct parameters is also affected by the new scheme. The relationship implied in each data pair can be satisfied by any parameter vector from a specific set in the parameter space. Each data pair implies a different specific set. The global solution, the solution that satisfies all the data pairs, is any of the parameter vectors located at the intersection of the different specific sets defined by all the data pairs. The traditional approach tries to find any of the global solutions located inside this overall intersection directly. The new approach produces a different strategy. Since the training set is grown incrementally, and at the end of each training stage the LB has a parameter vector located at the intersection of the various specific sets defined by the data pairs contained by the training set at that instant, the search for the correct parameters is guided and gradually converges to one of the global solutions. Each time the LB starts a training stage it uses the previously found parameters, which are guaranteed to be closer to the solution than most of the randomly chosen points. The overall effect of this procedure is that the big parameter corrections are likely to be executed when the training set is small, while the small ones are done when the training set is bigger, keeping the computational load of the training process small.

Nothing is said about a stopping rule for the algorithm because as the LB learns more, there are fewer training stages. Under the assumption that the LB can achieve zero recognition error, it will eventually learn to respond adequately to all the possible cases and it will never need to undergo a training stage again. In other words, the training process automatically shuts off when it learns to correctly react to all the cases.

Due to the same reasons, there is no need for a testing set. Once the system learns to correctly react to all cases, it will achieve a perfect score with any possible $S_{testing}$.

In summary, the new approach uses an optimally small training set and the computational load is kept small throughout the complete procedure. Finally, only the best case of the traditional approach can be equivalent to the worst case of the new approach, which makes the new approach the preferred one among the two.

4.1.2.2 Multi Layer Perceptron Experiment

The procedure presented above was applied to the recognition of all the digits. As expected, all the trajectories belonging to the training set were correctly classified. The results for the trajectories contained within the testing set are presented in Table 4-2.

Only 200 speech data pairs were available for training the MLP, which forced the training to stop once all the training utterances were used.

An epoch was defined as the period in which the MLP was trained with a fixed training set until all the errors produced by the data pairs in the training set were below a threshold of 0.5. Every epoch comprised a variable number of backpropagation iterations. The number of iterations within an epoch was variable because the system iterated until all the errors were below the threshold of 0.5, or until the number of iterations reached 100,000.

Once trained, if the output of the MLP was positive, the incoming trajectory was classified, as an example, as a ‘ZERO’. If it was negative, the trajectory was regarded as belonging, as an example, to the class ‘NON-ZERO’.

Digit	# of Errors	Recognition Accuracy
0	2	98%
1	9	91%
2	10	90%
3	4	96%
4	3	97%
5	13	87%
6	9	91%
7	8	92%
8	7	93%
9	12	88%

Table 4-2. Testing set results for MLP approach.

As a consequence of the training approach, as the epochs progressed the MLP learned. As it learned it was more rare to find a data pair which caused an erroneous output. Figure 4-4 shows the evolution in the number of ‘ZERO’ examples examined as the training set grew. It is important to note that since each time a new example was added to the training set a new epoch started, the size of the training set is the same as the epoch number. Figure 4-4 shows that the MLP generated the correct outputs for the entire training set after being trained with only 32 training examples. It also shows that in the first few epochs the MLP was completely ignorant and used almost all the examined training examples. As the MLP learned, it became more and more difficult to find unknown information, which explains the sort of exponential behavior of the curve.

Figure 4-4. Training examples use evolution.

Figure 4-5 shows that the percentage of computational resources is kept small, even as the training set grew. The curve shows that some epochs required more iterations in order to converge to the desired weight values. This behavior is expected, since for a given weight configuration it might be more difficult to find the optimal parameter combination given a training set.

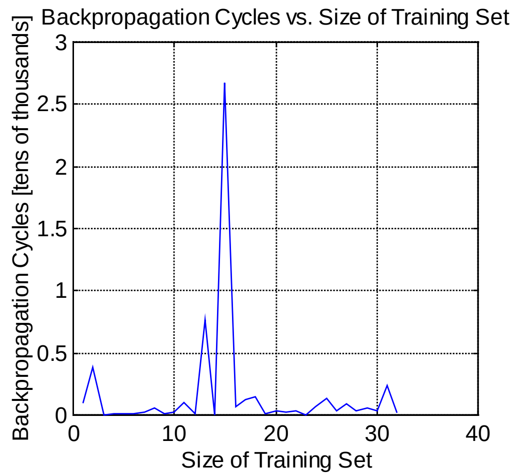

Figure 4-5. Percentage of computational resources as a function of the actual training set.

4.2 Trajectory Approach

This approach, rather than reducing the problem into a point classification one, deals directly with the normalized trajectories. The trajectories are directly fed into the recognizer, which, at the same time, is continuously producing an output that can be used for classification purposes.

4.2.1 Recurrent Neural Network Trained with Back-Propagation Through Time

Contrary to the MLP approach, which waits until a trajectory ends in order to process it, this approach works with the trajectories as they appear. It does not transform a trajectory into a point, but preserves its time characteristics. The only consideration though, is that it works with normalized trajectories.

As the MLP approach, an RNN constitutes a single, monolithic, block that simultaneously stores the template and discovers the proper comparison method, such that it is impossible to differentiate one from the other.

The RNN architecture was defined by a set of fully interconnected neurons, which are either input, hidden, or output neurons. The dynamics of the RNN was defined by [32]:

$$\tau_i \dot{x}_i = -x_i + \sum_j w_{ij} y_j + z_i \tag{4.2}$$

where τ_i is a time constant that regulates the speed at which the value of the states change, x_i is the state of neuron i, w_{ij} is the weight that goes from neuron j to neuron i, y_j is the output of neuron j, and z_i is the input to the neuron i. The state of a neuron and its output are related by the hyperbolic tangent function:

$$y_i = \frac{2}{1+e^{-x_i}} - 1 \tag{4.3}$$

which is one type of bipolar functions. z_i is zero for every neuron that does not have an input. In other words, it is zero for every hidden and output neuron.

The RNN was trained with the backpropagation Through Time (BTT) algorithm [33]. The equations that specify the implementation of this algorithm are obtained by minimizing the value of a functional along the normalized trajectory [10]:

$$J = \int_{t_0}^{t_1} L dt \tag{4.4}$$

where t_0 marks the starting time of the normalized trajectory, t_1 its ending time, and L the functional to be minimized. The functional is a Lagrangian defined by the Kullback-Leibler Divergence cost function and the dynamics of the system:

$$L = \sum_i \left\{ \left[\frac{1+t_i}{2} \log \frac{1+t_i}{1+y_i} + \frac{1-t_i}{2} \log \frac{1-t_i}{1-y_i} \right] + \lambda_i \left[\tau_i \dot{x}_i + x_i - \sum_j w_{ij} y_j - z_i \right] \right\} \tag{4.5}$$

where, $i \in [1, N]$ denotes all the neurons, t_i is the desired output for neuron i, and λ_i is the i^{th} Lagrangian coefficient. The desired output of the i^{th} neuron, t_i, is defined as follows:

$$t_i = \begin{cases} d_i & ; \quad \text{if the } i^{th} \text{ neuron is an output one} \\ y_i & ; \quad \text{otherwise} \end{cases} \tag{4.6}$$

In all the previous equations, $\dot{x}_i$, x_i, y_i, z_i, t_i, and λ_i are functions of time. On the other hand, τ_i and w_{ij} are not functions of time.

The training of the network proceeds from minimizing J using the RNN dynamics [3.2] as a constraint. The training method is obtained using the Euler-Lagrange equations from variational calculus:

$$\begin{aligned} \frac{\partial L}{\partial x_i} - \frac{d}{dt} \left\{ \frac{\partial L}{\partial \dot{x}_i} \right\} &= 0 \\ \frac{\partial L}{\partial \lambda_i} &= 0 \end{aligned} \tag{4.7}$$

along with the border conditions

$$\left. \frac{\partial L}{\partial \lambda_i} \right|_{t=t_1} = 0 \tag{4.8}$$

The result of this approach is the following set of continuous-time equations:

$$
\begin{aligned}
&\tau_k \dot{\lambda}_k = \lambda_k + \frac{1}{2}\left(y_k - t_k\right) + \frac{1}{2}\left(y_k^2 - 1\right)\sum_l \lambda_l w_{lk} \\
&\lambda_k\Big|_{t=t_1} = 0
\end{aligned}
\tag{4.9}
$$

Using a gradient descent method to obtain the analytical expressions that dictate the adjustment of weights and auxiliary variables:

$$
\begin{aligned}
\Delta w_{ij} &= -\eta \frac{\partial J}{\partial w_{ij}} = \eta \int_{t_0}^{t_1} \lambda_i y_j \, dt \\
\Delta \tau_j &= -\eta \frac{\partial J}{\partial \tau_j} = -\eta \int_{t_0}^{t_1} \lambda_j \dot{x}_j \, dt
\end{aligned}
\tag{4.10}
$$

Using the approximation:

$$
\frac{df(t)}{dt} \approx \frac{f(t+\Delta t) - f(t)}{\Delta t} \tag{4.11}
$$

to obtain discrete expressions, it follows that the dynamics of the RNN can be rewritten as:

$$
x_i(k+1) = x_i(k) + \sum_j w_{ij} y_j(k) + z_i(k) \tag{4.12}
$$

and the dynamics of the backpropagating signal can be rewritten as:

$$
\begin{aligned}
&\lambda_i(k_1) = 0 \\
&\lambda_i(k-1) = -\frac{1}{2}\left(y_i(k) - t_i(k)\right) - \frac{1}{2}\left(y_i^2(k) - 1\right)\sum_j \lambda_j(k) w_{ji}
\end{aligned}
\tag{4.13}
$$

and the weights modified by:

$$
\Delta w_{ij} = \eta \sum_{k=k_o}^{k_1} \lambda_i(k) y_j(k) \tag{4.14}
$$

where $\tau_i = 1$, $\forall i$, and $\Delta t = 1$.

The RNN that was used in this work had two input neurons, two hidden neurons, and 1 output neuron, totaling 5 fully interconnected neurons.

The value of d_i, the desired signal, was set to 0.95 if the corresponding input trajectory belonged to the class that the RNN had to recognize, or -0.95 if the corresponding input trajectory did not belong to the class that the RNN had to recognize.

Using the training method used for the MLP and the BTT, an RNN was trained to recognize the word ‘ZERO’. The RNN only converged when at most 3 training examples, 1 positive and two negatives, were used. It was not possible to make the weights converge when a bigger number of training trajectories was used. When more examples were used, the training was so slow, that the method seemed completely impractical when compared to the previous ones.

Although the RNN did not use the complete training set, it was applied to the testing set and a recognition accuracy of 78.80% was achieved. Plots of the outputs of the RNN for one positive and two negative trajectories are shown in Figure 4-6. The figure shows that the positive trajectory’s output settles around 0.95 after some milliseconds, while the negative trajectory’s outputs settle around -0.95 after a similar amount of time.

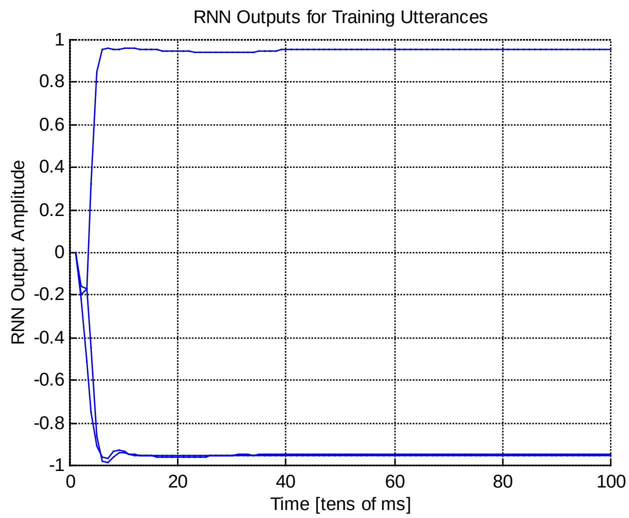

Figure 4-6. RNN output for training utterances.

In all the tests, the initial RNN states were set to 0, and the RNN initial weight values were randomly chosen in the interval [-0.1,0.1].

4.3 Discussion on the Various Approaches Used for Recognition

Before speaking about the different recognition schemes it is important to note the variability exhibited by the normalized trajectories in the reduced feature space. This might be a natural characteristic of the utterances or a shortcoming of the FE block. If it is the former, it is remarkable in the sense that when the associated utterances are heard they are regarded as very similar, if not identical, despite their variability in the feature space. If it is the latter, it suggests that the FE block must be improved.

In general, all the three approaches solved the scalability problem in a trivial way: to add a new word is just matter of adding an additional Template, MLP, or RNN.

The following comments concerning the Template approach are in order:

1. The major drawback of the method is its low recognition accuracy. The only way to improve its efficiency seems to be to use a higher number of normalized trajectories in the Tamplate, which inevitably would increase the memory requirements of the implementation.
2. Another important drawback of the approach was its memory requirements. Since each point of a trajectory corresponds to a point in the 16 by 16 neurons lattice of the SOM, each point can be stored in one byte, *i.e.* a 100 points trajectory requires 100 bytes. Since each digit had a Template defined by 10 trajectories, each digit required 1,000 bytes of memory, without using any data compression techniques. If we were working with a vocabulary of 50,000 words, and if each template had 10 trajectories in order to achieve high recognition accuracy, 50,000,000 bytes of memory would be required to store all the Templates. This is by no means a light, in terms of memory, system.
3. One advantage of the Template Approach is that it quickly allowed to capture some of the variability of a class of trajectories without any special computational requirements.

Some comments concerning the MLP approach can also be made:

1. Its recognition accuracies were far better than the ones obtained with the Template Approach. Even though its performance was better, it is still below the limits required for practical applications.
2. If the problem with the Template Approach was that it was memory thirsty, the MLP approach was even worse. The hidden layer was defined by 5 hidden neurons, each

with 200 weights and 1 bias, totaling 1005 floating point values. The output layer consisted of only 1 neuron, with 5 weights and 1 bias, totaling 6 floating point values. In total, each MLP required 1011 floating point values. Hence, in order to build a system with a 50,000 words capacity would require 202,200,000 bytes, assuming 4 bytes per each floating point value. Even though its recognition accuracy was far better when compared against the one of the Template approach, its memory requirements pose some limitations for practical implementations.

3. In terms of training, the MLP approach was definitely more complex than the one based on the Template approach. Although the incrementally growing training set approach considerably eased the training, it still required training times in the order of minutes to hours to train the MLP recognizers.

A few comments concerning the RNN can also be made in the light of the items used for comparison above:

1. It achieved only 80% of recognition accuracy, which sets this method as the worst of the three approaches. Nonetheless, this result is only preliminary in the sense that the system was only applied for recognizing the digit 'ZERO' and trained using only three training examples.
2. In terms of memory requirement, it is the best. The fully connected RNN with 5 hidden neurons requires only 25 floating point values, which implies that a system with a 50,000 words capacity would require 5,000,000 bytes, assuming 4 bytes per each floating point value.

3. It was by far the most complex system, so complex that it was not possible to train better RNNs by means of bigger training sets. This was the reason that caused the abandoning of this approach and that explains why only results for the 'ZERO' digit were obtained.

Concerning the new approach for training a learning system:

1. Within the scope of the conducted experiments, the obtained results validate the usefulness of the new learning approach. It achieved successful results using only a fraction of the training sets.
2. There is a strong need for more general and conclusive experiments. There is need for a thorough comparison between the traditional approach and this new method. Experiments with different estimation and classification problems using different data sets, different noise conditions, different learning blocks, different data orderings, and different initial conditions must be conducted before more general statements can be made concerning the new learning procedure. Additionally, all of the assumptions on which the new approach rests should be revisited and the method generalized even more according to them. This specially refers to the possibility of gradually diminishing the function approximation error ε towards the final desired value as the training set is augmented. This may speed up the training of the LB in the new approach.
3. If the method proves to be robust enough, it may be useful to measure how good a the LB block is within the context of a problem. So far, learning procedures are only compared on the basis of the number of components they are made of, and the

computational power that they require to solve a given problem. Another important measure would be the amount of training examples required to solve the problem. A powerful LB should require less training examples than a less powerful one. Such a measure would signify the amount of exposure a LB needs to a problem in order to solve it.

4. It may also be worthwhile to apply this method to learning problems in sequential analysis and adaptive control. Both of these fields obtain estimates using the data as it is observed in some adaptive manner. It would be interesting to see if the approach presented in this work can be used to modify those techniques in some useful manner.

5. CONCLUSIONS

5.1 Summary of Results Reported in This Thesis

Although none of the approaches proved to be good enough for practical purposes with the present extent of development, they were good enough to prove that translating speech into trajectories in a feature space works for recognition purposes. The human speech is an inherently dynamical process that can be properly described as a trajectory in a certain feature space. Even more, the dimensionality reduction scheme proved to reduce the dimensionality while preserving some of the original topology of the trajectories, *i.e.* it preserved enough information to allow a good recognition accuracy.

It is interesting to note that despite the fact that the SOM has been used in the speech recognition field for more than a decade, nobody has used it to produce trajectories, but only to generate sequences of labels.

Finally, the new approach developed for training the neural network's architecture proved to be simple and very efficient. It reduced considerably the amount of calculations needed for finding the correct set of parameters. If the traditional approach had been used instead, the amount of calculations would have been higher.

5.2 Directions for Future Research

Concerning future work, besides improving the FE block and devising a more robust Recognizer, the scope of the problem should be broadened to larger vocabularies, continuous speech, and different speakers. From this perspective, the results presented in this thesis are only preliminary.

As it could be appreciated, the reduced feature space trajectories were very noisy, which complicated the recognition process. This can be a natural consequence of the speech signal, or an artifact caused by the feature extraction scheme. If the latter were the case, it would not be surprising since the LPC Cepstrum coder scheme is not very efficient when it comes to represent noise-like sounds as the consonant ‘S’ [7]. It may be more appropriate to use feature extractors that do not throw away information before the dimensionality reduction scheme is used. As an example, the output of a Fourier or Wavelet transform, both of which retain all the information needed to reconstruct the original signal, could be directly used as input to the SOM.

With respect to the dimensionality reduction, as the vocabulary size grows, the reduced feature space will start to crowd with trajectories. It is important to study how this crowding effect affects the recognition accuracy when reduced space trajectories are used.

New approaches must be developed in order to avoid using the knowledge of starting and ending points, plus the need of trajectory normalization. There is a strong need for trajectory recognizers that are able to process highly distorted, both in space and time, unknown trajectories in a sequential manner.

Even though the Trajectory Approach did not produce encouraging results due to the training complexity of the RNN, and that it seems less efficient when compared with the Fixed Point approach, it does not have to be discarded because it should be possible to combine the lesser memory requirements of the Trajectory Approach with the simplicity of the Fixed Point Approach. In this context, it must also be taken into account that only one gradient descent method was used to train the RNN. It might be easier to train the RNN with other gradient descent based methods, or radically different techniques like reinforced learning [34], simulated annealing [17], or genetic algorithms [35]. Furthermore, the architecture of the RNN can be modified such that instead of training one RNN for recognising a long and complex trajectory, it may be easier to train a hierarchy of simpler RNNs, each designed to recognize portions of the long and complex trajectory. Such an approach will go along the lines of the HMM scheme, which organizes the HMMs in hierarchical layers.

So far, all the approaches that are used in speech recognition require a significant amount of examples for each class. In a thousands-of-words vocabulary problem this would require that the user of the system uttered hundreds of thousands of examples in order to train the system. New approaches must be developed such that the information acquired by one module can be used to train other modules, i.e. that use previously learned information to deduce the trajectories that correspond to non-uttered words.

6. REFERENCES

[1] S. Pinker, “The Language Instinct,” HarperPerennial, 1995.

[2] L. Rabiner and G. Juang, “Fundamentals of Speech Recognition,” Prentice-Hall, 1993.

[3] C. H. Prator and B. W. Robinett, “Manual of American English Pronunciation,” Harcourt, Brace and Co., 1985.

[4] N. Negroponte, “Being Digital,” Vintage Books, 1995.

[5] C. R. Jankowski Jr., H. H. Vo, and R. P. Lippmann, “A Comparison of Signal Processing Front Ends for Automatic Word Recognition,” IEEE Transactions on Speech and Audio processing, vol. 3, no. 4, July 1995.

[6] J. Tebelskis, “Speech Recognition Using Neural Networks,” PhD Dissertation, Carnegie Mellon University, 1995.

[7] S. Furui, “Digital Speech Processing, Synthesis and Recognition,” Marcel Dekker Inc., 1989.

[8] K. Torkkola and M. Kokkonen, “Using the Topology-Preserving Properties of SOMs in Speech Recognition,” Proceedings of the IEEE ICASSP, 1991.

[9] K-F Lee, H-W Hon, and R. Reddy, “An Overview of the SPHINX Speech Recognition System,” IEEE Transactions on Acoustic, Speech, and Signal Processing, vol. 38, no. 1, January 1990.

[10] H. Hasegawa, M. Inazumi, "Speech Recognition by Dynamic Recurrent Neural Networks," Proceedings of 1993 International Joint Conference on Neural Networks.

[11] M. Jamshidi, "Large-Scale Systems: Modelling and Control," North-Holland, 1983.

[12] P. Zegers, "Reconocimiento de Voz Utilizando Redes Neuronales," Engineer Thesis, Pontificia Universidad Católica de Chile, 1992.

[13] M. Woszczyna et al, "JANUS 93: Towards Spontaneous Speech Translation," IEEE Proceedings Conference on Neural Networks, 1994.

[14] T. Zeppenfeld and A. Waibel, "A Hybrid Neural Network, Dynamic Programming Word Spotter," IEEE Proceedings ICASSP, 1992.

[15] Y. Gong, "Stochastic Trajectory Modeling and Sentence Searching for Continuous Speech Recognition," IEEE Transactions on Speech and Audio Processing, vol. 5, no. 1, January 1997.

[16] D. Richard, C. Miall, and G. Mitchison, "The Computing Neuron," Addison-Wesley, 1989.

[17] S. Haykin, "Neural Networks: A Comprehensive Foundation," Macmillan College Publishing Company, 1994.

[18] T. Kohonen, "Self-Organization and Associative Memory," Springer-Verlag, 1984.

[19] T. Kohonen, "Self-Organizing Maps," Springer-Verlag, 1995.

[20] T. Kohonen et al, “Engineering Applications of the Self-Organizing Map,” Proceedings of the IEEE, 1996.

[21] H. Sagan, “Space-Filling Curves,” Springer-Verlag, 1994.

[22] G. Cybenko, “Approximation by Supperpositions of a Sigmoidal Function,” Mathematics of Control, Signals and Systems, vol. 2, 1989.

[23] K. Funahashi, “On the Approximate Realization of Continuous Mappings by Neural Networks,” Neural Networks, vol. 2, 1989.

[24] K-I Funahashi and Y. Nakamura, “Approximation of Dynamical Systems by Continuous Time Recurrent Neural Networks,” Neural Networks, vol. 6, 1993.

[25] T. Kohonen, “The Neural Phonetic Typewriter,” Computer, vol. 21, no. 3, 1988.

[26] K. J. Lang and A. H. Waibel, “A Time-Delay Neural network Architecture for Isolated Word Recognition,” Neural networks, vol. 3, 1990.

[27] E. Singer and R. P. Lippmann, “A Speech Recognizer using Radial Basis Function Neural Networks in an HMM Framework,” IEEE Proceedings of the ICASSP, 1992.

[28] H. Hild and A. Waibel, “Multi-Speaker/Speaker-Independent Architectures for the Multi-State Time Delay Neural Network,” IEEE Proceedings of the ICNN, 1993.

[29] R. M. Gray, “Vector Quantization,” IEEE ASSP Magazine, April 1984.

[30] G. Z. Sun et al, “Time Warping Recurrent Neural Networks,” IEEE Proceedings of the ICNN, 1992.

[31] A. Papoulis, “Probability, Random Variables, and Stochastic Processes,” McGraw-Hill, 1991.

[32] S. I. Sudharsanan and M. K. Sundareshan, “Supervised Training of Dynamical Neural Networks for Associative Memory Design and Identification of Nonlinear Maps,” International Journal of Neural Systems, vol. 5, no. 3, September 1994.

[33] B. A. Pearlmutter, “Gradient Calculations for Dynamic Recurrent Neural Networks,” IEEE Transactions on Neural Networks, vol. 6, no. 5, September 1995.

[34] M. K. Sundareshan and T. A. Condarcure, “Recurrent Neural-Network Training by a learning Automaton Approach for Trajectory Learning and Control System Design,” IEEE Transactions on Neural Networks, vol. 9, no. 3, May 1998.

[35] D. B. Fogel, “An Introduction to Simulated Evolutionary Optimization,” IEEE Transactions on Neural Networks, vol. 5, no. 1, January 1994.