INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

A

An Automatic Speech Recognition Oriented Study On Segmentation, Low Dimensional Feature Extraction, And Temporal Trajectory Information Capture

By

YONGGANG ZHU

A dissertation submitted to the Graduate Faculty in Physics in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2002

This manuscript has been read and accepted for the Graduate Faculty in Physics in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Sept 19, 2002
Date

Robert V. Fanelli
Professor Robert V. Fanelli, Brooklyn College
Chair of Examining Committee

Sept 19, 2002
Date

Sultan Catto
Professor Sultan Catto, The Graduate Center
Executive Officer

Michael Anshel
Professor Michael Anshel, City College

John Antrobus
Professor John Antrobus, City College

Bo Lawergren
Professor Bo Lawergren, Hunter College

Kenneth Miyano
Professor Kenneth Miyano, Brooklyn College

THE CITY UNIVERSITY OF NEW YORK

Abstract

An Automatic Speech Recognition Oriented Study On Segmentation, Low Dimensional Feature Extraction, And Temporal Trajectory Information Capture

By

Yonggang Zhu

Adviser: Professor Robert V. Fanelli

Accurate and efficient automatic speech recognition requires feature vectors highly discriminative for the categories of interest while at a low dimensionality. Recent studies on feature extractions from mel spectra show that classical mel-frequency cepstral coefficients (MFCCs) may not be able to capture some important cues existing in the local spectral correlates. Thus, we study feature extraction together with dimensionality reduction on mel spectra using the hybrid models of neural networks and Euclidean distance proposed by us. This is mainly inspired by the adaptive nature of neural networks. If we use classical MFCCs as a benchmark, features extracted by our hybrid models can give comparable or much better classification rates while with significant dimensionality reduction. Time warping recurrent neural network, aimed to recognize phonemes and CV syllables by efficiently capturing temporal trajectory information, is studied with mel features, MFCCs and our features, and the results suggest that low dimensional features extracted by linear Euclidean neural networks may be better for this purpose.

Acknowledgements

I would like to thank Professor Robert V. Fanelli for his supervision to finish this dissertation, thank my family for their supports and tolerances on the time and energy consuming Ph.D. studies, and thank my friends, Zohn Rosen, Ou Xu, Xiaoping Yang, Yuemei Dong, Chengwen Pei, Jin Mao, Linfeng Liu, and many others, for the helps they give me.

An Automatic Speech Recognition Oriented Study On Segmentation, Low Dimensional Feature Extraction, And Temporal Trajectory Information Capture

Yonggang Zhu
Department of Physics
Brooklyn College of CUNY

CONTENT

List of Figures

List of Tables

I. Introduction

Researches in automatic speech recognition (ASR) have been carried out for about four decades. In these four decades, speech recognition technologies have evolved so that research emphases have shifted from acoustic phonetic and pattern recognition approaches to statistical modeling methods since those new technologies have turned in much more successful results. Today, it is already possible for an ASR system to follow human voice commands, understand some simple human utterances, or even answer back [1,2,9,10]. Figure 1 schematically shows basic structure of an ASR system.

In this structure, speech waveform signals are input into the ASR system in a sequential manner. After certain preprocessing, speech spectra are processed with selected spectral analysis techniques. The obtained outputs, or so called feature measurements, are usually described as feature vectors with desired low dimensionality. Feature vectors are then passed to ASR classifier or classification system for recognition decisions. Although there are different approaches for speech recognition, our studies in this dissertation fall into the category of statistical pattern recognition approach.

To start a speech recognition study, the first thing one needs is the data preparation. Speech data are usually composed of spoken words or sentences. In most cases, people focus their studies on certain aspects and, thus, they do not need all of the data. However, to automatically select the portion of interest from the rest is often difficult. In our work, two databases are used. One is the widely used TIMIT corpus and the other one is the database created in Professor John Antrobus' lab. The TIMIT corpus

has all segmentation details, but we need to find a way to get similar information from the other database. Although there are many methods proposed for automatic audio data segmentation, none of them looks suitable for our purposes [11,12]. To solve this problem, in Section II, we describe in details about our practical segmentation methods, and we also show the satisfactory results obtained.

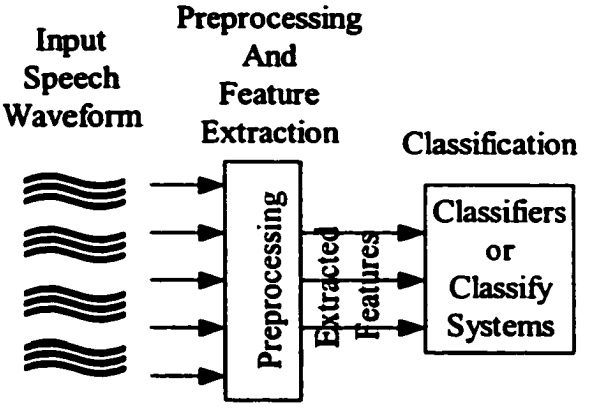


Figure 1: The basic schematic structure of automatic speech recognition systems

Generally, before an ASR system can be put to use, its classification component must be trained with certain features. These features are extracted in some way from speech signals and there are many types of such features. To be selected for training, a feature set not only must be highly discriminative with respect to the categories of interest and produce good classification performance (percentage of correct classifications), but also should have lower dimensionality to reduce computation cost since large or huge amount of training may be required. The most popular feature set used in the last twenty years is formed by mel-frequency cepstral coefficients (MFCCs) [13,14]. MFCCs are calculated by discrete Fourier transform on the filtered logarithmic spectra of speech

signals in mel scale, which is actually a kind of warping scale on speech frequencies to represent how human ears hear sound (see Appendix A). In Section III, we follow this classical method and calculate MFCCs for our speech data. However, recent studies have shown that by using features such as multi resolution cepstral features one can get better classification performance than MFCCs in phoneme recognition [15, 16]. These sub-band based researches are primarily inspired by Allen's paper [17] and are mainly based on the conjecture that important additional cues for phonetic discrimination may exist in local spectral correlates that are not captured by full band MFCCs. In another word, MFCCs are not able to extract some discriminative information from filtered mel spectra and how features can be most efficiently extracted from filtered mel spectra is unknown and is still under investigation.

Neural networks (NNs), or artificial neural networks, are parallel computational models comprised of densely interconnected adaptive processing units. A very important feature of these NNs is their adaptive nature, where "learning by examples" replaces "programming" in solving problems [3,4]. This nature makes NNs very useful when one has no complete understanding about the problems available but has input and output data sets readily available. In our work, we propose hybrid models that combine neural networks (NNs) with the concept of Euclidean distance in Euclidean classifiers (EUC). These hybrid models can extract low dimensional features from mel spectra with the neural network part and keep different categories separated through the Euclidean distance part. In Section IV and V, we make extensive studies on these models. From our studies, we show that features extracted by our models contain enough discriminative

information for higher or the same classification performance as obtained from MFCCs while with significantly lower dimensionality.

For the same phoneme in the same word, different speakers may speak at different speed, and thus phonemes of different length may be generated. This is so called time warping problem, which is especially critical to temporal pattern based recognition. Although Hidden Markov Model (HMM) has been proved very successful in sequence pattern recognition, HMM can not deal with time warping problem directly [1,18]. So, we decide to use time warping recurrent neural network to capture spectral trajectory information for better speech recognition. In Section VI, we report the studies we have, and the results are interesting and encouraging.

In Section VII, we summarize our studies. To make our text compact, we also move all mathematical derivations into Appendix.

II. Speech Signal Segmentation

The speech database that is created and collected in Professor Antrobus' lab is used in our early studies. Among all the phonemes contained in the database, we focus our studies on the following four target vowels, /IY/ as in "feet", /IH/ as in "sit", /EH/ as in "set", /OO/ as in "foot", and two target consonants, /f/ and /s/. Speech data collection is performed with 16kHz sampling rate and 16-bit quantization in mono-channel on a personal computer with Sound Blaster audio card and SUN standard vocal microphone. Speeches are recorded in a soundproof room. All these data are saved in Microsoft audio wave files. As a sample, the top plot in Figure 2 (a) shows waveform of such an audio wave file. Each wave file contains 10 isolated token words, and there are total 8 different groups of such token words (see Appendix B). Each word is a single syllable word ended with /t/, i.e. in /CVt/ format, where C stands for a consonant and V stands for a vowel. There is always a silence between vowel and its following consonant /t/ [19]. Only the words composed by our target consonants and target vowels are our target token words.

As we can see from Appendix B, our target token words are scattered in each token word group. For example, of the 10 words spoken in m13a.wav audio file, our target token words are "fet", "soot", "fit" and "seet". Then, the immediate concern is how to segment these target token words out from the rest. Or, more specifically for this speech signal segmentation problem, how to determine the start and end positions of the consonant and vowel in each target token word in selected audio files.

A. Some Basic Phoneme Properties

In order to have some basic understandings on the characteristics of our target token phonemes and target token words, we first manually segment some of our audio files using audio wave editing software. This manual segmentation is mainly based on the physical waveform contours and structures. For unvoiced consonants /f/ and /s/, their start positions are determined by noticeable amplitude changes from preceding silence while their end positions are determined more easily by following voiced vowels. For vowels, to better observe their periodic structures, three regions are defined and labeled as the follows.

(1) Initial region: The portion where amplitude has already significantly increased, but no steady periodic structure has been formed yet.

(2) Steady region: The portion where periodic structure has been fully formed and is steady.

(3) Finish region: The portion where periodic structure can no longer be held and the amplitude decreases significantly.

The statistical segmentation results are listed in Table 1. They show that our target vowels generally have average lengths in the range from 150ms (male spoken /IH/) to 201ms (female spoken /EE/) with about 6-7% initial region and 13-16% finish region. Meanwhile, our target consonants have average lengths in the range from 203ms (female spoken /f/) to 212ms (male spoken /f/).

In many studies, these segmented phonemes or words are not processed as a whole unit. Instead, their waveforms are usually divided into a sequence of fixed-size sections with some overlap between them in time domain. Each of the elementary fixed-

Table 1

Average Lengths (in Millisecond) of Target Phonemes From A Portion of Our Audio Wave Data. The Corresponding Standard Deviations Are inside The Parentheses.

Phoneme		Average Lengths of Phonemes and Standard Deviations			
		Initial Region	Steady Region	Finish Region	Total
/EE/	Female	10.71 (1.4)	167.49 (36.4)	23.06 (12.6)	201.26 (42.2)
	Male	10.79 (2.1)	126.56 (37.6)	23.67 (5.7)	161.01 (36.9)
	Overall	10.75 (1.7)	147.02 (41.7)	23.37 (9.5)	181.13 (43.8)
/IH/	Female	10.92 (2.4)	120.51 (30.9)	22.39 (7.0)	153.82 (31.2)
	Male	11.06 (1.9)	113.33 (45.6)	25.81 (10.5)	150.19 (41.9)
	Overall	10.99 (2.1)	116.92 (38.1)	24.10 (8.8)	152.00 (36.0)
/EH/	Female	12.01 (3.7)	151.17 (29.1)	19.93 (9.2)	183.11 (32.2)
	Male	13.63 (3.4)	129.8 (37.7)	24.8 (5.4)	168.13 (38.0)
	Overall	12.82 (3.5)	140.47 (34.7)	22.33 (7.8)	175.62 (35.1)
/OO/	Female	11.75 (2.6)	151.19 (54.0)	23.98 (10.5)	186.92 (54.4)
	Male	10.98 (4.5)	126.69 (52.8)	21.79 (6.5)	159.46 (54.1)
	Overall	11.36 (3.6)	138.94 (53.5)	22.88 (8.6)	173.19 (54.7)
/FF/	Female	-	-	-	202.94 (58.1)
	Male	-	-	-	212.04 (41.7)
	Overall	-	-	-	207.37 (50.4)
/SS/	Female	-	-	-	209.95 (54.2)
	Male	-	-	-	212.38 (26.0)
	Overall	-	-	-	211.13 (42.3)

Notes:

(1): The target phoneme samples used for this statistics are from the following target token words: feet, fit, fet, foot, seet, sit, set, and soot.

(2): The target token words used for this statistics are from the following audio wave files: m7a.wav, m7d.wav, m7f.wav, m10a.wav, m10d.wav, m10f.wav, m13a.wav, m13d.wav, m13f.wav, m17a.wav, m17d.wav, m17f.wav, m18a.wav, m18d.wav, m18f.wav, f10a.wav, f10d.wav, f10f.wav, f13a.wav, f13d.wav, f13f.wav, f17a.wav, f17d.wav, f17f.wav, f57a.wav, f57d.wav, f57f.wav, f58a.wav, f58d.wav, and f58f.wav.

(3): Of the audio file names, the first letter stands for the gender of the speaker, the number in the middle is the label number of a speaker, and the last letter stands for the token word group being spoken in this audio file.

size sections may be regarded as a sub-pattern and is called a frame. Therefore, research on a phoneme or a word is mainly made with its frames.

B. Segmentation of Target Token Words And Phonemes

Many methods have been proposed for automatic speech segmentation based on different features of speech. Some utilize reference templates, and some incorporate linguistic knowledge [11,12]. However, none of those proposed methods are universal. Rather, they are specific for certain types of signal processing, such as speech analysis, synthesis, or coding, and are usually complicated. So it is generally not easy to borrow such kinds of methods directly and efficiently. In our database, all speech data have clear predefined format. All the token words have /CVt/ pattern and are isolated from each other. And we know that there are exactly 10 such words in each audio file. Therefore, segmentation on our speech signals should not be so complicated. Thus, we decide to develop our own methods for segmentation.

Generally speaking, vowels have well defined periodic waveform structures, high amplitudes, and are long in duration. From our early stage experiments on determining fundamental periods of vowels, we find that our vowel samples do show very good periodic properties. Although there are many algorithms to find fundamental period T_0, Betancourt and Antrobus use the following function to determine T_0 with a 20ms window frame [21],

$$s(k) = \frac{\Sigma_{j=1}^{N-k}(f_{j+k} - f_j)^2}{2\Sigma_{j=1}^{N-k}(f_{j+k}^2 + f_j^2)} \qquad \text{(II-1)}$$

Fundamental period is then determined by the k value at which s(k) gives minimum value, where f is waveform signal and N is window size. They show that accurate period values can be achieved by using (II-1) together with some interpolations. When reviewing this B-A method, we first consecutively take 20ms frames from each selected word with certain overlap, and then we apply (II-1) on each of them. We find that s(k) values drop quickly as soon as steady periodic waveform signals are encountered and that s(k) values increase quickly as soon as steady periodic waveform signals vanish. This property thus can be used to primarily determine a vowel in given waveform signals.

However, from our experiments we also find that although B-A method is very good to detect the start and end positions for steady periodic regions of vowels, it is not so accurate in pinning down the start and end positions of entire vowels. For example, the plus signs in Figure 1-b shows the periodic region of /EH/ determined by B-A method, which has about 27.8ms offset from the real start position and about 28.9ms offset from the real end position. It is not surprising if we notice that there exist acoustic transitions in both initial and finish regions of vowels. In these transition regions, periodic structures are either not fully formed yet or about to vanish. Moreover, since window frames are taken in 20ms duration one after another, it is possible for a frame around transition region to contain both periodic signals from a vowel and non-periodic signals from a consonant. This mixing will also contribute to the failure in obtaining accurate positions. The first reason is intrinsic, while effects from the second one may be reduced somewhat by using smaller frames and larger overlaps.

To improve the accuracy in determining start positions of vowels, we propose a secondary method based on the fact that a vowel has much higher amplitude than its

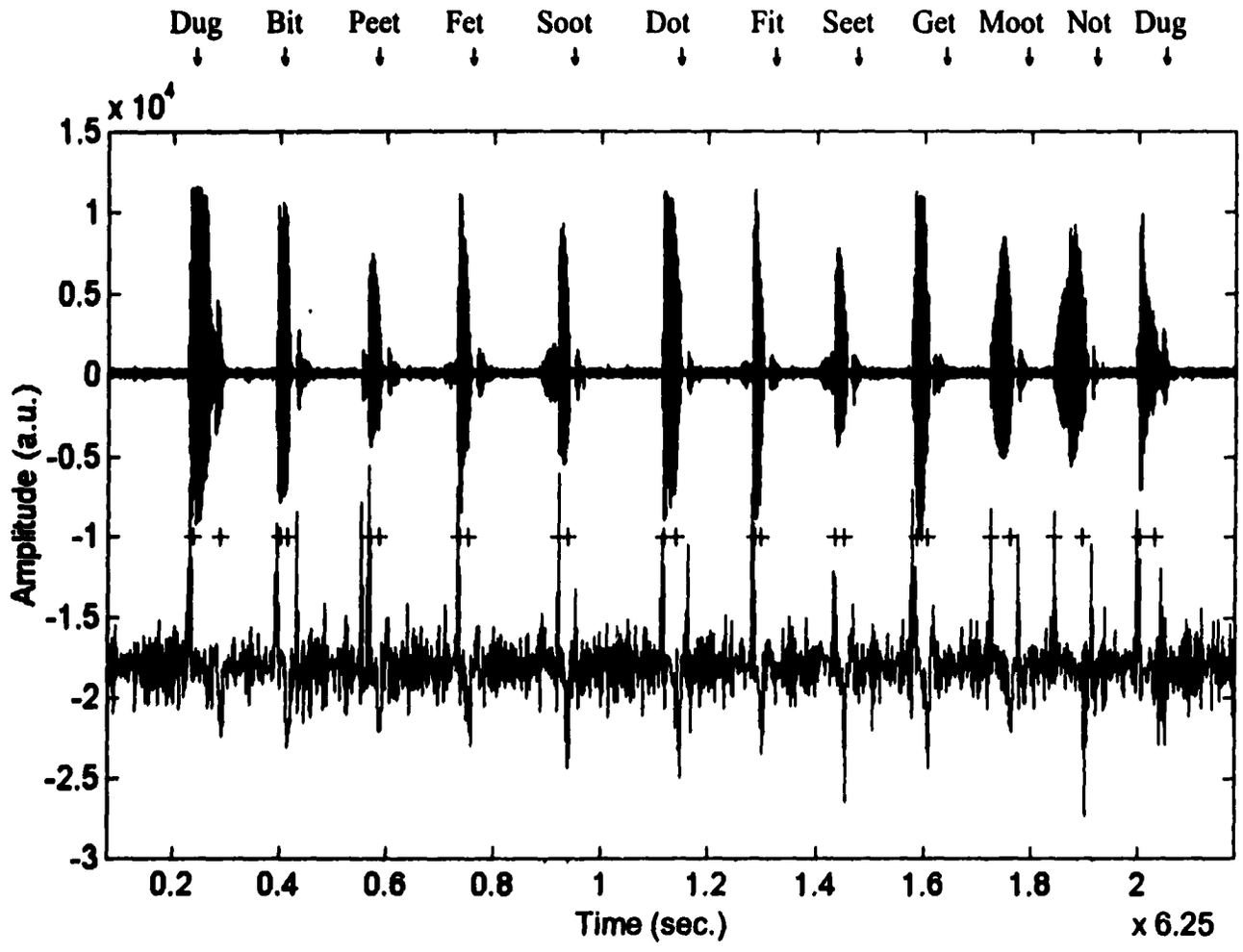
previous neighbor consonant /s/ or /f/. In this method, we first calculate amplitude mean square (MS) values for each consecutively taken 10ms waveform frame, then build up an array to store ln(MS) differences between two adjacent frames using the formula $\ln MSDiff(i)=\ln(MS_i)-\ln(MS_{i-1})$. Any significant change in two adjacent frames will indicate a possible transition from one phoneme to another. Therefore, at the vicinity of start position found by B-A method, we can further determine the more accurate start position for a given vowel by finding the greatest value of lnMSDiff(i). This start position can also be regarded as the end position of the vowel's preceding consonant.

For the determination of start positions of consonants and the more accurate end positions of vowels, we make use of the fact that there are silences in the region before consonant C and the region between vowel V and /t/ in each /CVt/ word. Of course, as one may know, these silences are actually the background noises. We create a silence spectrum template together with its standard deviations by sampling certain amount of 10ms silence frames. A frame will be a silence frame if more than 70% its data are within standard deviations when comparing with the silence spectrum template. Therefore, if we compare frames with the template in a temporally backward order starting from start position of a vowel, the start position of a consonant may be determined when a frame becomes a silence frame. On the other hand, if we compare frames with template in a temporally forward order starting from the end position of steady region of a vowel, which has been found by B-A method, the more accurate end position of a vowel may be determined when a frame becomes a silence frame. This straightforward silence spectrum template method turns out to be good enough for our segmentation here.

To test above segmentation methods, we arbitrarily select 3 male speakers and 3 female speakers. For each speaker, we select 3 token word groups they read, group a, d and f. There are 4, 2, and 2 target token words in group a, d and f, respectively (see Appendix B). Thus we have total 18 audio wave files, which give total 48 target token words. The notes in Table 2 gives more details on the selected data.

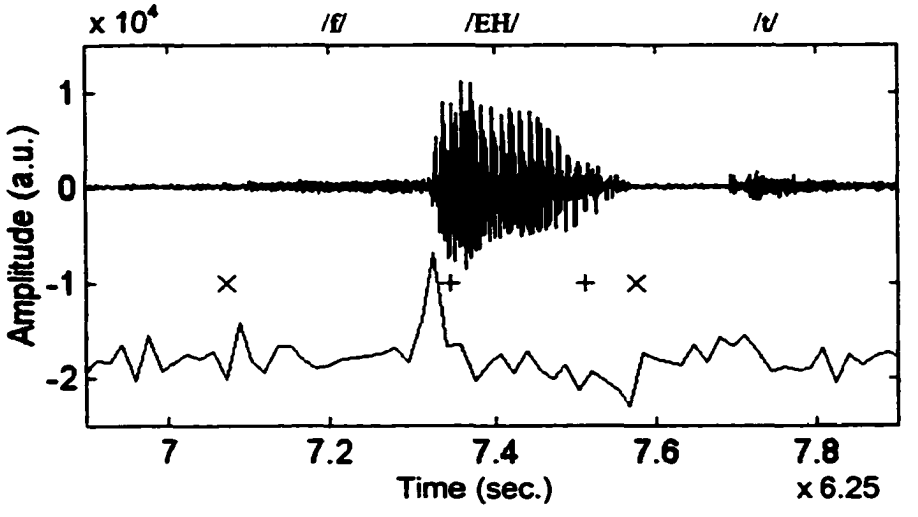
To illustrate our phoneme segmentation results, we first use m13a.wav audio file as a sample here. As shown in Figure 2 (a), the top plot is the waveform of audio file m13a.wav. The middle plus signs indicate the start and end positions of vowels determined by B-A method (B-A positions). The bottom plot is the spectrum of lnMSDiff(i) values for each pair of adjacent frames taken in m13a.wav. Its spikes are the positions where sharp ln(MS) value changes occur, which suggest transitions from one phoneme to another. As we can see from Figure 2 (a), these B-A positions are close to the corresponding lnMSDiff(i) spikes, and some of them almost overlap with each other (e.g. the start position of /OO/ in "soot"). Meanwhile, we also see offsets between them, for example, in words "fet" and "fit". Figure 2 (b) gives amplified picture from Figure 2 (a) for target token word "fet". We see that B-A positions are obviously off from correct positions, and, on the other hand, the lnMSDiff(i) spikes give much better determination on the start position of /EH/. Figure 2 (b) also gives the start position of /f/ and end position of /EH/ determined by silence spectrum template method, and the silence spectrum template method gives much better determination of the end position of /EH/.

Figure 2

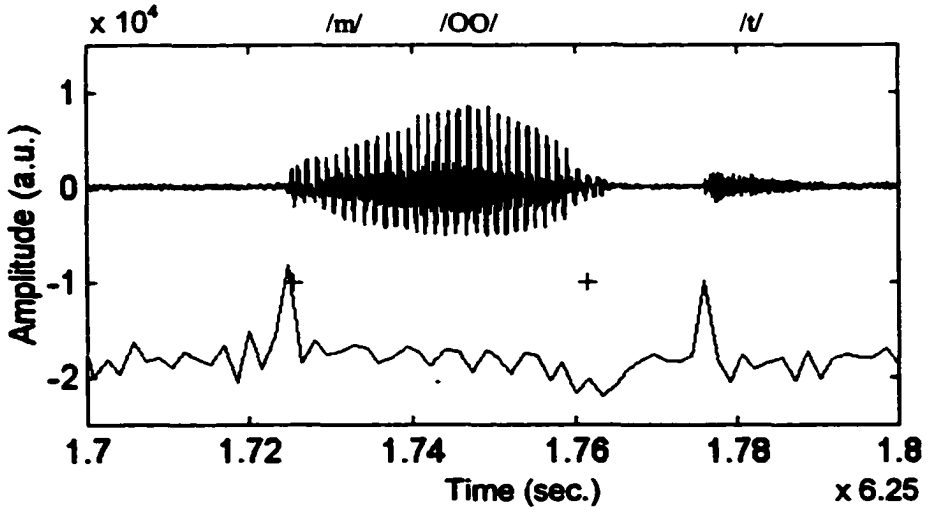


(a) The top plot is the speech waveform of audio file m13a.wav. The middle plus signs indicate the start and end positions of vowels determined by B-A method. The bottom plot is the spectrum of lnMSDiff(i) values.

Figure 2 (continued)



(b) Token word "fet" in m13a.wav. Here the extra x signs are the start position of consonant /f/ and end position of vowel /EH/ respectively, which are obtained by silence spectrum template method.



(c) The token word "moot" in m13a.wav. Both B-A method and logarithmic mean square difference method can not separate /m/ from /OO/.

Now, let's see the statistics of our segmentation results on all selected audio files. They are in Table 2. In this table, the values, Δpos, are the offsets of start or end positions of vowels from their correct positions, i.e.,

$$\Delta pos = pos_c - pos_{sm} \qquad \text{(II-2)}$$

where pos_c is the correct position determined manually with audio wave editing software and pos_{sm} is the position determined by our segmentation methods. The overall average error for start positions of all vowels is -12.20ms with standard deviation 6.6ms if we use B-A method only. However, if we also use logarithmic mean square differences after B-A method, this average error is reduced to -2.08ms with standard deviation 3.9ms. For the end positions of all vowels, if we apply silence spectrum template after B-A method, the overall average error is also dramatically reduced, from 21.34ms to 2.06ms. Considering that we usually use frames 20ms long, these resolutions thus can be accepted.

The above segmentation methods liberate us from heavy tedious work, such as manually selecting tens of thousands frames. Also, possible human bias in manual segmentation is eliminated. In the following sections, all waveform frames from this database are obtained by applying these methods with various desired parameter settings, such as window size, overlap length, etc., in our program.

At the end of this section, we would like to report that we also have tried to use above methods to segment vowels on non-target token words in those selected audio files. It turned out that all non-target token words are segmented approximately as good as in our target token words, except /mVt/ and /nVt/ words (see Figure 2 (a) and (c)). For

/mVt/ and /nVt/ words, we are not able to separate the consonant /m/, or /n/, from its following vowel. We believe the failure is because that, similar to vowels, nasals /m/ and

Table 2
Segmentation Results From 48 Target /CVt/ Token Words in 18 Selected Audio Wave Files (in Millisecond)

Vowels		Average Δpos of Vowels with Standard Deviation			
		V Initial Region		V Finish Region	
		B-A	ln MS Diff	B-A	S. S. T.
/EE/	female	-14.69 (5.4)	-3.56 (4.6)	13.31 (6.5)	12.75 (17.4)
	male	-11.41 (5.9)	-1.28 (3.7)	26.41 (7.8)	-2.40 (6.5)
	overall	-13.04 (5.7)	-2.42 (4.2)	19.86 (9.7)	5.18 (14.8)
/IH/	female	-10.66 (4.2)	-3.06 (5.6)	15.09 (7.1)	2.80 (5.3)
	male	-10.84 (2.9)	-2.00 (3.3)	25.63 (9.9)	1.60 (7.8)
	overall	-10.75 (3.4)	-2.53 (4.4)	20.36 (9.9)	2.20 (6.4)
/EH/	female	-14.38 (7.9)	-1.13 (2.9)	19.84 (3.9)	0.36 (4.4)
	male	-16.56 (11.5)	-2.31 (3.3)	30.02 (10.8)	-2.03 (3.2)
	overall	-15.47 (9.5)	-1.72 (3.0)	24.93 (9.4)	-0.83 (3.9)
/OO/	female	-10.25 (4.0)	-3.88 (3.5)	17.63 (9.8)	3.88 (9.6)
	male	-8.82 (6.9)	0.56 (3.7)	22.81 (11.0)	-0.47 (3.0)
	overall	-9.54 (5.5)	-1.66 (4.1)	20.22 (10.3)	1.70 (7.1)
All	female	-12.49 (5.0)	-2.91 (3.7)	16.47 (9.8)	4.95 (10.8)
	male	-10.50 (7.5)	-1.41 (3.4)	24.34 (9.6)	1.71 (5.4)
	overall	-12.20 (6.6)	-2.08 (3.9)	21.34 (9.7)	2.06 (9.0)

Notes:
(1): The 18 selected audio wave files are: m10a.wav, m10d.wav, m10f.wav, m13a.wav, m13d.wav, m13f.wav, m17a.wav, m17d.wav, m17f.wav, f10a.wav, f10d.wav, f10f.wav, f13a.wav, f13d.wav, f13f.wav, f17a.wav, f17d.wav, and f17f.wav.
(2): The target token words are: feet, fit, fet, foot, seet, sit, set, and soot.
(3): B-A stands for Betancourt and Antrobus period determination method.
(4): ln MS Diff stands for logarithmic mean square difference method.
(5): S. S. T. stands for silence spectrum template method.
(6): The unit for the Δpos data is ms (millisecond).
(7): The negative Δpos data means the determined position is offset to the right from its correct position in a left-to-right time axis.
(8): The positive Δpos data means the determined position is offset to the left from its correct position in a left-to-right time axis.

/n/ also have defined periodic waveform structures with high amplitudes. These characteristics of /m/ and /n/ make our assumptions to distinguish consonant from vowel in a single syllable word no longer valid.

III. Feature Extraction

A. mel Feature Vectors

We first take waveform window frames from target token phonemes at the desired positions determined by our segmentation methods described in last section. 12 speakers, 6 males and 6 females, are selected. For each speaker, we also select the same 3 token word groups, group a, d and f, which give total 96 target token words. With this selection, we will have equal number of samples for each token vowel and consonant respectively (see the notes in Table 3 and Appendix B). Waveform frames are taken with a length of 20ms. For each target token word, 4 frames are taken in vowel region at its start, 1/3, 2/3 and end positions respectively, 2 frames are taken in consonant region at its 1/3 and 2/3 positions respectively. Since silence is often encountered in ASR and is usually regarded as a null phoneme, we also take 1 frame from the silence region at the start position in each target token word. Therefore, besides the 6 target token phonemes, we will have one more null phoneme for silence, /SL/. Each target token phonemes will then form a phoneme category, which gives 7 categories. Total 672 sample frames are obtained with equal number of frames for each category and each gender.

Feature extraction procedure used by us in this section is basically as the follows [22]. Each frame obtained above is normalized to a root mean square value of 1.0 and passes Hamming window (see Appendix C). After that, we apply the discrete Fourier transform (DFT) to the output of windowing and take logarithm of the result power values. The logarithmic magnitude spectrum will be modified slightly with a floor at -

50dB value. A warped frequency scale, the mel scale, is applied by using a filter bank (see Appendix A). In the mel filter bank, center frequencies of these filters are spaced equally on a linear scale from 100 to 1000Hz. Above 1000Hz, each center frequency is 1.1 times the center frequency of the previous filter. The magnitude frequency response will have a triangular shape that is equal to unity at the center frequency of each filter and linearly decreasing to zero at the center frequencies of the two adjacent filters. Each vector of log magnitudes is then weighted by this mel filter bank and the set of filter bank outputs is called the mel feature vector.

B. MFCC Feature Vectors

As in the classical work of Davis *et al* [13], each mel vector will be further processed using the following function for the creation of a vector of mel-frequency cepstral coefficients (MFCCs),

$$MFCC_i = \sum_k^N X_k \cos[i(k-0.5)\frac{\pi}{N}] \qquad \text{(III-1)}$$

where i is MFCC index, N is the number of filters, X_k is the k^{th} element value of mel vector. This signal representation of DFT-derived MFCCs has been shown to outperform other representations and has been widely accepted as a standard. Although Davis *et al* use only the first 20 filters, in our work, 30 full filters are used with center frequencies ranging from 100Hz to 6727.5Hz since it has been shown that spectrum at high frequency also contributes to phoneme intelligibility [23].

C. Evaluation with Euclidean And Diagonal Bayesian Maximum Likelihood Classifiers

Table 3
Percentages of Correct Classifications for Selected Phoneme Samples
* The number of mel vector elements used in Davis *et al*'s work

Feature Vectors	EUC	D-BML
mel Feature Vectors (20)*	77.4	81.8
mel Feature Vectors (30)	81.1	82.4
MFCC Feature Vectors (1)	36.3	40.5
MFCC Feature Vectors (2)	58.8	63.1
MFCC Feature Vectors (3)	63.2	72
MFCC Feature Vectors (4)	67.7	74.9
MFCC Feature Vectors (5)	68.8	77.2
MFCC Feature Vectors (6)	73.4	82.4
MFCC Feature Vectors (7)	75.3	82.4
MFCC Feature Vectors (8)	75.4	82.9
MFCC Feature Vectors (9)	76.0	85.0
MFCC Feature Vectors (10)	76.4	85.7
MFCC Feature Vectors (12)	76.5	86.6
MFCC Feature Vectors (15)	76.3	86.3
MFCC Feature Vectors (30)	76.5	86.3

Notes:
The audio wave files used for this statistics include m3a.wav, m3d.wav, m3f.wav, m7a.wav, m7d.wav, m7f.wav, m10a.wav, m10d.wav, m10f.wav, m13a.wav, m13d.wav, m13f.wav, m17a.wav, m17d.wav, m17f.wav, m18a.wav, m18d.wav, m18f.wav, f4a.wav, f4d.wav, f4f.wav, f10a.wav, f10d.wav, f10f.wav, f13a.wav, f13d.wav, f13f.wav, f17a.wav, f17d.wav, f17f.wav, f57a.wav, f57d.wav, f57f.wav, f58a.wav, f58d.wav, and f58f.wav.

All feature vectors are then classified using Euclidean classifier (EUC) and diagonal Bayesian maximum likelihood classifier (D-BML) (see Appendix D). Here normal conditional density function is assumed. Since we have a relatively small number of sample feature vectors, the leave-one-out method is used to evaluate performance for these two classifiers (see Appendix E). With both mel vectors and MFCC vectors, classification rates, i.e. the percentages of correct classifications, are calculated. Statistical

results are listed in Table 3. The number inside the parenthesis of the first column, n, indicates the number of elements used for calculations, i.e., the first n feature vector elements are used for each trial. By doing so, we can evaluate the relationship between the classification rate and the number of MFCC features being used.

For mel feature, the maximum classification rate is 81.1% for EUC and 82.4% for D-BML with all 30 mel elements, respectively. For MFCC features, the maximum classification rate is 76.5% for EUC and 86.6% for D-BML with the first 12 elements, respectively. According to Davis *et al* work, most information relevant to speech recognition can be captured with about 6 MFCCs, although up to 10 MFCCs may be used for better performance. Our MFCC results agree with these basic conclusions when using both classifiers. For EUC, when n=6, the classification rate is about 95.9% of the maximum value. When n=10, the classification rate becomes 99.9% of the maximum value. For D-BML, when n=6, the classification rate is about 95.2% of the maximum value. When n=10, the classification rate becomes 99.0% of the maximum value. Table 3 also shows that MFCC features can give better classification rates than mel features, although they are actually calculated from mel data.

Although MFCC features have been widely used, recent studies have shown that better classification performance for phoneme recognition can be achieved if one uses a feature set called multi-resolution cepstral features [15] or combines MFCCs with features based on mel-based nonlinear discrete-time energy operator [24]. With MFCCs, people may use as few as 6 elements to capture most speech information for good classification performance. However, it does not necessarily mean that this is the minimum number of dimensions of a feature set one has to use in speech recognition.

Then, very naturally, a question will be asked: Is there the lowest boundary on the number of feature dimensions required to characterize a phoneme? Or, more practically for us, can we have smaller number of dimensions while keeping classification rates at least the same as the ones obtained by using EUC or D-BML on both mel features and MFCC features?

Although different algorithms have been tried by us for this purpose, such as discriminant analysis with scatter matrices and principal component analysis [7,25,26], none of them yields satisfactory classification rates at low dimensionalities. Then, when considering the neural network technique, one very attractive possibility is to make use of its adaptive nature, i.e., the feature that enable neural networks to learn for classification, clustering, optimization, etc., from input samples. Thus, if we can build a network that can be trained to map input samples into a low dimensional space in which EUC is optimally favored, we may be able to keep classification performance the same or to achieve even better performance at the same time. This idea eventually leads us to consider the hybrid models of neural networks and Euclidean distance. In the following two sections, we will give detail descriptions about these hybrid models and the studies on them.

IV. The Hybrid Models of Neural Networks and Euclidean Distance

Feature extraction from mel spectra may be regarded as a mapping procedure for classification in which mel feature vectors originally in their m-dimensional space are mapped to the new feature vectors in an n-dimensional space under certain transform function. During this process, the dimensionality of feature vectors is changed from m to n. Accurate and efficient ASR will then require that these new feature vectors must be highly discriminative with respect to the categories of interest while having low dimensionality.

To be more specific, suppose we have a vector data set, $\{\mathbf{x}^p\}$, in an m-dimension real space R^m. There are total N vectors and C distinct categories in this set, and each $\mathbf{x}^p$ must belong to one of the categories, say category c. In their original space R^m, average values for category c can be calculated as

$$\mathbf{X}^c = \frac{1}{N_c} \sum_p^{N_c} \mathbf{x}^{p,c} \qquad \text{(IV-1)}$$

where $\mathbf{x}^{p,c} \in c$, N_c is the total number of vectors in category c with the constraint $\sum_c^C N_c = N$, and p numerates each of them. The square Euclidean distance to the mean of category c for any vector $\mathbf{x}^q$ in its original space then can be written as

$$dx(q, c) = \|\mathbf{x}^q - \mathbf{X}^c\|^2 = \sum_k^m (x_k^q - X_k^c)^2 \qquad \text{(IV-2)}$$

After mapping each input vector $\mathbf{x}^q$ in R^m to an output vector $\mathbf{o}^q$ in the new space R^n under a mapping Φ, $\mathbf{o}^q = \Phi(\mathbf{x}^q)$, average values for category c will become

$$\mathbf{O}^c = \frac{1}{N_c} \sum_p^{N_c} \mathbf{o}^{p,c} \qquad \text{(IV-3)}$$

and the square Euclidean distance to the mean of category c for any vector $\mathbf{o}^q$ in the new space will become

$$do(q, c) = \|\mathbf{o}^q - \mathbf{O}^c\|^2 = \sum_k^n (o_k^q - O_k^c)^2 \qquad \text{(IV-4)}$$

Meanwhile, the dimensionality of vectors is changed from m to n. Now we want this mapping to be optimized so that we can keep classification rates with EUC in space R^n at least the same as, if not higher than, in space R^m for n as small as possible.

A. The Simple Linear Hybrid Model

Our initial implementation for the above ideas is quite straightforward. A simple two-layer linear neural network is used to map $\mathbf{x}^q$ to $\mathbf{o}^q$. Therefore,

$$o_i = \sum_j^{M_{in}} wl_{ij} * x_j + w_{i,bias} \qquad \text{(IV-5)}$$

where wl_{ij} is the network weights, M_{in} is the number of units in input layer, which is equal to the number of dimensions of input samples, and $w_{i,bias}$ is the bias term. The number of units in output layer, M_{out}, will be varied to see how small it can be, certainly desiring $M_{out} < M_{in}$. In order for us to obtain comparable or better classification rates with EUC in the new space $R^{M_{out}}$, the cost function is designed as

$$E = \sum_q^N E_q = \sum_q^N \sum_c^C \frac{1}{2} \left(\frac{do(q,c)}{\sum_{c'}^C do(q,c')} - \zeta_c \right)^2$$

$$= \sum_q^N \sum_c^C \frac{1}{2} \left(\frac{do(q,c)}{S(q)} - \zeta_c \right)^2 \qquad \text{(IV-6)}$$

where $S(q) \equiv \sum_{c'}^C do(q, c')$, ζ_c is the desired target value for the square Euclidean distance of sample $\mathbf{o}^q$ to the mean of category c, and N is the total number of training samples.

Here, $\frac{do(q,c)}{S(q)} \leq 1$ will prevent E from blowing up and may be regarded as the normalized square Euclidean distance of sample o^q to the mean of category c. For target values, one reasonable choice is to set $\zeta_c = 0$ if $o^q \in c$ and $\zeta_c = 1/(C-1)$ otherwise. Minimizing this cost function will move the transformed input vectors as close as possible to their category means in space $R^{M_{out}}$, while trying to keep the means themselves separated. Thus, it attempts to create a new set of feature vectors for which a EUC will perform as well as possible and, at the same time, the dimensionality will be reduced from M_{in} to M_{out}. Figure 3 shows the architecture of linear neural network used for this simple linear hybrid model. Its outputs may be called the simple linear hybrid model (SLHM) feature vectors. Thus the output layer may be also called the feature layer. The update rule for **w1** is derived in Appendix F.

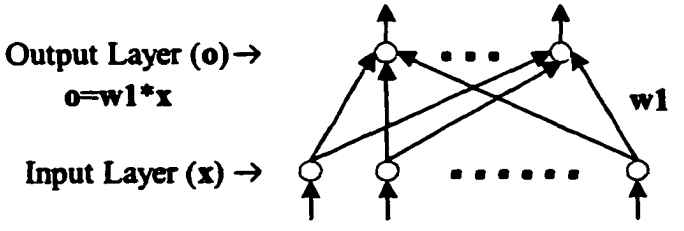


Figure 3:
The architecture of linear neural network used in the simple linear hybrid model

To train this hybrid model, the 672 samples introduced in section III are used. With a limited number of training samples, there is always a generalization problem [3,4,27,28]. In our work for this model and thereafter, we employ cross validation with early stopping strategy to test and improve generalization. Therefore, we randomly partition our samples into training, validation, and testing sets with the constraint that the

numbers of samples for each phoneme and each gender in each set are equal. There are 336 samples in training set, 168 samples in validation set, and 168 samples in testing set. Total 12 such partitions are created. All input mel features are linearly rescaled to the range 0 to 1. Network weights are initialized with random real numbers ranging from −1 to 1 for each trial, and 13 trials are carried out for each partition. The momentum coefficient, α, is set to 0.7 for this model and thereafter. The learning rate is chosen as 0.02. In this model and thereafter, we update network weights after each sample input instead of after all sample inputs, i.e., we prefer the incremental update mode since it generally makes convergence faster. Although many papers have discussed the selection of stopping criterion, there is no universal way that is good to all problems [29,30]. For our data here, the stopping criterion is chosen, after some experimentation, as the follows. Suppose the minimum cost function value of validation set in the first t epochs occurs at t_{min}. Then the training will stop at t if $t-t_{min}=1,500$, where t is up to 3,000 epochs. In another word, we train our networks and stops at the minimum cost function value up to 3,000 epochs. For each partition, the trial with the lowest validation set cost function at t_{min} is chosen as the best. Once such a trial with the minimum cost function value is selected for this partition, all input mel vectors will be mapped into SLHM vectors. As we have done with MFCC vectors, the leave-one-out method is then used to evaluate classification performance for EUC and D-BML with these SLHM vectors. Final classification rates for this network structure are then the averages from its 12 partitions. To see how reducing mel vector dimensionality affects classification performance, we vary the number of units in the output layer. Values for M_{out} from 1 to 10 are tried, and the results are given in Table 4.

Table 4
Percentages of Correct Classifications for SLHM Feature Vectors Obtained with The Simple Linear Hybrid Model of Neural Networks And Euclidean Distance (A Network Structure [M_{in}, M_{out}] Stands for A Linear Neural Network with M_{in} Input Units And M_{out} Output Units)

Network Structure	EUC	D-BML
[30, 1]	17.6	19.6
[30, 2]	16.5	18.9
[30, 3]	31.6	33.4
[30, 4]	55.3	53.4
[30, 5]	82.6	83.2
[30, 6]	84.5	86.0
[30, 7]	85.4	85.3
[30, 8]	85.4	86.1
[30, 9]	85.4	86.6
[30, 10]	85.8	86.5

From Table 4, as one can expect, classification rates increase with the number of dimensions of the new space. By comparing with Table 3, we can see that:

(1) When M_{out} is increased to 5, we get classification rates 82.6% for EUC and 83.2% for D-BML, respectively. They are approximately the same as the classification rate obtained from original mel features, but dimensionality is significantly reduced from n=20 (or 30) to only 5. They are also comparable with the classification rate obtained with MFCC features when n=6, and thus, with SLHM features, dimensionality may be further reduced by 1.

(2) For $5 \leq M_{out} \leq 10$, classification rate for D-BML from SLHM features is better than the ones for D-BML from MFCC features at each corresponding $n=M_{out}$.

(3) The maximum classification rate for D-BML is approximately the same in both tables, but may be obtained at a lower dimensionality with SLHM features.

(4) The difference between classification rates obtained for EUC and D-BML is much smaller here. This result is expected since the cost function is based on the Euclidean classifier.

(5) After $M_{out}=5$, classification rate does not have significant change. So, classification information in a mel vector with 20 or 30 dimensions may be described in a space of 5 dimensions. All other extra dimensions may then be regarded as redundant. This basically agrees with Davis *et al.*'s work as we have discussed in Section III.

The above results are better than we have expected. A lot of methods on feature extraction and linear mapping from perspective of classification have been studied [7,31]. However, from our experiences when trying some of those methods, it is very hard for them to have such good results. Since the neural network used in this model is relatively simple, we believe our selection of optimization criterion, i.e. the cost function, is primarily responsible for the quality of the classification performance.

B. The Simple Nonlinear Hybrid Model

As an alternative, we propose our second hybrid model, a simple nonlinear hybrid model, in which a three layer nonlinear neural network replaces the two layer linear one. Figure 4 shows the architecture of this network. The activation function between the input layer and hidden layer is a sigmoid function

$$y_i = f(H_i) = 1/(1+\exp(-H_i)) \qquad \text{(IV-7)}$$

where $H_i = \sum_j^{M_{in}} w1_{ij} * x_j + w1_{i,bias}$. The activation function between the hidden layer and output layer is a linear function, that is

$$o_i = \sum_j^{M_h} w2_{ij} * y_j + w2_{i,bias}. \qquad \text{(IV-8)}$$

where M_h is the number of hidden units. This nonlinear hybrid model has the same cost function as the linear one. Appendix G gives the derivation of update rules for **w1** and **w2**.

A nonlinear neural network is considered more powerful than a linear one due to its universal approximation capability [3,4]. However, in our models, this nonlinear neural network also requires a lot of more computation time. For the simple linear hybrid model, $M_{out}(M_{in}+1)$ network weights need to be updated. So, the training time for one epoch in incremental mode may be roughly estimated as

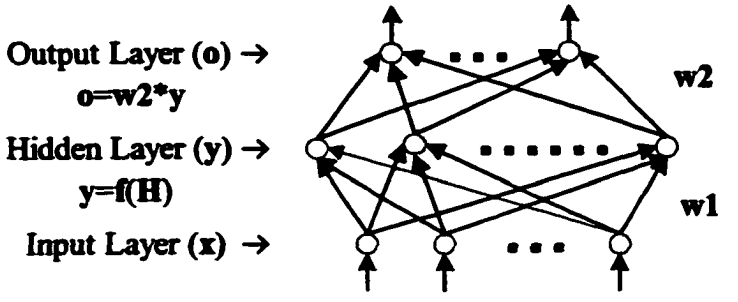


Figure 4: The architecture of nonlinear neural network used in the simple nonlinear hybrid model

$$O\left(N_{trn}\, M_{out}(M_{in}+1)\right)$$

where N_{trn} is the number of samples in training data set. In nonlinear case, we have $M_{out}(M_h+1)+M_h(M_{in}+1)$ network weights. Since the mean values for each category in space $R^{M_{out}}$ must be recalculated again from all N_{trn} samples after each update, the training time for one epoch in incremental mode may be roughly estimated as

$$O\left(N_{trn}^2\,(M_{out}(M_h+1)+M_h(M_{in}+1))\right)$$

This is about N_{trn} times longer than linear case. As a reference for how long the training time could be, here is a sample: On an IBM PC with Pentium IV 1.6GHz processor and 256MB memory, it takes about 27 hours to finish one 3,000 epochs trial for a middle size nonlinear neural network with 30, 15, and 4 units in its input, hidden and output layers, respectively. Considering we may need at least 2,400 such trials for different network structures if we follow similar training procedure as we have done for the simple linear hybrid model, this huge amount of computation easily surpasses our capability and time limit. Since we are going to spend a lot of time in the following on a more powerful modified model, the nonlinear Euclidean neural networks, here we present this simple nonlinear hybrid model as a conceptual completion of our hybrid models rather than an actually trained one.

C. Linear Euclidean Neural Networks

So far, all we have done with our simple linear hybrid model are straightforward:

(1) Training neural networks with lower output dimensions, M_{out}, under a cost function that is designed to keep samples from different categories apart in terms of Euclidean distance;

(2) Mapping input samples into space $R^{M_{out}}$ using transformation function obtained from training;

(3) Applying EUC and D-BML to all mapped SLHM features as we have done with mel data and MFCC data.

Although the great dimensionality reduction with basically unchanged classification rates is impressive when considering the simplicity of this model, classification rates are not as good as the ones using MFCC features when at very low dimensions, $M_{out} \leq 4$. Instead, they are worse.

After carefully reviewing this model, it should not be surprising if we realize that, in fact, the design of cost function (IV-6) has the following side effect: It forces the distances among each pair of category means to be equal when setting target values, ζ_c. Since N categories can be equidistant only in a space of N-1 dimensions or more, this design will certainly lower classification rates at very low dimensions. As a result, from perspective of classification, mean values that are calculated from mapped samples for EUC may still not be optimal even we have seen great improvements from their original space. Consequently, the obtained dimensionality reduction may not be optimal either. In above models, we also leave mean value templates of EUC being calculated in their conventional way. However, since optimal templates are so critical for high classification rates, there is really no reason to leave those templates untouched. Templates should be

closer to the optimal if they can be adaptively trained during training. By doing so, templates may no longer necessarily be the mean values of EUC.

In the attempt to remove the side effect and optimize templates directly, we modify our hybrid models above by adding one more layer to their neural networks, respectively. We now have total three layers for linear case and four layers for nonlinear case. The details of these two new models are as the follows, and let's see the linear one first.

In linear case, the architecture of the network is shown in Figure 5. Its first two layers are kept unchanged. They have the same activation function between them,

$$y_i = \sum_j^{M_{in}} w1_{ij} * x_j + w1_{i,bias} \qquad \text{(IV-9)}$$

but the output layer now becomes a hidden layer. The new added layer is the output layer, which uses a quadratic distance function as the activation function between itself and the hidden layer,

$$o_i = \sum_j^{M_h} (y_j - w2_{ij})^2 \qquad \text{(IV-10)}$$

where M_h is the number of hidden units. The number of units in output layer, M_{out}, will be equal to the number of categories, C.

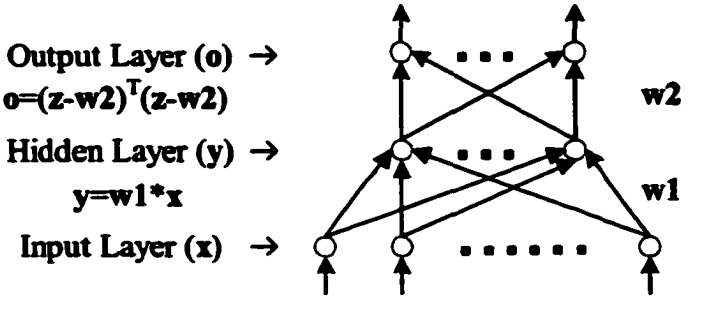


Figure 5:
The architecture of linear Euclidean neural network

With this design, since M_{out}=C and the activation function in the output layer is in the form of quadratic distance, network weights $\mathbf{w2}_i$ in (IV-10) then may be regarded to form a template for category i. Thus, output value o_i in (IV-10) will be the quadratic distance to the template of category i for vector $\mathbf{y}^q$. Here, vector $\mathbf{y}^q$ is formed by each unit value of hidden layer, which is calculated by (IV-9) from network weights and input vector $\mathbf{x}^q$. Comparing with the simple linear hybrid model above, we see that vector $\mathbf{y}^q$ can be regarded as an M_h-dimension vector mapped from input M_{in}-dimension vector $\mathbf{x}^q$. Therefore, for each input vector, this new hybrid model first maps it to a vector in new space R^{M_h}, then calculates its distance to the template of each category i. By finding the smallest output unit value from $\mathbf{o}^q$, classification for this input sample can be readily determined at the output layer. When each template $\mathbf{w2}_i$ equals to the mean values of each category i, network will make its classifications like a EUC. However, templates $\mathbf{w2}_i$ may not necessarily be EUC templates. We call the neural network in this linear hybrid model the linear Euclidean neural network (Euc-NN), and call the mapped vectors in new space R^{M_h} the linear Euclidean neural network (LENN) feature vectors. Thus, its hidden layer may also be called the feature layer.

As to the cost function, there exists another disadvantage when using the same form as (IV-6). It is often noted that many cost functions used for pattern classification may not be consistent with the minimum misclassification rate. In another word, minimized error measure may not necessarily mean minimized error count [6,32]. Therefore, the ideal solution would be the direct minimization on error count. From (IV-

10), if o_j^q has the smallest value, we may classify $\mathbf{x}^q$ as $\mathbf{x}^q \in j$. If the correct target category is t, we should have j=t. Otherwise, there will be a misclassification. Therefore, if we define the closest rival category r such that o_r^q has the smallest value among all categories except t, we can express the total number of misclassifications as

$$E = \sum_q^N S(o_t^q - o_r^q) \qquad \text{(IV-11)}$$

where S(e) is a step function with the property S(e)=0 for e<0 and S(e)=1 otherwise.

However, since step function in (IV-14) is not continuously differentiable for gradient decent based training, it needs to be replaced. For this purpose, sigmoid function is a widely accepted choice [32, 33]. Therefore, our cost function for this linear Euc-NN is defined as

$$E = \sum_q^N E_q = \sum_q^N \frac{1}{1+\exp(-a*(o_t^q - o_r^q))}$$

$$= \sum_q^N \frac{1}{1+\exp(-a*d^q)} \qquad \text{(IV-12)}$$

where $d^q \equiv o_t^q - o_r^q$, a is a positive number. This approach is a simplified version of minimum classification error (MCE) method of [32]. It is no longer based on any strict arguments or assumptions about samples' statistical distributions. However, one could make an analysis of what kind of distributions it would work best on. As one can see, forcing equidistance among category templates is also removed. The corresponding update rules for weights are given in Appendix H.

When comparing this linear Euc-NN with our previous hybrid models, there exists one more major difference, i.e., templates used for classification can be adaptively learned here as neural network weights through training. In other words, the values of

templates used for classification are no longer passively calculated from mapped vectors. Instead, they themselves will be optimized directly as part of a neural network during training, and thus are more general. There are a few papers that employ ideas similar to ours [34,35].

To test this model, the same data set and partitions described for the simple linear hybrid model are used. All the data preprocessing, network initializations, etc. are kept the same, unless otherwise stated. In our studies, we set a=1.0 and let learning rate equal to 0.001. Since our major concern is error count and our cost function is designed to simulate step function for this count, the stopping criterion is modified as the follows. If the minimum error count (with the smaller cost function as a tie breaker) in the first t epochs occurs at t_{min}, then the training will stop at t if $t-t_{min}=1,500$, where t is up to 3,000 epochs. As a reference, however, we also apply the same stopping criterion used in the

Table 5
Percentages of Correct Classifications with Linear Euclidean Neural Networks (A Network Structure [M_{in}, M_h, M_{out}] Stands for A Linear Euclidean Neural Network with M_{in} Input Units, M_h Hidden Units And M_{out} Output Units)

Network Structures	By Error Count Stopping Criterion			By Cost Function Value Stopping Criterion		
	Training Data Set	Validation Data Set	Testing Data Set	Training Data Set	Validation Data Set	Testing Data Set
[30, 1, 7]	59.9	60.6	57.2	56.4	54.3	53.6
[30, 2, 7]	87.8	82.1	79.3	89.6	81.3	79.4
[30, 3, 7]	93.6	86.3	83.1	94.3	85.6	82.9
[30, 4, 7]	95.3	89.8	87.1	95.5	88.8	86.6
[30, 5, 7]	95.9	89.6	87.5	96.4	88.8	87.4
[30, 6, 7]	96.6	89.7	88.1	96.8	88.4	87.6
[30, 7, 7]	96.0	90.7	88.2	96.9	89.6	88.6
[30, 8, 7]	96.5	90.0	88.1	97.1	88.7	88.6
[30, 9, 7]	96.6	89.5	87.8	97.1	88.8	88.1
[30, 10, 7]	96.3	89.3	88.1	96.4	88.2	88.4

simple linear hybrid model and monitor it. Therefore, we will have two sets of results: one from error count stopping criterion and the other from cost function value stopping criterion.

For error count stopping criterion, in each partition, classification rates from the trial that has the minimum error count in validation set are selected for this partition. For cost function value stopping criterion, in each partition, classification rate from the trial that has the minimum cost function value in validation set are selected for this partition. In both cases, the final results are the averages from 12 partitions. To see how dimensionality is reduced from M_{in}=30 to M_h, we vary the number of units in hidden layer from 1 to 10. Table 5 lists all the results. When comparing Table 5 with Table 3, we find great improvement by these linear Euc-NNs.

(1) The Maximum classification rate one can get may be increased from 86.6% to 88.2%.

(2) For any number of dimensions from 1 to 10, classification rate here is better than the corresponding one in Table 3, especially for $M_h \leq 4$.

(3) If, again, we use the classification rates obtained for D-BML at n=6 from MFCC features and n=30 from mel features as a benchmark, this linear Euc-NN can give comparable classification rate on testing set with M_h=3, which is 83.1%. We see significant dimensionality reduction, down to 3.

Since classification rates in Table 5 are obtained using the trained templates, the above discussion may also be regarded as a comparison between two classification

approaches. To directly compare between features, we need to follow the same procedure described for SLHM features to obtain LENN features first, except the trial selection for a partition is now based on error count stopping criterion. After that, the leave-one-out method used for Table 3 and 4 needs to be applied here to evaluate classification performance for EUC and D-BML with these LENN vectors. Table 6 gives the results.

Table 6
Percentages of Correct Classifications for LENN Feature Vectors Obtained with The Linear Euclidean Neural Networks (A Network Structure [M_{in}, M_h, M_{out}] Stands for A Linear Euclidean Neural Network with M_{in} Input Units, M_h Hidden Units And M_{out} Output Units)

Network Structure	EUC	D-BML
[30, 1, 7]	61.0	60.4
[30, 2, 7]	79.8	83.0
[30, 3, 7]	84.9	86.1
[30, 4, 7]	86.0	87.7
[30, 5, 7]	86.7	88.9
[30, 6, 7]	86.5	88.5
[30, 7, 7]	86.7	88.9
[30, 8, 7]	86.2	88.6
[30, 9, 7]	86.0	88.3
[30, 10, 7]	86.6	88.1

When comparing Table 6 with Table 3 and 4, we see that both EUC and D-BML give better classification rates with LENN features than with MFCC or SLHM features.

(1) With the benchmark classification rates obtained by using D-BML at n=6 with MFCC features, n=30 with mel features, we can get comparable classification rate, 83.0%, when M_h=2 by using D-BML. Again, we get significant dimensionality reduction, down to 2.

(2) For any number of dimensions from 1 to 10, corresponding classification rate here is significantly higher than the one in Table 3, and is greatly improved from Table 4, especially at low dimensions.

(3) The Maximum classification rate one can get may be increased from 86.6% to 88.9% with M_h=7.

The above results show that linear Euc-NNs are not only capable for highly discriminative, low dimensional feature extraction but also give high classification performance as a classifier. With such success, we continue our studies on nonlinear case next.

D. Nonlinear Euclidean Neural Networks

In nonlinear case, a nonlinear neural network replaces the linear one. The network architecture is shown in Figure 6. In this model, we now have two hidden layers. The activation function between first hidden layer and input layer is

$$y_i = f(H_i) = 1/(1+\exp(-H_i)) \qquad \text{(IV-13)}$$

where $H_i = \sum_j^{M_{in}} w1_{ij}*x_j + w1_{i,bias}$. And the activation function between the second hidden layer and the first hidden layer is

$$z_i = \sum_j^{M_{h1}} w2_{ij}*y_j + w2_{i,bias} \qquad \text{(IV-14)}$$

where M_{h1} is the number of units in first hidden layer. These two functions are actually the same as in the simple nonlinear hybrid model. Again, the new added output layer employs a quadratic distance function as the activation function between itself and the second hidden layer

$$o_i = \sum_j^{M_{h2}} (z_j - w3_{ij})^2 \qquad \text{(IV-15)}$$

where M_{h2} is the number of units in second hidden layer. The number of units in output layer, M_{out}, will also be equal to the number of categories, C.

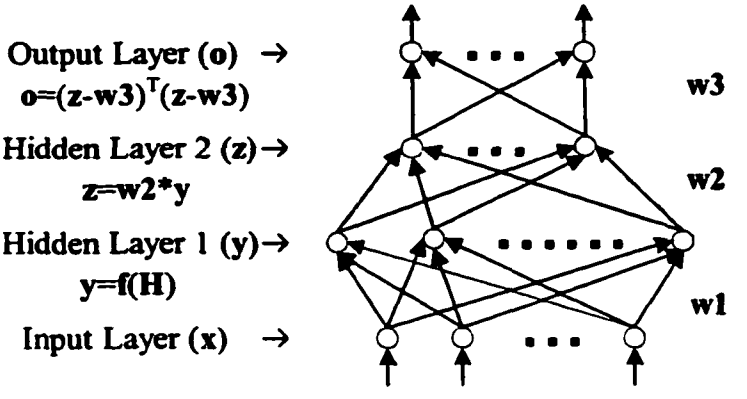


Figure 6:
The architecture of nonlinear Eulidean neural network

This new nonlinear hybrid model employs the same ideas as the linear hybrid model with a linear Euclidean neural network, except it has one more nonlinear layer in its network architecture. So, we call its network a nonlinear Euclidean neural network. For each input vector in $R^{M_{in}}$ space, a nonlinear Euc-NN first maps it into a vector, called the nonlinear Euclidean neural network (NENN) feature vector, in $R^{M_{h2}}$ space, then classifies it based on output unit values. Thus, the second hidden layer may also be called the feature layer. All our discussions on the characteristics and properties of a linear Euc-NN are valid for a nonlinear Euc-NN, too. The update rules for its weights are derived in Appendix I.

Table 7

Percentages of Correct Classifications with Nonlinear Euclidean Neural Networks (A Network Structure [M_{in}, M_{h1}, M_{h2}, M_{out}] Stands for A Nonlinear Euclidean Neural Network with M_{in} Input Units, M_{h1} First Hidden Layer Units, M_{h2} Second Hidden Layer Units, And M_{out} Output Units)

(a) M_{h2} Is Fixed while M_{h1} Is Varying

Network Structures	By Error Count Stopping Criterion			By Cost Function Value Stopping Criterion		
	Training Data Set	Validation Data Set	Testing Data Set	Training Data Set	Validation Data Set	Testing Data Set
[30, 1, 4, 7]	61.8	60.7	57.7	57.5	55.3	54.6
[30, 2, 4, 7]	86.9	82.4	77.5	89.3	80.7	78.2
[30, 3, 4, 7]	92.8	87.0	84.4	94	86.1	84.1
[30, 5, 4, 7]	96.0	88.5	85.1	95.9	87.6	85.8
[30, 7, 4, 7]	94.3	89.2	86.1	95.8	88.3	86.5
[30, 9, 4, 7]	95.1	88.9	86.5	95.5	88.4	86.1
[30, 11, 4, 7]	95.6	89.7	86.7	96	88.8	86.9
[30, 13, 4, 7]	95.7	89.6	87.2	96	88.8	87.5
[30, 15, 4, 7]	95.8	89.6	87.2	96.4	88.8	87.2
[30, 17, 4, 7]	95.6	89.2	86.6	96.2	88.4	86.6
[30, 19, 4, 7]	95.3	89.3	86.4	95.7	88.6	86.9
[30, 21, 4, 7]	96.0	89.5	86.8	96.5	89	86.8
[30, 23, 4, 7]	95.1	89.4	86.3	95.9	88.3	86.2

(b) M_{h1} Is Fixed while M_{h2} Is Varying

Network Structures	By Error Count Stopping Criterion			By Cost Function Value Stopping Criterion		
	Training Data Set	Validation Data Set	Testing Data Set	Training Data Set	Validation Data Set	Testing Data Set
[30, 15, 1, 7]	57.6	57.8	53.9	55.3	53.4	53.6
[30, 15, 2, 7]	92.4	84.8	82.8	92.7	84.1	83
[30, 15, 3, 7]	95.0	88.6	85.5	95.4	87.6	85.4
[30, 15, 4, 7]	95.8	89.6	87.2	96.4	88.8	87.2
[30, 15, 5, 7]	95.9	89.5	87.5	96.4	88.3	87.2
[30, 15, 6, 7]	95.7	90.0	87.5	96.5	89.3	87.5
[30, 15, 7, 7]	96.3	90.3	87.8	96.9	89.6	87.7
[30, 15, 8, 7]	96.7	89.7	87.7	96.9	88.7	87.7
[30, 15, 9, 7]	96.0	90.8	87.9	96.8	89.8	87.9
[30, 15, 10, 7]	96.7	89.7	87.7	96.6	89	87.6

The data partitions and the training procedures for linear Euc-NNs are also used for nonlinear Euc-NNs. However, studies are conducted in two steps since a nonlinear Euc-NN has two hidden layers. First, we need to determine the number of units to be used in the first hidden layer, M_{h1}. This is done as the follows: We vary the values of M_{h1} while setting M_{h2}=4. With each M_{h1} value, a nonlinear Euc-NN [30, M_{h1}, 4, 7] is trained with the same procedure as a linear Euc-NN. Table 7 (a) lists the results. Since the highest classification rate in testing set is obtained when M_{h1}=13 and 15, either 13 or 15 may be chosen for M_{h1}. In our work, M_{h1}=15 is selected. Secondly, we vary the values of M_{h2} while setting M_{h1}=15, i.e., we now try to reduce the dimensionality from 30 to M_{h2}. Again, $1 \leq M_h \leq 10$ and Table 7 (b) shows the results.

From Table 7 (b), we see great improvement similar to what we have obtained from linear Euc-NNs when comparing with Table 3.

(1) The Maximum classification rate one can get may be increased from 86.6% to 87.9%.

(2) For any number of dimensions from 1 to 10, classification rate here is better than the corresponding one in Table 3, especially at low dimensions.

(3) When M_{h2}=2, classification rate on testing set is 82.8%, which is comparable to the benchmark classification rates obtained by using D-BML at n=6 from MFCC features, n=30 from mel features. In another word, our nonlinear Euc-NNs can classify 7 phonemes in a space of just 2 dimensions. This is really impressive since it actually allows us to visually see the classifications in a X-Y plane. Figure 7 shows such a

graph for the classifications of samples in a testing set. And we see vowels, consonants and silence null phoneme basically form three groups, which are separated from each other clearly.

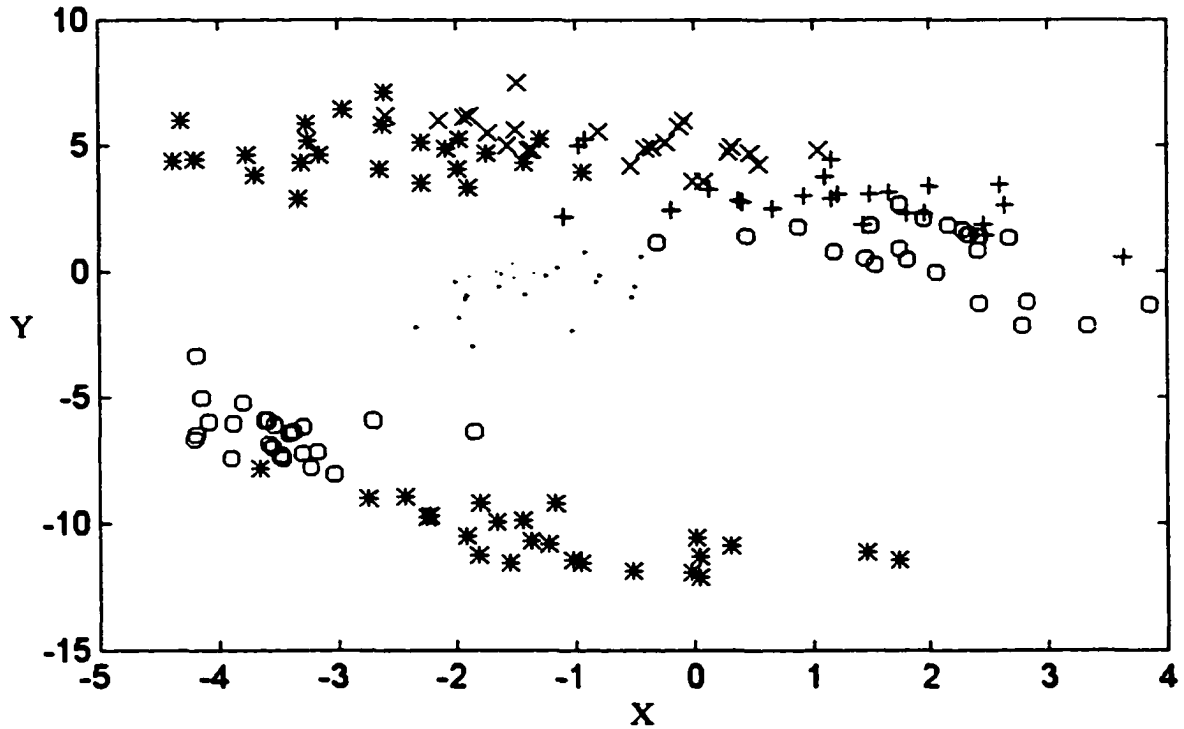


Figure 7: Two-dimensional graph for the classifications of 7 target phonemes of a testing data set with 87.5% classification rate. These data are NENN features. At the left-lower region, o for /f/ and * for /s/. For the rest, o for /EE/, x for /IH/, * for /EH/, + for /OO/, and the dots in the middle region for silence null phoneme. Total 168 samples.

By the same token, we now evaluate NENN features using the same procedure described above for LENN features. The results are in Table 8. When comparing with Table 3, 4 and 6, we see that both EUC and D-BML gave much better classification rates with NENN features.

(1) With the benchmark classification rates obtained by using D-BML at n=6 from MFCC features and n=30 from mel features, we can get comparable classification rate, 86.6%, when M_{h2}=2 by using D-BML. This classification rate is also higher than the one with LENN features. Again, we get significant dimensionality reduction, down to 2.

(2) For any number of dimensions from 1 to 10, corresponding classification rate here is significantly higher than the one in Table 3, and has been greatly improved from Table 4, especially at low dimensions.

(3) The Maximum classification rate one can get may be increased from 86.6% to 89.7% with M_h=10.

Table 8

Percentages of Correct Classifications for NENN Feature Vectors Obtained with The Nonlinear Euclidean Neural Networks (A Network Structure [M_{in}, M_{h1}, M_{h2}, M_{out}] Stands for A Nonlinear Euclidean Neural Network with M_{in} Input Units, M_{h1} First Hidden Layer Units, M_{h2} Second Hidden Layer Units, And M_{out} Output Units)

Network Structure	EUC	D-BML
[30, 15, 1, 7]	56.2	56.4
[30, 15, 2, 7]	83.6	86.6
[30, 15, 3, 7]	87.6	88.5
[30, 15, 4, 7]	87.6	89.5
[30, 15, 5, 7]	87.4	89.2
[30, 15, 6, 7]	87.6	89.0
[30, 15, 7, 7]	87.9	89.4
[30, 15, 8, 7]	87.5	89.4
[30, 15, 9, 7]	87.5	89.4
[30, 15, 10, 7]	87.7	89.7

Again, the above results show that a nonlinear Euc-NN is also capable for highly discriminative and low dimensional feature extraction. Moreover, as a classifier, it can give better classification performance than a linear one at very low dimensions, except at 1 dimension.

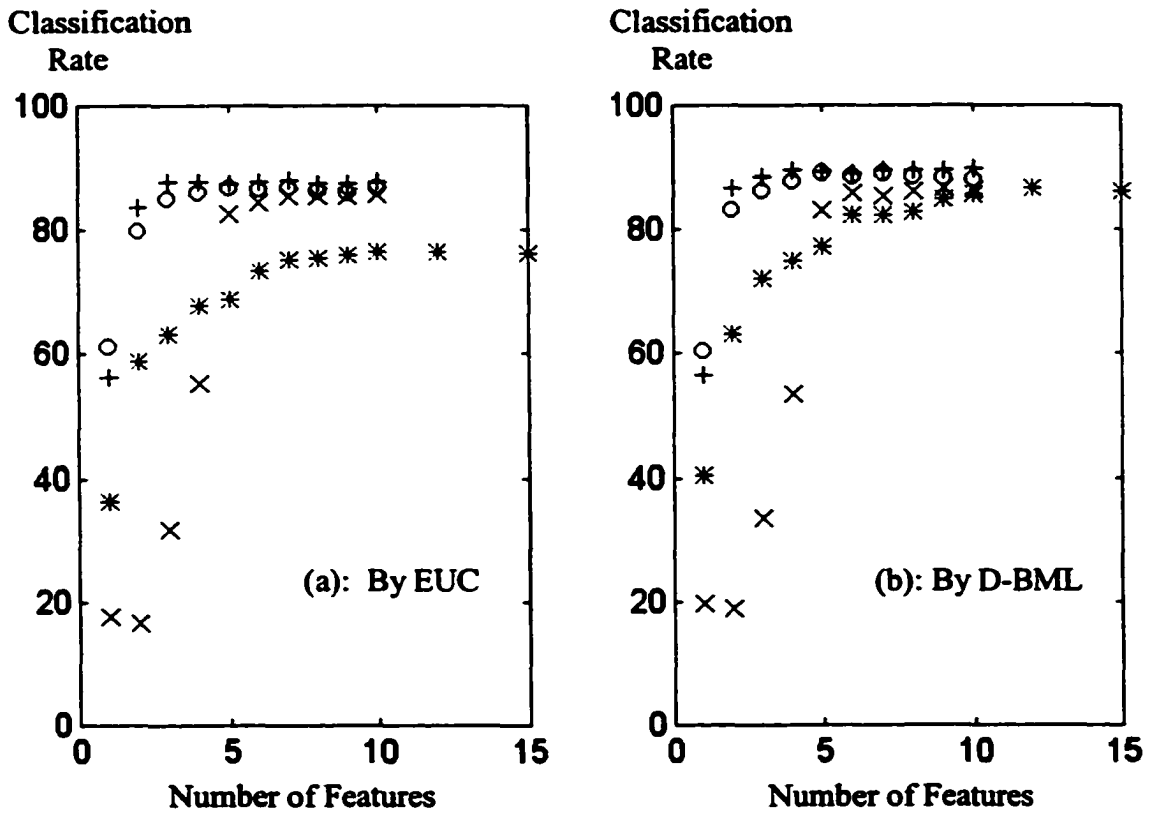


Figure 8: Correct Classifications from MFCC (*), SLHM (x), LENN (o), and NENN (+) features by Euclidean classifier and diagonal Bayes maximum likelihood classifier.

Since we apply the same classifiers in Table 3, 4, 6, and 8 to MFCC, SLHM, LENN, and NENN features respectively and all these features are extracted from the same mel spectra, classification rates in those tables can then give a direct comparison on their capability of capturing highly discriminative speech information at low dimensions.

Figure 8 schematically shows those results, (a) for EUC and (b) for D-BML. With both classifiers, classification rates obtained from our features are constantly higher than from classical MFCC features for the number of features 5 or greater. When the number of features is 4 or less, LENN and NENN features can give substantial improvement on classification rates. Especially for the classification rates obtained by using EUC, their improvement from the results with MFCC features is much greater than the corresponding one by using D-BML. However, as we have discussed above, SLHM features are not as good as MFCC features in this very low dimension range.

E. Further Discussions

Although we have great success with linear and nonlinear Euc-NNs, it is still worth to address that the templates obtained by our Euc-NNs are not the mean values calculated in EUC. It would be very nice if we could directly compare mean values with Euc-NN templates. Here, in nonlinear case, mean values for each unit of the second hidden layer are calculated in two methods:

(1) Calculate mean values from the samples in training set only since Euc-NN templates are obtained mainly by training with those samples.

(2) Calculate mean values from the samples in both training and validation sets since Euc-NN templates are really determined by both of them.

We apply these two methods to the samples used in Figure 7 and list the results in Table 9 (a). In either case, mean values are different from trained templates. For linear Euc-NNs, we have the similar results, too. Table 9 (b) gives a comparison in three dimensions. To

our knowledge, we have not seen similar templates like what we obtain here, especially in speech recognition.

Table 9

(a) Templates Obtained by A Nonlinear Euc-NN [30, 15, 2, 7] And Mean Values of EUC

Phonemes	Templates by Nonlinear Euc-NN	Mean Values (Training Samples Only)	Mean Values (Training & Validation Samples)
/EE/	(3.38, 0.14)	(1.95, 0.65)	(2.04, 0.56)
/IH/	(0.82, 7.41)	(-0.86, 5.53)	(-0.74, 5.42)
/EH/	(-5.13, 3.44)	(-2.70, 4.46)	(-2.62, 4.34)
/OO/	(3.25, 3.85)	(0.43, 3.34)	(0.38, 3.38)
/f/	(-1.85, -4.15)	(-3.59, -5.96)	(-3.60, -5.89)
/s/	(0.56, -7.86)	(-1.69, -10.12)	(-1.63, -10.03)
/SL/	(-1.16, -2.15)	(-1.31, -0.66)	(-1.33, -0.58)

(b) Templates Obtained by A Linear Euc-NN [30, 3, 7] And Mean Values of EUC

Phonemes	Templates by Linear Euc-NN	Mean Values (Training Samples Only)	Mean Values (Training & Validation Samples)
/EE/	(2.72, -1.53, 1.41)	(2.38, -0.38, 0.69)	(2.33, -0.41, 0.58)
/IH/	(3.15, 1.92, -2.09)	(2.01, 1.37, –0.79)	(1.95, 1.43, -0.74)
/EH/	(0.89, 4.12, 1.17)	(0.04, 3.09, –0.42)	(0.038, 3.07, -0.57)
/OO/	(-0.56, -0.90, -2.28)	(0.18, 1.00, –1.48)	(0.31, 1.00, -1.41)
/f/	(-3.14, 2.07, 2.80)	(-3.32, 0.84, 2.64)	(-3.28, 0.86, 2.69)
/s/	(-2.71, -2.39, 0.86)	(-3.75, -1.69, 2.72)	(-3.69, -1.96, 2.73)
/SL/	(-1.45, -1.41, 0.36)	(-0.61, 0.28, 0.88)	(-0.55, 0.23, 0.90)

We have shown above that our hybrid models of neural networks and Euclidean distance can be very good in feature extraction and dimensionality reduction. Especially with linear and nonlinear Euc-NNs, we are able to reduce input dimensionality significantly while preserving highly discriminative information. Some of these results

are very impressive. To see if we can extent our Euc-NN models to other widely used databases, we decide to conduct more studies on TIMIT data in the following section.

V. Euclidean Neural Network Studies on TIMIT Data

The Texas Instruments/Massachusetts Institute of Technology (TIMIT) corpus of read speech has been widely used in speech processing in the last two decades [36,37]. It contains 630 male/female speakers representing 8 major dialect divisions of American English. In our studies, a total of 16 vowels are used. They are, as labeled in TIMIT documentation [38], /IY/, /IH/, /EH/, /EY/, /AE/, /AA/, /AW/, /AY/, /AH/, /AO/, /OY/, /OW/, /UH/, /UW/, /UX/, and /ER/ (see Appendix K). Samples from SX sentences in TIMIT training set are chosen as our training data since SX samples provide a good hand-designed coverage of phones comprehensively and compactly. All samples from SX sentences in TIMIT core testing set are chosen as our testing data. In this section and the next one, all tests are conducted on these selected core testing samples, unless otherwise stated.

A. Data Preparation

A 20ms frame is taken from the center of each vowel in both SX training set and core testing set. Here, we use the existing TIMIT segmentation results. Total 18,996 samples for our training set and 953 samples for our testing set are obtained respectively. All frames are then processed in the same way as we have done in Section III to get TIMIT mel feature vectors and MFCC feature vectors. Since samples of TIMIT corpus are already separated into a training set and a testing set, the leave-one-out method is not

Table 10

Percentages of Correct Classifications of TIMIT SX Samples by Using D-BML And EUC

* The number of mel data used in Davis et al.'s work

Feature Vectors	D-BML		EUC	
	Training Data Set	Testing Data Set	Training Data Set	Testing Data Set
mel Feature Vectors (20)*	43	43.1	40.6	39.9
mel Feature Vectors (30)	42.7	43.5	40.4	40.7
MFCC Feature Vectors (1)	16.6	17.3	12.2	12.5
MFCC Feature Vectors (2)	25.4	26.7	19.7	20.5
MFCC Feature Vectors (3)	28.5	30.4	22.7	22.8
MFCC Feature Vectors (4)	35.7	36.3	29.3	28.4
MFCC Feature Vectors (5)	40.3	39.6	33.1	32.1
MFCC Feature Vectors (6)	42.1	42.3	34.6	34.4
MFCC Feature Vectors (7)	43.2	43.2	35.9	36.3
MFCC Feature Vectors (8)	44.6	44.6	36.6	36.7
MFCC Feature Vectors (9)	44.6	44.8	36.8	37.1
MFCC Feature Vectors (10)	45.6	46.0	37.0	37.9
MFCC Feature Vectors (12)	46.5	46.9	37.3	38.6
MFCC Feature Vectors (15)	47.4	48.2	37.8	38.7
MFCC Feature Vectors (30)	45.9	46.4	37.9	38.8

used in the studies of this section. The D-BML and EUC are trained with all TIMIT SX training samples and tested with the testing set. The results are shown in Table 10, and the number inside the parenthesis of the first column has the same meaning as in Table 3. By using D-BML, with the first 6 and 10 MFCCs, we have classification rates on testing set 42.3% and 46.0% respectively. The highest classification rate on testing set obtained is 48.2% when using the first 15 MFCCs. By using EUC, with the first 6 and 10 MFCCs, we have classification rates on testing set 34.4% and 37.9% respectively. The highest classification rate on testing set obtained is 40.7% when using all 30 mel features.

Since SX training data set is big and the training on it is very time consuming, we will study our models with portion of it. In these studies, about 15% of samples in SX

training set are randomly selected to form a small data set under the constraint that each vowel has approximately the same percentage as in SX training set, which gives total 2,843 samples. This small data set is further randomly partitioned into a training set and a validation set with the constraint that in either set each vowel keeps approximately the same percentage as in SX training set. There are 1,892 samples in training set and 951 samples in validation set. Total 6 such partitions are created. For each partition, a Euc-NN will be initialized 12 times with different random weights, one for each trial. The range of initial weights is (-1, 1). All input mel data are linearly rescaled into [0, 1].

In each trial, the stopping criterion is as follows: Suppose the minimum error count with the minimum cost function value in validation set occurred at t_{min} in the first t epochs. Then the training will stop if $t-t_{min}=5,000$. Again, as a reference, we also apply the cost function value stopping criterion. For both linear and nonlinear Euc-NNs, a=3.0, and the learning rate is initially equal to 0.005. For nonlinear Euc-NNs, the learning rate will decrease with a factor 0.75 for every 5,000 epochs. These network parameters are determined from our experimental trials and can give us better classification rates and faster convergence.

Since this small data set is different from the training data set used in Table 10, any comparison between the results obtained with the small data set and the results of Table 10 will be a restricted one only. In order to have a direct comparison, after we have studied with this small data set, we will select one network structure and train it with all SX training samples. This will be done in both linear and nonlinear cases.

B. Studies With Linear Euclidean Neural Network

In these studies, classification rates with each different linear Euc-NN structure are obtained the same way as we have described for Table 5, except we have 6 partitions here. Table 11 shows all these results. When M_h=5, the classification rate on testing set is 46.2%. If we compare it with the results on MFCCs in Table 10, this value is comparable to the classification rate obtained by using D-BML with the first 10 MFCCs and close to the classification rate with the first 12 MFCCs. For M_h>5, classification rates basically do not have great changes.

Table 11

Percentages of Correct Classifications with Linear Euclidean Neural Networks on Small Data Set (A Network Structure [M_{in}, M_h, M_{out}] Stands for A Linear Euclidean Neural Network with M_{in} Input Units, M_h Hidden Layer Units, And M_{out} Output Units)

Network Structures	By Error Count Stopping Criterion			By Cost Function Value Stopping Criterion		
	Training Data Set	Validation Data Set	Testing Data Set	Training Data Set	Validation Data Set	Testing Data Set
[30, 1, 16]	29.0	29.2	29.3	22.4	22.4	20.8
[30, 2, 16]	36.5	36.8	34.7	29.1	28.0	26.9
[30, 3, 16]	47.1	45.8	44.1	34.9	34.1	32.8
[30, 4, 16]	49.0	47.8	45.3	49.0	47.8	45.3
[30, 5, 16]	49.7	47.7	46.2	49.7	47.7	46.2
[30, 6, 16]	50.0	47.7	45.9	50.0	47.7	45.9
[30, 7, 16]	51.9	48.6	46.9	50.0	46.9	44.6
[30, 8, 16]	51.8	48.7	46.5	51.8	48.7	46.7
[30, 9, 16]	51.7	48.8	46.9	51.7	48.8	46.7
[30, 10, 16]	51.5	48.2	46.6	51.3	48.2	46.6

Similar to what we have done for Table 6, for each partition we select a trial based on error count stopping criterion and then map samples of the small data set and the testing set into LENN vectors. All samples in the small data set are used to train D-BML

while test is conducted on testing set. The final results are the averages from all partitions and are listed in Table 12. When M_h=4, the classification rate on testing set is 46.2%. Again, this value is comparable to the classification rate obtained by using D-BML with the first 10 MFCCs and close to the classification rate with the first 12 MFCCs in Table 10. However, the dimensionality here is dramatically reduced to only 4. As a reference, we also apply EUC to those LENN features using the same procedure and the results are given in Table 12. If we compare them with the results by using EUC in Table 10, we see that the classification rates obtained with LENN features are much higher than with MFCC features.

Table 12

Percentages of Correct Classifications for LENN Feature Vectors Obtained with Linear Euclidean Neural Networks of Table 11 by Using D-BML And EUC (A Network Structure [M_{in}, M_h, M_{out}] Stands for A Linear Euclidean Neural Network with M_{in} Input Units, M_h Hidden Units And M_{out} Output Units)

Network Structures	D-BML		EUC	
	Training Data Set	Testing Data Set	Training Data Set	Testing Data Set
[30, 1, 16]	30.8	28.9	22.5	21.5
[30, 2, 16]	37.8	37.0	32.7	30.9
[30, 3, 16]	45.9	45.8	42.1	41.5
[30, 4, 16]	47.5	46.2	43.6	43.2
[30, 5, 16]	48.1	46.7	44.0	43.3
[30, 6, 16]	48.0	46.3	44.0	43.4
[30, 7, 16]	48.4	46.6	44.6	44.2
[30, 8, 16]	48.6	46.4	44.4	44.2
[30, 9, 16]	48.3	46.3	44.7	44.2
[30, 10, 16]	48.2	46.6	44.6	44.1

The above results are obtained with a portion of SX training samples, about 15%. As we have discussed above, we then select a network structure and train it with all SX

training samples. Considering that Davis *et al.* use 6 to 10 MFCCs, network structure [30, 5, 16] is selected.

All TIMIT SX training samples are randomly partitioned into a training set and a validation set with the constraint that each vowel keeps approximately the same percentage in each set. There are 18,044 samples in training set and 952 samples in validation set. Total 6 such partitions are created. The training procedures are the same as what we have done with the small data set. Table 13 shows the results. On testing set,

Table 13

Percentages of Correct Classifications with Linear Euclidean Neural Networks on All SX Data (A Network Structure [M_{in}, M_h, M_{out}] Stands for A Linear Euclidean Neural Network with M_{in} Input Units, M_h Hidden Layer Units, And M_{out} Output Units)

Network Structure	By Error Count Stopping Criterion			By Cost Function Value Stopping Criterion		
	Training Data Set	Validation Data Set	Testing Data Set	Training Data Set	Validation Data Set	Testing Data Set
[30, 5, 16]	46.4	48.0	47.3	46.4	48.0	47.3

Table 14

Percentages of Correct Classifications for LENN Feature Vectors Obtained with Linear Euclidean Neural Networks of Table 13 by Using D-BML And EUC (A Network Structure [M_{in}, M_h, M_{out}] Stands for A Linear Euclidean Neural Network with M_{in} Input Units, M_h Hidden Units, And M_{out} Output Units)

Network Structure	D-BML		EUC	
	Training Data Set	Testing Data Set	Training Data Set	Testing Data Set
[30, 5, 16]	46.8	47.8	42.9	44.8

47.3% classification rate is obtained for M_h=5. This is higher than the 46.0% classification rate obtained by using D-BML on the first 10 MFCCs, and the 46.9%

classification rate with the first 12 MFCCs. So, again here, we get significant dimensionality reduction by our linear Euc-NNs.

We now apply D-BML to LENN features obtained from the linear Euc-NN of Table 13. Following the same procedure as we have done for Table 12, except that all SX TIMIT samples are used this time, we obtain the results as shown in Table 14. When we compare these results with the ones in Table 10, we see our LENN features can preserve approximately the same highest classification rates as MFCC features at a significantly lower dimensionality. If we use the results with 6 MFCCs and 10 MFCCs as a benchmark, the results with 5 LENN features are much better. The results by using EUC are also given in Table 14, and they are better than the ones in Table 10.

C. Studies With Nonlinear Euclidean Neural Network

Studies on nonlinear Euc-NNs are also conducted in two steps. First, we need to determine M_{h1} value. From the results obtained in linear Euc-NNs, we decide to set $M_{h2}=5$ while varying M_{h1}. A pilot testing set is formed by randomly selecting 952 samples from the SX training data that are not selected in the previous small data set. The final results are obtained the same way as in linear case and are shown in Table 15 (a). As we can see, network structures with $M_{h1}=27$ give the highest classification rates on the pilot testing set. Therefore, we select $M_{h1}=27$. In the second step, we set $M_{h1}=27$ while varying M_{h2}. All results for $1 \leq M_{h2} \leq 10$ are in Table 15 (b). We see that the classification rate on testing set at $M_h=5$ is 46.2%, which again is comparable to the classification rate obtained by using D-BML with the first 10 MFCCs.

Table 15

Percentages of Correct Classifications with Nonlinear Euclidean Neural Networks on Small Data Set (A Network Structure [M_{in}, M_{h1}, M_{h2}, M_{out}] Stands for A Nonlinear Euclidean Neural Network with M_{in} Input Units, M_{h1} First Hidden Layer Units, M_{h2} Second Hidden Layer Units, And M_{out} Output Units)

(a) Studies to Determine M_{h1}: M_{h1} Is Varying while M_{h2} Is Fixed

Network Structures	By Error Count Stopping Criterion			By Cost Function Value Stopping Criterion		
	Training Data Set	Validation Data Set	Pilot Testing Data Set	Training Data Set	Validation Data Set	Pilot Testing Data Set
[30, 13, 5, 16]	52.6	44.7	43.5	42.8	39.1	37.9
[30, 15, 5, 16]	51.9	45.7	44.6	47.0	41.6	40.6
[30, 17, 5, 16]	57.0	47.4	44.9	45.7	39.6	38.4
[30, 19, 5, 16]	56.8	47.3	45.7	47.6	41.4	40.0
[30, 21, 5, 16]	56.8	47.9	45.1	49.1	41.0	41.9
[30, 23, 5, 16]	58.1	47.6	45.9	46.5	40.3	39.6
[30, 25, 5, 16]	56.1	47.4	45.3	49.1	42.2	40.1
[30, 27, 5, 16]	56.6	47.8	46.7	48.6	41.0	40.3
[30, 29, 5, 16]	54.0	46.7	45.4	53.1	46.5	45.0
[30, 31, 5, 16]	54.3	46.9	44.9	52.8	44.7	44.3
[30, 33, 5, 16]	55.7	47.0	44.7	48.4	43.4	42.8

(b) M_{h2} Is Varying while M_{h1} Is Fixed

Network Structures	By Error Count Stopping Criterion			By Cost Function Value Stopping Criterion		
	Training Data Set	Validation Data Set	Testing Data Set	Training Data Set	Validation Data Set	Testing Data Set
[30, 27, 1, 16]	31.5	29.0	28.1	21.8	20.2	20.4
[30, 27, 2, 16]	46.6	37.1	35.7	29.2	26.6	25.6
[30, 27, 3, 16]	61.2	46.9	44.0	37.5	34.0	34.4
[30, 27, 4, 16]	58.5	48.1	45.1	43.6	38.4	38.3
[30, 27, 5, 16]	57.3	47.7	46.2	48.4	41.2	41.3
[30, 27, 6, 16]	46.9	44.8	45.0	44.9	43.3	43.7
[30, 27, 7, 16]	51.5	45.3	45.1	49.3	44.1	44.7
[30, 27, 8, 16]	44.7	43.5	42.4	43.3	41.8	41.8
[30, 27, 9, 16]	42.1	39.9	39.1	41.6	39.4	38.7
[30, 27, 10, 16]	44.5	42.5	42.1	44.1	41.8	41.5

Following the similar procedure used for Table 12, we calculate the classification rates with NENN features obtained from Table 15 (b). Table 16 shows all the results. When M_h=5, the classification rate on testing set using D-BML is 45.1%. Although this value is close to the classification rate obtained by using D-BML with the first 10 MFCCs in Table 10, it is not as good as the one obtained in linear case. If we compare EUC results with Table 10, we see that the classification rates obtained with NENN features are much higher than with MFCC features.

Table 16

Percentages of Correct Classifications for NENN Feature Vectors Obtained with The Nonlinear Euclidean Neural Networks of Table 15 (b) by Using D-BML And EUC (A Network Structure [M_{in}, M_{h1}, M_{h2}, M_{out}] Stands for A Nonlinear Euclidean Neural Network with M_{in} Input Units, M_{h1} First Hidden Layer Units, M_{h2} Second Hidden Layer Units, And M_{out} Output Units)

Network Structures	D-BML		EUC	
	Training Data Set	Testing Data Set	Training Data Set	Testing Data Set
[30, 27, 1, 16]	29.2	27.4	25.2	23.3
[30, 27, 2, 16]	35.6	32.0	35.5	31.0
[30, 27, 3, 16]	49.6	42.5	49.5	40.8
[30, 27, 4, 16]	50.8	44.2	48.9	41.4
[30, 27, 5, 16]	50.5	45.1	48.4	42.8
[30, 27, 6, 16]	45.8	44.6	44.2	43.7
[30, 27, 7, 16]	48.8	46.4	46.6	44.2
[30, 27, 8, 16]	44.7	44.4	43.3	43.2
[30, 27, 9, 16]	44.3	43.9	42.2	41.4
[30, 27, 10, 16]	44.7	43.9	43.1	42.9

Using the same reasoning as in the linear case, we select network structure [30, 27, 5, 16] and train it with all SX training data. The same SX sample partitions that we have used for linear Euc-NN [30, 5, 16] are also applied for this nonlinear Euc-NN network [30, 27, 5, 16] and the same training procedure is followed. Table 17 shows the

results. On testing set, 47.6% classification rate is obtained for M_h=5. This classification rate is higher than the 46.0% classification rate obtained by using D-BML with the first 10 MFCCs and the 46.9% classification rate with the first 12 MFCCs.

In both linear and nonlinear cases, classification rates on testing set obtained using all SX training samples are higher than the ones obtained using the small data set, suggesting that our Euc-NNs have better generalization when trained with all SX training samples. This agrees with the results in [41].

Table 17
Percentages of Correct Classifications with Nonlinear Euclidean Neural Networks on All TIMIT SX data (A Network Structure [M_{in}, M_{h1}, M_{h2}, M_{out}] Stands for A Nonlinear Euclidean Neural Network with M_{in} Input Units, M_{h1} First Hidden Layer Units, M_{h2} Second Hidden Layer Units, And M_{out} Output Units)

Network Structure	By Error Count Stopping Criterion			By Cost Function Value Stopping Criterion		
	Training Data Set	Validation Data Set	Testing Data Set	Training Data Set	Validation Data Set	Testing Data Set
[30, 27, 5, 16]	50.6	49.9	47.6	47.6	45.7	45.3

Table 18
Percentages of Correct Classifications for NENN Feature Vectors Obtained with The Nonlinear Euclidean Neural Networks of Table 17 by Using D-BML And EUC (A Network Structure [M_{in}, M_{h1}, M_{h2}, M_{out}] Stands for A Nonlinear Euclidean Neural Network with M_{in} Input Units, M_{h1} First Hidden Layer Units, M_{h2} Second Hidden Layer Units, And M_{out} Output Units)

Network Structure	D-BML		EUC	
	Training Data Set	Testing Data Set	Training Data Set	Testing Data Set
[30, 27, 5, 16]	28.1	27.4	36.7	35.3

Following the same procedure as in linear case, we now apply D-BML and EUC to NENN features obtained with the Euc-NNs in Table 17. The results are listed in Table

18. Although classification rates calculated from the trained templates are good, the results for D-BML are much worse than those in Table 10. It suggests that the nonlinear mapping with nonlinear Euc-NNs may have greatly changed sample distribution and the assumption for D-BML can no longer be held. Further investigations will be needed to better understand these results.

D. Comparing Linear And Nonlinear Euclidean Neural Networks

In both previous section and this one, we obtain very good results from both linear and nonlinear Euc-NNs. In the previous section, if we compare Table 5 with Table 7 (b) and Table 6 with Table 8, we find that classification rates from nonlinear Euc-NNs are moderately better than from linear Euc-NNs at low dimensions, $2 \leq n \leq 5$. This can be expected since the nonlinear neural network in nonlinear Euc-NNs is considered more powerful due to its universal approximation capability. In this section, at low dimensions $2 \leq n \leq 5$, if we compare Table 11 with Table 15 (b) and Table 13 with Table 17, we find that classification rates from nonlinear Euc-NNs are almost the same as from linear Euc-NNs. However, if we compare Table 12 with Table 16 and Table 14 with Table 18, we find that classification rates from linear Euc-NNs are better than from nonlinear Euc-NNs. This suggests that the NENN feature vectors may not preserve discriminative information well for D-BML and the reasons behind it need further investigation.

Studies on both linear and nonlinear cases in these two sections also show that when the number of categories and database size increase, problems given to Euc-NNs become more complicated and thus training these Euc-NNs are thus more difficult in the sense of network parameter selection, computation time, etc.

In our studies, we keep network parameters, such as the value of "a" in cost function, learning rate, initial weights, etc., the same for each network structure of a hybrid model. For example, in Table 15 (b), all the network structures share the same network parameter values or parameter generation logic. But we see their classification rates on testing set gradually decrease, rather than increase or saturate, when M_{h2} increases from 5. This contradicts the belief that higher dimensionality generally allows better description. We believe this is partially because the network parameters may not be optimal for network structures with higher M_{h2} values, since they are experimentally determined with M_h value equal to 5. So, ideally we may need to determine those network parameters for each network structure respectively to ensure classification performance, especially for nonlinear Euc-NNs or complicated problems. However, as one can imagine, this will be very time consuming.

Our studies also show that nonlinear Euc-NNs are generally harder than linear ones in terms of the following aspects.

(1) It is more difficult to implement and test a nonlinear Euc-NN than a linear one from the perspective of programming.

(2) The training time for a nonlinear Euc-NN is much longer than a linear one since it has one more layer with, generally, lots of units. For example, when using an IBM PC with Pentium IV 1.6GHz and 256MB memory, for a [30, 5, 16] linear Euc-NN, the training time (including computation time on validation set and network weights saving time) for the first 100 epochs is about 36 seconds. But for a [30, 27, 5, 16]

nonlinear one, the corresponding training time is about 895 seconds. The later one is about 23 times longer.

(3) One needs to experimentally determine an optimal M_{h1} value for each specific problem since a nonlinear Euc-NN has one more hidden layer. In our work in both this section and the previous section, we fix only one M_{h2} value when determining the optimal M_{h1} value. However, since an optimal M_{h1} value may be associated with proper M_{h2} values, one should fix each appropriate M_{h2} value and vary M_{h1} values respectively. In other words, the procedure for M_{h1} determination should actually be a 2-dimensional search. And, obviously, this will introduce a lot of more computation time.

(4) More trials will be needed for a nonlinear Euc-NN to converge to global minimum than a linear Euc-NN. As well known in neural network field, there is only one global minimum in a linear network with cost function quadratic in weights. However, if the activation function is nonlinear sigmoid function, there will be additional local minima [3,4]. In this case, convergence to a local minimum, which usually gives poor classification performance, will be an important issue. It is possible that the different cost function and additional sigmoid activation functions of the nonlinear Euc-NN make the local minimum problem more severe. As a result, on the average, more trials will be needed here to find the global minimum. For example, the classification rates of Euc-NN [30, 27, 9, 16] in Table 15 (b) is lower than its previous and next neighbors',

> although it suggests there should be no such a decrease there. When we trace back on each trial in each partition, we find that the classification rates from two partitions are low for all their 12 trials. After we add more trials with new random initial weight values for these two partitions, we basically can match classification rates with other partitions. Obviously, those original trainings are trapped at local minima with high error counts while the new added ones do not. However, this kind of problems happens much less in linear case.

So, when training time or convergence becomes a serious problem, our studies above suggest that using linear Euc-NNs probably be a better choice.

In cost function (IV-12), theoretically speaking, the value of "a" determines how close this sigmoid function to a step function: The bigger "a" value is, the closer it will be. However, since the values of d^q may vary widely in the initial training stages, big "a" value could make training very difficult, or even blowing up. On the other hand, if the value of "a" is too small, the sigmoid function may be too far away from a step function. And as a result, the minimization of the cost function may not necessarily give minimum error count. In fact, we have spent a lot of time for proper "a" value selections in the above studies on both linear and nonlinear cases. So, one may regard it as a disadvantage in selecting this kind of cost function.

Early stopping of training is aimed to prevent a neural network from being over trained due to too powerful a model class, noisy training samples, or a small training set [3,30]. As for the two stopping criteria we have used for both linear and nonlinear Euc-NNs, they basically give no significant classification performance difference in Section

IV and in linear Euc-NNs of this section, except in the range of very low dimensions, in which error count stopping criterion gives moderately better results. However, the corresponding results in nonlinear Euc-NN studies of this section are not consistent. Error count stopping criterion gives constantly better classification performance on testing set. Thus, the choice of this criterion is more practical. Since cost function (IV-12) suggests that there should be close correlation between the results of the two criteria, one might be able to improve it if different network parameters, the value of "a" and the range of initial weights, are selected.

VI. Progress Report on Studies with Time Warping Recurrent Neural Network

In our work above, all sample frames in our database are taken at certain regions of phonemes (see part A in Section III) and each of them is treated equally as an independent individual sample, even for frames taken from the same phonemes. Although these static feature processing is most important for phoneme discrimination, we have not considered the variation of features over time. For example, diphthongs, are pronounced in a gliding manner and thus their spectral characteristics vary in time [1]. These variations over the duration of a phoneme can yield additional temporal information that can improve classification performance [39,40,41, 42]. Therefore, a complete analysis for phoneme recognition should include the entire signal history and classifications should be based on all the frames taken sequentially from a phoneme segment.

This kind of sequence analysis, or time series analysis, has drawn a lot of attentions in the past two decades with its applications not only for speech recognition [5,43,44]. Many methods have been proposed, such as dynamic programming, hidden Markov model (HMM), neural networks, etc. In all these methods, HMM has been proved the most successful in speech recognition and has been widely used [46]. However, for time warping problem as discussed in introduction, HMM can not deal with it directly. Rather, HMM learns its state transition probabilities from the statistics of training samples [18]. As a result, if a testing sample with time warping pattern different from what being presented by training samples, it is unlikely that HMM will produce

correct classification. To overcome this problem, we decide to try the time warping recurrent neural network (TWRNN) as described in the following.

A. Time Warping Recurrent Neural Network: Derivations And Properties

Suppose we have a dynamic system with input vector $\mathbf{x}(t)$ and state vector $\mathbf{s}(t)$. Its state values at time t are determined by a function $\mathbf{F}(\cdot)$ as

$$\frac{d\mathbf{s}(t)}{dt} = \mathbf{F}(\mathbf{s}(t), \mathbf{x}(t), \frac{d\mathbf{x}(t)}{dt}) \qquad \text{(VI-1)}$$

After receiving all input vectors, it arrives the final state $\mathbf{s}(T)$,

$$\mathbf{s}(T) = \mathbf{s}(0) + \int_0^T \mathbf{F}(\mathbf{s}(t), \mathbf{x}(t), \frac{d\mathbf{x}(t)}{dt})dt \qquad \text{(VI-2)}$$

where T is the length of time duration. For example, $\mathbf{x}(t)$ in our studies are the feature vectors extracted from frames which are taken at a fixed rate from continuous speech signals. After the last vector of a sequence sample is received, we will compare target values to some or all state unit values, depending on whether $M_s>C$ or $M_s=C$, where M_s is the number of state units and C is the number of categories, and thus we treat these state values as outputs. Since different speakers speak at different and varying rates, changes of $\mathbf{x}(t)$ will be at different rates accordingly even for samples from the same category. Therefore, $\mathbf{s}(T)$ values may have large variances in both within category and between category cases, and these large variances may result poor classification. So, here we want to have a time warping invariant system to suppress these rate differences by treating equal and small changes in its inputs as representing equal and small intervals of its warped time.

Let an input vector, $\xi(\tau)$, be generated at a speed that is different from $\mathbf{x}(t)$ in time scale τ. Then the corresponding state vector, $\eta(\tau)$, will be

$$\frac{d\eta(\tau)}{d\tau} = \mathbf{F}(\eta(\tau), \xi(\tau), \frac{d\xi(\tau)}{d\tau}) \qquad \text{(VI-3)}$$

If $\xi(\tau)$ and $\mathbf{x}(t)$ belong to the same category and the time warping function is

$$\tau = \omega(t) \qquad \text{(VI-4)}$$

then ideally, for a time warping invariant system, we should have

$$\xi(\tau) = \xi(\omega(t)) = \mathbf{x}(t) \qquad \text{(VI-5)}$$

$$\eta(\tau) = \mathbf{s}(t) \qquad \text{(VI-6)}$$

Therefore, from (VI-3), (VI-5), and (VI-6),

$$\frac{d\eta(\tau)}{d\tau} = \mathbf{F}(\mathbf{s}(t), \mathbf{x}(t), \frac{d\mathbf{x}(t)}{d\tau})$$

$$= \mathbf{F}(\mathbf{s}(t), \mathbf{x}(t), \frac{d\mathbf{x}(t)}{dt}\frac{dt}{d\tau}) \qquad \text{(VI-7)}$$

Since $d\mathbf{s}(t) = d\eta(\tau)$, then by using (VI-1) and (VI-7), we will have

$$\mathbf{F}(\mathbf{s}(t), \mathbf{x}(t), \frac{d\mathbf{x}(t)}{dt})\, dt = \mathbf{F}(\mathbf{s}(t), \mathbf{x}(t), \frac{d\mathbf{x}(t)}{dt}\frac{dt}{d\tau})\, d\tau \qquad \text{(VI-8)}$$

To satisfy equation (VI-8), one solution is

$$\mathbf{F}(\mathbf{s}(t), \mathbf{x}(t), \frac{d\mathbf{x}(t)}{dt}) = \mathbf{F}'(\mathbf{s}(t), \mathbf{x}(t))\frac{d\mathbf{x}(t)}{dt} \qquad \text{(VI-9)}$$

where $\mathbf{F}'(\cdot)$ is a matrix function implemented by a neural network, and this neural network may be chosen differently according to the specified problem. Therefore, a general form for such a dynamic system that can handle time warping sequences may be written as

$$\frac{d\mathbf{s}(t)}{dt} = \mathbf{F}'(\mathbf{s}(t), \mathbf{x}(t))\frac{d\mathbf{x}(t)}{dt}$$

or

$$ds(t) = F'(s(t), x(t))\, dx(t) \qquad \text{(VI-10)}$$

After receiving the whole input sequence, we will have

$$s(T) = s(0) + \int_{x(0)}^{x(T)} F'(s(t), x(t))\, dx(t) \qquad \text{(VI-11)}$$

It is important to notice here that an integral along time t in (VI-2) is now changed to an integral along input trajectory x in (VI-11).

In discrete implementation of (VI-10) with a TWRNN, we may have:

$$s(t+1) = s(t) + F'(s(t), x(t))\,(x(t+1) - x(t)) \qquad \text{(VI-12)}$$

Without losing its properties, we make our TWRNN simpler by replacing its original multidimensional vector distance, $(x(t+1)-x(t))$, with its absolute vector distance value, $\delta L(t) = \|x(t+1)-x(t)\|$. Then (VI-12) becomes

$$s(t+1) = s(t) + F'(s(t), x(t)) * \delta L(t) \qquad \text{(VI-13)}$$

and $F'(\cdot)$ stands for a vector now. Our pilot studies show no significant differences between (VI-12) and (VI-13). In our studies, a three layer nonlinear neural network is used to implement $F'(\cdot)$. Figure 9 graphically shows its architecture. The activation function to calculate values for hidden layer units is again a sigmoid function, and at time t,

$$y_i(t) = f(H_i(t)) = 1/(1+\exp(-H_i(t))) \qquad \text{(VI-14)}$$

where

$$H_i(t) = \sum_j^{M_{in}} w1_{ij} * x_j(t) + \sum_j^{M_s} w2_{ij} * s_j(t) + w1_{i,bias} \qquad \text{(VI-15)}$$

M_{in} is the number of units in input layer, and $w1_{i,bias}$ is the bias term. The activation function between the third layer and hidden layer is a linear one,

$$z_i(t) = \sum_j^{M_h} w3_{ij} * y_j(t) \qquad \text{(VI-16)}$$

where M_h is the number of units in hidden layer. Finally, the state units may be calculated as

$$s_i(t+1) = s_i(t) + z_i(t)\,\delta L(t) \qquad \text{(VI-17)}$$

In our work, we set the number of state units equal to the number of categories, i.e., M_s=C.

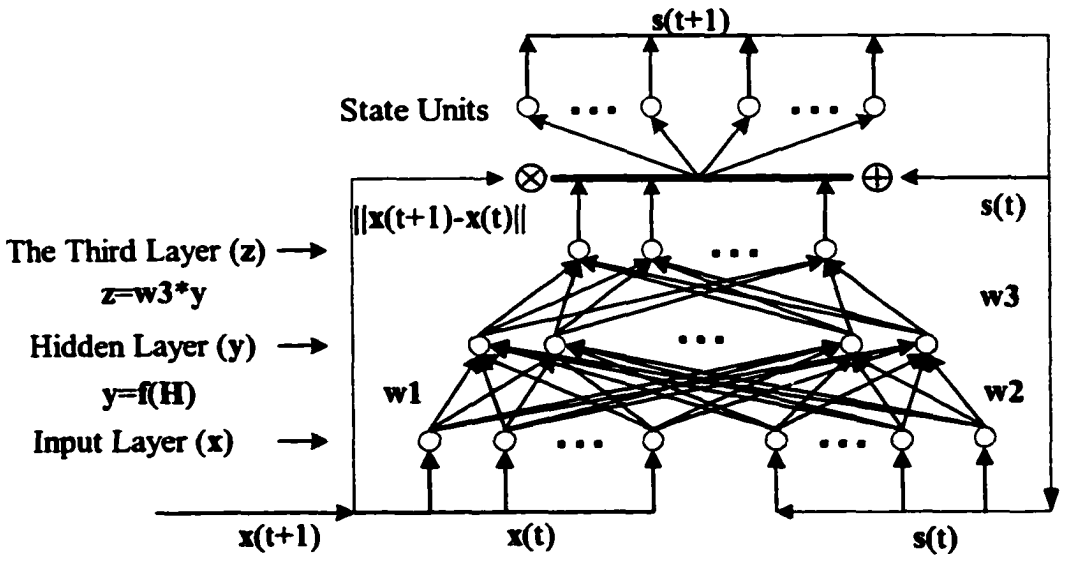


Figure 9: The Architecture of Time Warping Recurrent Neural Network with Nonlinear Structures.

As a study in early stage, here we follow the conventional minimum square error objective approach for cost function definition. If the target value for a sequence sample to category c is ζ_c, then we define our cost function as

$$E = \sum_q^N E_q = \sum_q^N \sum_c^C \tfrac{1}{2} (s_c^q(T) - \zeta_c)^2 \qquad \text{(VI-18)}$$

where N is the total number of samples, $s_c^q(T)$ is the final output of TWRNN. The update rules for network weights are derived in Appendix J.

B. Studies with TIMIT Data

To see if our TWRNN is able to capture temporal information, we test it with diphthong samples. Among all vowels, diphthongs are believed to contain more distinguishable temporal information than monothongs since their sounds change much more over their durations. From TIMIT SX training data set, we randomly select 360 /EY/ samples. Their waveform segments are taken sequentially into 20ms frames with 10ms overlap. Each frame is then processed the same way as we have done above to extract mel feature vectors and MFCC feature vectors. All feature values are then linearly rescaled around −1 to 1. For our first study, each /EY/ sample is a series of mel feature frames with different length, ranging from 3 frames to 30 frames. Since we want TWRNN to make its classifications based on temporal information, we reverse the frame order of each /EY/ sample to form the second category. Therefore, the only difference between the original category and the new category is their frame orders. For the total 720 in-order and reversed-order /EY/ samples, we further randomly partition them into a training set and a validation set with the constraint that each category has the same number of samples in each set. There are 600 samples in training set and 120 samples in validation set. Total 6 such partitions are created. Similarly, we build our testing set from the 63 /EY/ samples of core testing set, which gives total 126 samples. For each partition, TWRNN is initialized with 12 different random weights, one for each trial. The learning rate is selected as 0.002 and stopping criterion is the cost function value stopping

criterion. The training of network will stop if $t-t_{min}=5,000$, where t_{min} is the epoch at which we get the minimum cost function value in the first t epochs. The result of each partition is chosen from the trial that has the lowest cost function value. Final results are then the averages of the results from these 6 partitions. Table 19 shows the results. With basically only temporal information, our TWRNN gives about 80% classification rate on testing set. This result is very encouraging. As a reference, we also use the first 10 MFCC features to train TWRNN with the same partitions under the same training procedure. The results are in Table 19, which gives about 82% classification rate on testing set.

Table 19
Percentages of Correct Classifications with Time Warping Recurrent Neural Networks on /EY/ Samples And Their Reversed Order Counterparts in TIMIT SX Data. (A Network Structure [M_{in}, M_h, M_s] Stands for A TWRNN with M_{in} Input Units, M_h Hidden Units, And M_s State Units)

Network Structures	Features Used	Correct Classifications		
		Training Data Set	Validation Data Set	Testing Data Set
[30, 2, 2]	mel Features	83.7	83.7	80.6
[10, 2, 2]	MFCC Features	82.2	83.2	82.0
[5, 2, 2]	Linear Euc-NN Processed Features	82.6	84.6	81.9
[5, 2, 2]	Nonlinear Euc-NN Processed Features	71.1	71.0	70.8

Then next, we study with LENN features. We use one of trained linear Euc-NNs in Table 13 to generate LENN feature vectors for each /EY/ and reversed /EY/ samples. All samples in training, validation and testing sets are kept unchanged, and the same training procedure is applied. Results are shown in Table 19. The classification rate for each data set is comparable with the results from mel features and MFCC features. This indicates that our LENN feature vectors can at least keep, or might be able to enhance,

discriminative temporal information of mel spectra when using 5 elements. With these low-dimensional features, computation cost will be dramatically reduced.

In nonlinear case, we use one of the trained Euc-NNs in Table 17 to generate NENN features. Following the same way as with LENN features, we obtain the results in Table 19. In this nonlinear case, classification rates are greatly decreased. On testing set, it drops from 80.6% to 70.8%. This suggests that nonlinear transformation in this Euc-NN can distort or average out discriminative temporal information of mel spectra. Therefore, NENN features may not be good for temporal information related processing, instead, LENN feature set is a better choice.

VII. Summary

We have automatically segmented speech signals of audio files in our database with satisfactory resolutions. This not only greatly increases efficiencies for waveform manipulations but also eliminates possible human errors if processed manually. New hybrid models comprised of neural networks and Euclidean distance are proposed. These models make use of the important adaptive feature of neural networks and are mainly for feature extraction and dimensionality reduction from the perspective of classification. Especially with linear and nonlinear Euclidean neural networks, the number of dimensions of extracted feature vectors can be significantly lower than classical MFCCs while the classification performance is approximately unchanged, or even better. Results on both our data and TIMIT data suggest that Euc-NN generated features may be better than the standard MFCC features when one considers the number of elements required for classifications and the corresponding classification performance in phoneme recognition. Time warping problem in speech signals is also studied. Our time warping recurrent neural network suggests promising capability in capturing temporal trajectory information using mel features, MFCCs and Euc-NN features, which is critical for further improvement of speech recognition.

VIII. Appendixes

A. mel Scale

One widely used scale to represent human perception of the frequency content of sounds in speech feature extraction is so called "mel" scale [1]. For mel scale, the pitch of a 1kHz tone, 40dB above the perceptual hearing threshold, is first defined as 1000mels. Other subjective pitch values of tones are obtained by adjusting the actual frequency of a tone such that it is half or twice the perceived pitch of a reference tone with a known mel frequency. The obtained relationship between original frequency and mel frequency may be described like this: For frequencies below about 1,000Hz, mel frequencies approximately follow a linear scale; For frequencies above 1,000Hz, mel frequencies approximately have the logarithmic frequency scale. One of the analysis methods on the spectra in this warped frequency scale is the filter-bank spectrum model. Figure 11 schematically shows filters used for MFCCs [13].

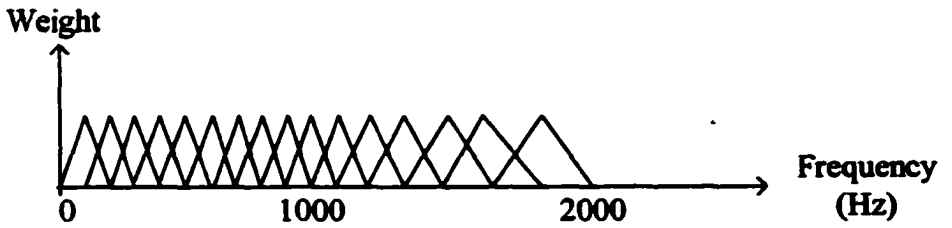


Figure 10: Filters used for generating mel-frequency cepstrum coefficients.

B. Token Word Groups Used in The Database of Professor Antrobus' Lab

Group a: Dug Bit Peet Fet Soot Dot Fit Seet Get Moot Not Dug

Group b: Dug Beet Pit Set Doot Got Feet Pet Goot Mot Nit Dug

Group c: Dug Bet Dit Meet Poot Sot Git Deet Noot Fot Net Dug

Group d: Dug Boot Mit Det Bot Geet Sit Foot Pot Met Neet Dug

Group e: Dug Beet Fit Moot Pet Dot Pet Got Seet Pit Noot Dug

Group f: Dug Bit Sot Peet Doot Set Dit Feet Boot Det Not Dug

Group g: Dug Bot Sit Met Foot Neet Git Mot Goot Meet Net Dug

Group h: Dug Bet Poot Deet Fot Mit Geet Pot Soot Get Nit Dug

C. Hamming Window

For each frame taken from waveforms with fixed size, generally speaking, there are always signal discontinuities at its beginning and end. Such discontinuities will distort discrete Fourier transform in the next processing step [1,20,21]. To minimize this effect, people usually taper the signals by windowing each individual frame so that

$$y(n)=x(n)w(n)$$

where x(n) is the n^{th} waveform signal in a frame, w(n) is the window, and y(n) is the output of windowing. One widely used window is the Hamming window, which has the form

$$w(n)=0.54-0.46\cos\left(\frac{2\pi n}{N-1}\right) \qquad \text{(C-1)}$$

where $0\le n\le N-1$, and N is the total number of signal values in a frame.

D. Euclidean Classifier And Diagonal Baysian Maximum Likelihood Classifier

In statistical pattern recognition, a sample is generally described as a random n-dimensional vector,

$$\mathbf{x}=(x_1,x_2,\ldots,x_n)^T \qquad \text{(D-1)}$$

One way to characterize its distribution is to use conditional probability density function of category c,

$$p(\mathbf{x}|c) \qquad \text{(D-2)}$$

The unconditional density function is then given by

$$p(\mathbf{x})=\Sigma_c^C P_c p(\mathbf{x}|c) \qquad \text{(D-3)}$$

where C is the total number of categories, and P_c is *a priori* probability of category c. According to Bayes theorem, the *a posteriori* probability for a given vector x belonging to category c is

$$p(c|\mathbf{x})=\frac{P_c p(x\,|\,c)}{p(x)} \qquad \text{(D-4)}$$

Then, the Bayes decision rule for minimum error will classify x as in category ω if and only if

$$\omega=\text{argmax}(p(c|x)) \qquad \text{(D-5)}$$

or,

$$\omega=\text{argmax}(P_c p(\mathbf{x}|c)) \qquad \text{(D-6)}$$

since p(x) is the same for each category [7].

However, it is very hard to get accurate p(x|c) in most realistic problems. As in many cases, we then assume normal distribution for each category here, i.e.,

$$p(\mathbf{x}|c)=\frac{1}{(2\pi)^{\frac{n}{2}}\,|\Sigma_c|^{\frac{1}{2}}}\exp\left(-\tfrac{1}{2}(\mathbf{x}-\mathbf{m}_c)^T\Sigma_c^{-1}(\mathbf{x}-\mathbf{m}_c)\right) \qquad \text{(D-7)}$$

where $\mathbf{m}_c$ is the mean of vectors of category c,

$$\mathbf{m}_c = \int p(\mathbf{x}|c)\mathbf{x}d\mathbf{x} \tag{D-8}$$

Σ_c is the covariance matrix of category c,

$$(\Sigma_c)_{ij} = \int (x_i-(m_c)_i)(x_j-(m_c)_j)dx_i dx_j \tag{D-9}$$

and $\mathbf{x}\in c$.

If we take minus logarithmic on (D-6) and use (D-7), we will have that for any given vector $\mathbf{x}$, $\mathbf{x}\in\omega$ if and only if

$$\omega = \text{argmin}\{-\ln P_c + \frac{2}{n}\ln(2\pi) + \frac{1}{2}\ln|\Sigma_c| + \frac{1}{2}(\mathbf{x}-\mathbf{m}_c)^T\Sigma_c^{-1}(\mathbf{x}-\mathbf{m}_c)\}$$

$$= \text{argmin}\{\ln|\Sigma_c| + (\mathbf{x}-\mathbf{m}_c)^T\Sigma_c^{-1}(\mathbf{x}-\mathbf{m}_c) - 2\ln P_c\} \tag{D-10}$$

This is so called Baysian maximum likelihood classifier. If we assume that only the diagonal elements σ_{kk} in each covariance matrix is not zero, we may simplify (D-10) to a diagonal Baysian maximum likelihood classifier, where $\mathbf{x}\in\omega$ if and only if

$$\omega = \text{argmin}\{\ln(\prod_{k=1}^{n}\sigma_{kk}^c) + \Sigma_k^n(\sigma_{kk}^c)^{-1}(\mathbf{x}-\mathbf{m}_c)_{kk}(\mathbf{x}-\mathbf{m}_c)_{kk} - 2\ln P_c\} \tag{D-11}$$

If we further assume that all σ_{kk} are equal to 1, and all *a priori* possibilities for each category are also equal, then we will have Euclidean classifier, where $\mathbf{x}\in\omega$ if and only if

$$\omega = \text{argmin}\{\Sigma_k^n(\mathbf{x}-\mathbf{m}_c)_{kk}(\mathbf{x}-\mathbf{m}_c)_{kk}\} \tag{D-12}$$

From statistical pattern recognition point of view, the mean vector $\mathbf{m}_c$ of Euclidean classifier is actually a reference pattern of category c, with which each given vector is compared. Such a reference pattern is also called a template, or a prototype. Different classifiers may have different algorithms to obtain their templates. And, these

templates will be critical to a classifier's performance. As one may know, however, some classifiers do not use templates and the estimates of likelihood are calculated differently.

E. Leave-One-Out Method

When a finite number of samples is used and the performance of a classifier is to be estimated, one needs to determine which samples will be used to train the classifier and which samples will be used to test the classifier. Since the true probability density function for each category is unknown, one can not achieve the optimal Bayes error with these samples. However, we can use the following methods to estimate the lower and upper bound of the Bayes error [7].

(1) Resubstitution (R) method: Use all samples to train and test a classifier. This gives the lower bound of the Bayes error.

(2) Holdout (H) method: Use two independent sample sets, one for classifier training and the other for classifier testing. This gives the upper bound of the Bayes error.

In holdout method, if we partition a given sample set into two, we must implement a proper algorithm to allocate samples in each set and a proper dividing algorithm to assure that the distributions of samples in the two sets are very close. One procedure, called leave-one-out (L) method, helps for this implementation. In this method, one example is excluded for testing and the rest are for training. Such operation will repeat until every sample has been used for testing once. Then, the number of misclassified samples is counted to obtain the performance of the classifier.

F. Update Rules for The Simple Linear Hybrid Model

By differentiating cost function (IV-6), we have

$$\frac{\partial E}{\partial w_{ij}} = \sum_q^N \frac{\partial E_q}{\partial w_{ij}}$$

$$= \sum_q^N \sum_c^C \left(\frac{do(q,c)}{S(q)} - \zeta_c\right)\left(\frac{\partial do(q,c)}{\partial w_{ij}} \frac{1}{S(q)} - \frac{do(q,c)}{S(q)^2} \frac{\partial S(q)}{\partial w_{ij}}\right)$$

(F-1)

Using function (IV-3) and (IV-4),

$$\frac{\partial do(q,c)}{\partial w_{ij}} = 2\sum_k^{M_{out}} (o_k^q - O_k^c)\left(\frac{\partial o_k^q}{\partial w_{ij}} - \frac{1}{N_c}\sum_p^{N_c} \frac{\partial o_k^p}{\partial w_{ij}}\right) \qquad \text{(F-2)}$$

From the definition of S(q) in (IV-6),

$$\frac{\partial S(q)}{\partial w_{ij}} = \sum_{c'}^C \frac{\partial do(q,c')}{\partial w_{ij}} = 2\sum_{c'}^C \sum_k^{M_{out}} (o_k^q - O_k^{c'})\left(\frac{\partial o_k^q}{\partial w_{ij}} - \frac{1}{N_{c'}}\sum_p^{N_{c'}} \frac{\partial o_k^p}{\partial w_{ij}}\right)$$

(F-3)

If we assume x^q has one more element with constant value 1 for bias term, then from (IV-5) we have

$$\frac{\partial o_k^q}{\partial w_{ij}} = \sum_s^{M_{in}} x_s^q \, \delta_{ki} \, \delta_{sj} = x_j^q \, \delta_{ki} \qquad \text{(F-4)}$$

where δ denotes the Kronecher delta function.

By combining equations (F-1), (F-2), (F-3) and (F-4), we obtain

$$\frac{\partial E}{\partial w_{ij}} = \sum_q^N \sum_c^C 2\left(\frac{do(q,c)}{S(q)} - \zeta_c\right)\left\{\sum_k^{M_{out}} (o_k^q - O_k^c)\left(x_j^q \delta_{ki} - \frac{1}{N_c}\sum_p^{N_c} x_j^p \delta_{ki}\right)\right.$$

$$\left. \frac{1}{S(q)} - \frac{do(q,c)}{S(q)^2} \sum_{c'}^C \sum_k^{M_{out}} (o_k^q - O_k^{c'})\left(x_j^q \delta_{ki} - \frac{1}{N_{c'}}\sum_p^{N_{c'}} x_j^p \delta_{ki}\right)\right\}$$

(F-5)

Applying conventional gradient decent method for neural networks [3,4], we have

$$\Delta w_{ij}(t) = -\lambda \frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(t-1) \qquad \text{(F-6)}$$

where t stands for weight updates step, λ is the learning rate, α is the momentum coefficient, and $\alpha\Delta w_{ij}(t-1)$ is the momentum term.

For incremental training mode [3,4], the summation on q in (F-5) may be taken away.

G. Update Rules for The Simple Nonlinear Hybrid Model

By differentiating cost function (IV-6), we have

$$\frac{\partial E}{\partial w_{ij}} = \sum_q^N \frac{\partial E_q}{\partial w_{ij}}$$

$$= \sum_q^N \sum_c^C \left(\frac{do(q,c)}{S(q)} - \zeta_c\right)\left(\frac{\partial do(q,c)}{\partial w_{ij}} \frac{1}{S(q)} - \frac{do(q,c)}{S(q)^2} \frac{\partial S(q)}{\partial w_{ij}}\right)$$

(G-1)

Using function (IV-3) and (IV-4),

$$\frac{\partial do(q,c)}{\partial w_{ij}} = 2\sum_k^M (o_k^q - O_k^c)\left(\frac{\partial o_k^q}{\partial w_{ij}} - \frac{1}{N_c}\sum_p^{N_c} \frac{\partial o_k^p}{\partial w_{ij}}\right) \qquad \text{(G-2)}$$

From the definition of S(q) in (IV-6),

$$\frac{\partial S(q)}{\partial w_{ij}} = \sum_{c'}^C \frac{\partial do(q,c')}{\partial w_{ij}} = 2\sum_{c'}^C \sum_k^M (o_k^q - O_k^{c'})\left(\frac{\partial o_k^q}{\partial w_{ij}} - \frac{1}{N_{c'}}\sum_p^{N_{c'}} \frac{\partial o_k^p}{\partial w_{ij}}\right)$$

(G-3)

Let's first consider the weights between output layer and hidden layer, w2. If we assume y^q has one more element with constant value 1 for bias term, by taking derivatives on (IV-8) we have

$$\frac{\partial o_k^q}{\partial w2_{ij}} = \sum_u^{M_h} y_u^q \, \delta_{ki} \, \delta_{uj} = y_j^q \, \delta_{ki} \qquad \text{(G-4)}$$

where δ denotes the Kronecher delta function.

By combining equations (G-1), (G-2), (G-3) and (G-4), we obtain

$$\frac{\partial E}{\partial w2_{ij}} = \sum_q^N \sum_c^C 2\left(\frac{do(q,c)}{S(q)} - \zeta_c\right)\left\{\sum_k^{M_{out}} (o_k^q - O_k^c)\left(y_j^q \, \delta_{ki} - \frac{1}{N_c} \sum_p^{N_c} y_j^p \, \delta_{ki}\right)\right.$$

$$\left. \frac{1}{S(q)} - \frac{do(q,c)}{S(q)^2} \sum_{c'}^C \sum_k^{M_{out}} (o_k^q - O_k^{c'})\left(y_j^q \, \delta_{ki} - \frac{1}{N_{c'}} \sum_p^{N_{c'}} y_j^p \, \delta_{ki}\right)\right\}$$

(G-5)

And we again use equation (F-6) for w2 update.

Now let's consider the weights between hidden layer and input layer, w1. If we assume x^q has one more element with constant value 1 for bias term and use the chain rule on (IV-7) and (IV-8), we have

$$\frac{\partial o_k^q}{\partial w1_{ij}} = \sum_u^{M_h} w2_{ku} \frac{\partial y_u^q}{\partial w1_{ij}}$$

$$= \sum_u^{M_h} w2_{ku} f'(H_u) \sum_v^{M_{in}} x_v^q \, \delta_{ui} \, \delta_{vj}$$

$$= w2_{ki} \, f'(H_i) \, x_j^q \qquad \text{(G-6)}$$

By combining equations (G-1), (G-2), (G-3) and (G-7), we obtain

$$\frac{\partial E}{\partial w1_{ij}} = \sum_q^N \sum_c^C 2\left(\frac{do(q,c)}{S(q)} - \zeta_c\right)$$

$$\{\sum_k^{M_{out}}(o_k^q - O_k^c)(w2_{ki}f'(H_i)x_j^q - \frac{1}{N_c}\sum_p^{N_c} w2_{ki}\, f'(H_i)x_j^p)\frac{1}{S(q)} -$$

$$\frac{do(q,c)}{S(q)^2}\sum_{c'}^{C}\sum_k^{M_{out}}(o_k^q - O_k^{c'})(w2_{ki}f'(H_i)x_j^q - \frac{1}{N_{c'}}\sum_p^{N_{c'}} w2_{ki}f'(H_i)x_j^p)\}$$

(G-7)

And we again use equation (F-6) for **w1** update.

For incremental mode, the summations on q in (G-5) and (G-7) may be taken away.

H. Update Rules for Linear Euclidean Neural Network

By differentiating cost function (IV-12), we have

$$\frac{\partial E}{\partial w_{ij}} = \sum_q^N \frac{\partial E_q}{\partial w_{ij}}$$

$$= \sum_q^N a\, E_q(1 - E_q)\frac{\partial d^q}{\partial w_{ij}} \qquad \text{(H-1)}$$

From the definition of d^q in function (IV-12),

$$\frac{\partial d^q}{\partial w_{ij}} = \frac{\partial o_t^q}{\partial w_{ij}} - \frac{\partial o_r^q}{\partial w_{ij}} \qquad \text{(H-2)}$$

Let's first consider the weights between output layer and hidden layer, **w2**. By taking derivatives on (IV-10) we have

$$\frac{\partial o_t^q}{\partial w2_{ij}} = -2\sum_u^{M_h}(y_u^q - w2_{tu})\delta_{ti}\delta_{uj} = -2(y_j^q - w2_{tj})\delta_{ti} \qquad \text{(H-3)}$$

and

$$\frac{\partial o_r^q}{\partial w2_{ij}} = -2\sum_u^{M_h}(y_u^q - w2_{ru})\delta_{ri}\delta_{uj} = -2(y_j^q - w2_{rj})\delta_{ri} \qquad \text{(H-4)}$$

where δ denotes the Kronecher delta function.

By combining equations (H-1), (H-2), (H-3) and (H-4), we obtain

$$\frac{\partial E}{\partial w2_{ij}} = \sum_q^N a\, E_q(1- E_q)\,(2(y_j^q - w2_{rj})\delta_{ri} - 2(y_j^q - w2_{tj})\delta_{ti}) \qquad \text{(H-5)}$$

And we again use equation (F-6) for w2 update.

Now let's consider the weights between hidden layer and input layer, w1. If we assume x^q has one more element with constant value 1 for bias term and use the chain rule on (IV-10) and (IV-9), we have

$$\frac{\partial o_t^q}{\partial w1_{ij}} = 2\sum_u^{M_h}(y_u^q - w2_{tu})\frac{\partial y_u^q}{\partial w1_{ij}}$$

$$= 2\sum_u^{M_h}(y_u^q - w2_{tu})\sum_v^{M_{in}} x_v^q\,\delta_{ui}\,\delta_{vj}$$

$$= 2(y_i^q - w2_{ti})\,x_j^q \qquad \text{(H-6)}$$

and

$$\frac{\partial o_r^q}{\partial w1_{ij}} = 2\sum_u^{M_h}(y_u^q - w2_{ru})\frac{\partial y_u^q}{\partial w1_{ij}}$$

$$= 2\sum_u^{M_h}(y_u^q - w2_{ru})\sum_v^{M_{in}} x_v^q\,\delta_{ui}\,\delta_{vj}$$

$$= 2(y_i^q - w2_{ri})\,x_j^q \qquad \text{(H-7)}$$

By combining equations (H-1), (H-2), (H-7) and (H-8), we obtain

$$\frac{\partial E}{\partial w1_{ij}} = \sum_q^N a\, E_q(1- E_q)\,(2(y_i^q - w2_{ti})\,x_j^q - 2(y_i^q - w2_{ri})\,x_j^q) \qquad \text{(H-8)}$$

And we again use equation (F-6) for w1 update.

For incremental mode, the summations on q in (H-5) and (H-8) may be taken away.

I. Update Rules for Nonlinear Euclidean Neural Network

By differentiating cost function (IV-12), we have

$$\frac{\partial E}{\partial w_{ij}} = \sum_q^N \frac{\partial E_q}{\partial w_{ij}}$$

$$= \sum_q^N a\, E_q(1- E_q)\frac{\partial d^q}{\partial w_{ij}} \qquad \text{(I-1)}$$

From the definition of d^q in function (IV-12),

$$\frac{\partial d^q}{\partial w_{ij}} = \frac{\partial o_t^q}{\partial w_{ij}} - \frac{\partial o_r^q}{\partial w_{ij}} \qquad \text{(I-2)}$$

Let's first consider the weights between output layer and second hidden layer, **w3**.

By taking derivatives on (IV-15) we have

$$\frac{\partial o_t^q}{\partial w3_{ij}} = -2\sum_u^{M_{h2}} (z_u^q - w3_{tu})\delta_{ti}\delta_{uj} = -2(z_j^q - w3_{tj})\delta_{ti} \qquad \text{(I-3)}$$

and

$$\frac{\partial o_r^q}{\partial w3_{ij}} = -2\sum_u^{M_{h2}} (z_u^q - w3_{ru})\delta_{ri}\delta_{uj} = -2(z_j^q - w3_{rj})\delta_{ri} \qquad \text{(I-4)}$$

where δ denotes the Kronecher delta function.

By combining equations (I-1), (I-2), (I-3) and (I-4), we obtain

$$\frac{\partial E}{\partial w3_{ij}} = \sum_q^N a\, E_q(1- E_q)\,(2(z_j^q - w3_{rj})\delta_{ri} - 2(z_j^q - w3_{tj})\delta_{ti}) \qquad \text{(I-5)}$$

And we again use equation (F-6) for **w3** update.

Now let's consider the weights between second hidden layer and first hidden layer, **w2**. If we assume y^q has one more element with constant value 1 for bias term and use the chain rule on (IV-15) and (IV-14), we have

$$\frac{\partial o_t^q}{\partial w2_{ij}} = 2\sum_u^{M_{h2}} (z_u^q - w3_{tu}) \frac{\partial z_u^q}{\partial w2_{ij}}$$

$$= 2\sum_u^{M_{h2}} (z_u^q - w3_{tu}) \sum_v^{M_{h1}} y_v^q \delta_{ui}\delta_{vj}$$

$$= 2(z_i^q - w3_{ti})\, y_j^q \qquad \text{(I-6)}$$

and

$$\frac{\partial o_t^q}{\partial w2_{ij}} = 2\sum_u^{M_{h2}} (z_u^q - w3_{ru}) \frac{\partial z_u^q}{\partial w2_{ij}}$$

$$= 2\sum_u^{M_{h2}} (z_u^q - w3_{ru}) \sum_v^{M_{h1}} y_v^q \delta_{ui}\delta_{vj}$$

$$= 2(z_i^q - w3_{ri})\, y_j^q \qquad \text{(I-7)}$$

By combining equations (I-1), (I-2), (I-7) and (I-8), we obtain

$$\frac{\partial E}{\partial w2_{ij}} = \sum_q^N a\, E_q(1 - E_q)\,(2(z_i^q - w3_{ti})\, y_j^q - 2(z_i^q - w3_{ri})\, y_j^q) \qquad \text{(I-8)}$$

And we again use equation (F-6) for **w2** update.

At last, let's consider the weights between first hidden layer and input layer, **w1**. If we assume x^q has one more element with constant value 1 for bias term and use the chain rule on (IV-15), (IV-14) and (IV-13), we have

$$\frac{\partial o_t^q}{\partial w1_{ij}} = 2\sum_u^{M_{h2}} (z_u^q - w3_{tu}) \sum_v^{M_{h1}} w2_{uv} \frac{\partial y_v^q}{\partial w1_{ij}}$$

$$= 2\sum_u^{M_{h2}} (z_u^q - w3_{tu}) \sum_v^{M_{h1}} w2_{uv}\, f'(H_v) \sum_l^{M_{in}} x_l^q \delta_{vi}\delta_{lj}$$

$$= 2\sum_u^{M_{h2}} (z_u^q - w3_{ru})\, w2_{ui} f'(H_i)\, x_j^q \qquad \text{(I-9)}$$

and

$$\frac{\partial o_r^q}{\partial w1_{ij}} = 2\sum_u^{M_{h2}} (z_u^q - w3_{ru}) \sum_v^{M_{h1}} w2_{uv} \frac{\partial y_v^q}{\partial w1_{ij}}$$

$$= 2\sum_u^{M_{h2}} (z_u^q - w3_{ru}) \sum_v^{M_{h1}} w2_{uv}\, f'(H_v) \sum_l^{M_{in}} x_l^q \delta_{vi}\delta_{lj}$$

$$= 2\sum_u^{M_{h2}} (z_u^q - w3_{ru})\, w2_{ui} f'(H_i)\, x_j^q \qquad \text{(I-10)}$$

By combining equations (I-1), (I-2), (I-11) and (I-12), we obtain

$$\frac{\partial E}{\partial w1_{ij}} = \sum_q^N a\, E_q(1 - E_q) \left(2\sum_u^{M_{h2}} (z_u^q - w3_{ru})\, w2_{ui} f'(H_i)\, x_j^q - \right.$$

$$\left. 2\sum_u^{M_{h2}} (z_u^q - w3_{ru})\, w2_{ui} f'(H_i)\, x_j^q \right) \qquad \text{(I-11)}$$

And we again use equation (F-6) for **w1** update.

For incremental mode, the summations on q in (I-5), (I-8) and (I-11) may be taken away.

J. Update Rules for Time Warping Recurrent Neural Network

By differentiating cost function (VI-15), we have

$$\frac{\partial E}{\partial w_{ij}} = \sum_q^N \frac{\partial E_q}{\partial w_{ij}}$$

$$= \sum_q^N \sum_c^C (s_c^q(T) - \zeta_c) \frac{\partial s_c^q(T)}{\partial w_{ij}} \qquad \text{(J-1)}$$

Let's first consider the weights between the third layer and hidden layer, **w3**. By taking derivatives on (VI-17)

$$\frac{\partial s_c^q(T)}{\partial w3_{ij}} = \frac{\partial s_c^q(T-1)}{\partial w3_{ij}} + \frac{\partial z_c^q(T-1)}{\partial w3_{ij}} \delta L(T\text{-}1) \qquad \text{(J-2)}$$

From (VI-14), (VI-15) and (VI-16), we have

$$\frac{\partial z_c^q(T-1)}{\partial w3_{ij}} = \sum_u^{M_b} \left(y_u^q(T\text{-}1)\, \delta_{ci}\delta_{uj} + w3_{cu} \frac{\partial y_u^q(T-1)}{\partial w3_{ij}} \right)$$

$$= \sum_u^{M_b} \left(y_u^q(T\text{-}1)\, \delta_{ci}\delta_{uj} + w3_{cu}\, f'(H_u^q(T\text{-}1)) \sum_v^{M_s} w2_{uv} \frac{\partial s_v^q(T-1)}{\partial w3_{ij}} \right)$$

(J-3)

where δ denotes the Kronecher delta function. Therefore,

$$\frac{\partial s_c^q(T)}{\partial w3_{ij}} = \frac{\partial s_c^q(T-1)}{\partial w3_{ij}} + \sum_u^{M_b} \left(y_u^q(T\text{-}1)\, \delta_{ci}\delta_{uj} + \right.$$

$$\left. w3_{cu}\, f'(H_u^q(T\text{-}1)) \sum_v^{M_s} w2_{uv} \frac{\partial s_v^q(T-1)}{\partial w3_{ij}} \right) \delta L(T\text{-}1) \qquad \text{(J-4)}$$

Following the same way, we can get $\frac{\partial s_v^q(T-1)}{\partial w3_{ij}}$, $\frac{\partial s_v^q(T-2)}{\partial w3_{ij}}$, $\frac{\partial s_v^q(T-3)}{\partial w3_{ij}}$, ...,

until t=1. By properly selecting initial values for these terms, we can accumulate $\frac{\partial s_c^q(T)}{\partial w3_{ij}}$

values for a sequence sample at its each vector input. Then,

$$\frac{\partial E}{\partial w3_{ij}} = \sum_q^N \sum_c^C (s_c^q(T) - \zeta_c) \left(\frac{\partial s_c^q(T-1)}{\partial w3_{ij}} + \sum_u^{M_b} \left(y_u^q(T\text{-}1)\, \delta_{ci}\delta_{uj} + \right. \right.$$

$$\left. \left. w3_{cu}\, f'(H_u^q(T\text{-}1)) \sum_v^{M_s} w2_{uv} \frac{\partial s_v^q(T-1)}{\partial w3_{ij}} \right) \delta L(T\text{-}1) \right) \qquad \text{(J-5)}$$

And we again use equation (F-6) for w3 update.

Now let's consider the weights between the hidden layer and state units layer, **w2**. By taking derivatives on (VI-17)

$$\frac{\partial s_c^q(T)}{\partial w2_{ij}} = \frac{\partial s_c^q(T-1)}{\partial w2_{ij}} + \frac{\partial z_c^q(T-1)}{\partial w2_{ij}} \delta L(T\text{-}1) \qquad \text{(J-6)}$$

From (VI-14), (VI-15) and (VI-16), we have

$$\frac{\partial z_c^q(T-1)}{\partial w2_{ij}} = \sum_u^{M_h} w3_{cu} \frac{\partial y_u^q(T-1)}{\partial w2_{ij}}$$

$$= \sum_u^{M_h} w3_{cu}\, f'(H_u^q(T\text{-}1)) \sum_v^{M_s} (s_v^q\ \delta_{ui}\delta_{vj} + w2_{uv} \frac{\partial s_v^q(T-1)}{\partial w2_{ij}}) \qquad \text{(J-7)}$$

Following the same way, we can get $\frac{\partial s_v^q(T-1)}{\partial w2_{ij}}$, $\frac{\partial s_v^q(T-2)}{\partial w2_{ij}}$, $\frac{\partial s_v^q(T-3)}{\partial w2_{ij}}$, ..., until t=1. By properly selecting initial values for these terms, we can accumulate $\frac{\partial s_c^q(T)}{\partial w2_{ij}}$ values for a sequence sample at its each vector input. Then,

$$\frac{\partial E}{\partial w2_{ij}} = \sum_q^N \sum_c^C (s_c^q\ (T)\text{-}\ \zeta_c) (\frac{\partial s_c^q(T-1)}{\partial w2_{ij}} + \sum_u^{M_h} w3_{cu}\, f'(H_u^q(T\text{-}1))^*$$

$$\sum_v^{M_s} (s_v^q\ \delta_{ui}\delta_{vj} + w2_{uv} \frac{\partial s_v^q(T-1)}{\partial w2_{ij}}) \delta L(T\text{-}1)) \qquad \text{(J-8)}$$

And we again use equation (F-6) for **w2** update.

Finally, let's consider the weights between the hidden layer and input layer, **w1**. By taking derivatives on (VI-17)

$$\frac{\partial s_c^q(T)}{\partial w1_{ij}} = \frac{\partial s_c^q(T-1)}{\partial w1_{ij}} + \frac{\partial z_c^q(T-1)}{\partial w1_{ij}} \delta L(T\text{-}1) \qquad \text{(J-9)}$$

From (VI-14), (VI-15) and (VI-16), we have

$$\frac{\partial z_c^q(T-1)}{\partial w1_{ij}} = \sum_u^{M_h} w3_{cu} \frac{\partial y_u^q(T-1)}{\partial w1_{ij}}$$

$$= \sum_u^{M_h} w3_{cu} f'(H_u^q(T\text{-}1))(\sum_l^{M_{in}} x_l^q(T\text{-}1)\delta_{ui}\delta_{lj} + \sum_v^{M_s} w2_{uv} \frac{\partial s_v^q(T-1)}{\partial w1_{ij}})$$

(J-10)

Following the same way, we can get $\frac{\partial s_v^q(T-1)}{\partial w1_{ij}}$, $\frac{\partial s_v^q(T-2)}{\partial w1_{ij}}$, $\frac{\partial s_v^q(T-3)}{\partial w1_{ij}}$, ..., until t=1. By properly selecting initial values for these terms, we can accumulate $\frac{\partial s_c^q(T)}{\partial w1_{ij}}$ values for a sequence sample at its each vector input. Then,

$$\frac{\partial E}{\partial w1_{ij}} = \sum_q^N \sum_c^C (s_c^q(T) - \zeta_c)(\frac{\partial s_c^q(T-1)}{\partial w1_{ij}} + \sum_u^{M_h} w3_{cu} f'(H_u^q(T\text{-}1))*$$

$$(\sum_l^{M_{in}} x_l^q(T\text{-}1)\delta_{ui}\delta_{lj} + \sum_v^{M_s} w2_{uv} \frac{\partial s_v^q(T-1)}{\partial w1_{ij}})\delta L(T\text{-}1)) \quad (J\text{-}11)$$

And we again use equation (F-6) for **w1** update.

For incremental mode, the summations on q in (J-5), (J-8) and (J-11) may be taken away.

K. Vowels in TIMIT Corpus

The vowels in TIMIT corpus are defined as the follows. The asterisk besides a vowel label indicates a diphthong. The numbers are the percentages of vowels in TIMIT's SX training data set.

Vowel Label	Percentages In SX set	Sample Word	Vowel Label	Percentages In SX set	Sample Word
/IY/	0.13992	beet	/AH/	0.06744	but
/IH/	0.12766	bit	/AO/	0.06117	bought
/EH/	0.09555	bet	/OY/*	0.01227	boy
/EY/*	0.07233	bait	/OW/	0.04969	boat
/AE/	0.07444	bat	/UH/	0.01379	book
/AA/	0.07728	bott	/UW/	0.01800	boot
/AW/*	0.02253	bout	/UX/	0.05033	toot
/AY/*	0.06307	bite	/ER/	0.05454	bird

IX. References

[1] Lawrence R. Rabiner, Biing-Hwang Juang *Fundamentals of Speech Recognition* Prentice Hall PTR, NJ 1993

[2] National Academy of Sciences *Voice Communication Between Humans And Machines* National Academy Press, Washington D.C., 1994

[3] Mohamad H. Hassoun *Fundamentals of Artificial Neural Networks* The MIT Press, MA 1995

[4] John Hertz, Anders Krogh, Richard G. Palmer *Introduction to The Theory of Neural Computation* Addison Wesley Publishing Company, CA 1991

[5] Yoshua Bengio *Neural Networks for Speech and Sequence Recognition* International Thompson Computer Press, MA 1996

[6] Shigeru Katagiri (Editor) *Handbook of Neural Networks for Speech Processing* Artech House, Inc., MA 2000

[7] Keinosuke Fukunaga *Introduction to Statistical Pattern Recognition (Second Edition)* Academic Press, CA 1990

[8] William H. Press, Saul A. Teukolsky, William T. Vettering, Brian P. Flannery *Numerical Recipes in C* Cambridge University Press, (2nd Edition), UK 1992

[9] Biing-Hwang Juang (Editor) *"The Past, Present, and Future of Speech Processing"* IEEE Signal Processing Magazine, May 1998

[10] Richard Comerford *"The Voice of The Computer Is Heard in The Land (And It Listens Too)"* IEEE Spectrum, pp39-47, Dec. 1997

[11] Ta-Hsin Li, Jerry D. Gibson *"Speech Analysis And Segmentation by Parametric Filtering"* IEEE Trans. on Speech and Audio Processing, Vol. 4, No. 3, pp203-213, May 1996

[12] Thomas Kemp, Michael Schmidt, Martin Wsetphal, Alex Waibel *"Strategies for Automatic Segmentation of Audio Data"* IEEE ICASSP Vol. 3, pp1423-1426, 2000

[13] Steven B. Davis, Paul Mermelstein *"Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken*

Sentences" IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-28, No.4, pp357-366, Aug. 1980

[14] Montri Karnjanadecha, Stephen A. Zahorian *"Signal Modeling for High-Performance Robust Isolated Word Recognition"* IEEE Trans. on Speech and Audio Processing, Vol.9, No.6, pp647-654, Sept. 2001

[15] Paul McCourt, Saeed Vaseghi, Naomi Harte *"Multi-Resolution Cepstral Features for Phoneme Recognition Across Speech Sub-Bands"* ICASSP, Vol.1, pp557-560, 1998

[16] H. Bourlard, S. Dupont *"Sub-band Based Speech Recognition"* ICASSP, Vol.2, pp1251-1254, 1997

[17] Jont B. Allen *"How Do Humans Process and Recognize Speech"* IEEE Transaction on Speech and Audio Processing, Vol.2(4) pp567-577, October 1994

[18] G. Z. Sun, H. H. Chen, Y. C. Lee, Y. D. Liu *"Time Warping Recurrent Neural Networks And Trajectory Classification"* IJCNN, Vol.1, pp431-436, 1992

[19] Zohn Rosen *"Data Set: Speech Recognition Project"* Internal communication, 1996.

[20] Octavio Betancourt, John Antrobus *"Vowel Identification From Harmonic Contours of Vowel Centers Using An Automatic Algorithm That Eliminates Windowing Error"* J. Acoust. Soc. Am. Vol.99 p2497, 1996

[21] Octavio Betancourt, John Antrobus *"Using Natural Harmonics as Acoustic Features in Speech Recognition: A Vowel Classification Example"* J. Acoust. Soc. Am. 1999

[22]Charles R. Jankowski Jr., Hoang-Doan Vo, Richard P. Lippmann *"A Comparison of Signal Processing Front Ends for Automatic Word Recognition"* IEEE Trans. on Speech and Audio Processing Vol.3(4) pp286-293, July 1995

[23] Richard P. Lippmann *"Accurate Consonant Perception Without Mid-Frequency Speech Energy"* IEEE Trans. on Speech and Audio Processing, Vol.4(1), pp66-69, January, 1996

[24] Hesham Tolba, Douglass O'Shaughnessy *"Automatic Speech Recognition Based on Cepstral Coefficients And A Mel-Based Discrete Energy Operator"* ICASSP Vol.2, 1998

[25]Nandakishore Kambhatla, Todd K. Leen *"Dimension Reduction by Local Principal Component Analysis"* Neural Computation 9, pp1493-1516, 1997

[26] Ramesh R. Sarukkai *"Supervised Networks That Self-Organize Class Outputs"* Neural Computation Vol.9(3) pp637-48, 1997

[27] Kam-Chuen Jim, C. Lee Giles, Bill G. Horne *"An Analysis of Noise in Recurrent Neural Networks: Convergence and Generalization"* IEEE Trans. on Neural Networks, Vol.7, No.6, Nov. 1996

[28] George N. Karystinos, Dimitris A. Pados *"On Overfitting, Generalization, and Randomly Expanded Training Sets"* IEEE Trans. on Neural Networks, Vol.11, No.5, pp1050-1057, Sept. 2000

[29] Lutz Prechelt *"Automatic Early Stopping Using Cross Validation: Quantifying The Criteria"* Neural Networks 11, pp761-767, 1998

[30] Zehra Cataltepe, Yaser S. Abu-Mostafa, Malik Magdon-Ismail *"No Free Lunch For Early Stopping"* Neural Computation 11(4), pp995-1009, 1999

[31] Jianchang Mao, Anil K. Jain *"Artificial Neural Networks for Feature Extraction And Multivariate Data Projection"* IEEE Trans. on Neural Networks, Vol.6, No.2, pp296-317, Mar. 1995

[32] Biing-Hwang Juang, Wu Chou, Chin-Hui Lee *"Minimum Classification Error Rate Methods for Speech Recognition"* IEEE Trans. on Speech And Audio Processing, Vol.5, No.3, pp257-265, May 1997

[33] Rohit Lotlikar, Ravi Kothari *"Bayes-Optimality Motivated Linear And Multilayered Perceptron-Based Dimensionality Reduction"* IEEE Trans. on Neural Networks, Vol.11, No.2, Mar. 2000

[34] Jan Larsen, Lars Nonboe Anderson, Mads Hintz-Madsen, Lars Kai Hansen *"Design of Robust Neural Network Classifiers"* ICASSP, Vol.2, pp1205-1208, 1998

[35] M. Thomae, G. Ruske, T. Pfau *"A New Approach to Discriminative Feature Extraction Using Model Transformation"* ICASSP, Vol.3, pp1615-1618, 2000

[36] Hong C. Leung, Victor W. Zue *"Phonetic Classification Using Multi-Layer Perceptrons"* ICASSP, pp525-528, 1990

[37] Philip Clarkson, Pedro J. Moreno *"On The Use of Support Vector Machines for Phonetic Classification"* ICASSP, Vol.2, pp585-588, 1999

[38] John S. Garofolo, Lori F. Lamel, William M. Fisher, Jonathan G. Fiscus, David S. Pallett, Nancy L. Dahlgren *NIST Speech Disk Documentation: DARPA TIMIT*

Acoustic-Phonetic Continuous Speech Corpus, NISTIR4930, National Institute of Standards And Technology 1993

[39] Zaki B. Nossair, Stephen A. Zahorian *"Dynamic Spectral Shape Features as Acoustic Correlates for Initial Stop Consonants"* J. Acoust. Soc. Am. Vol.89(6) pp2978-91, June 1991

[40] Stephen A. Zahorian, Amir J. Jaghrghi *"Spectral-Shape Features Versus Formants as Acoustic Correlates for Vowels"* J. Acoust. Soc. Am. Vol.94(4) pp1966-81, October 1993.

[41] Stephen A. Zahorian, Zaki B. Nossair *"A Partitioned Neural Network Approach for Vowel Classification Using Smoothed Time/Frequency Features"* IEEE Trans. on Speech And Audio Processing, Vol.7, No.4, pp414-425, Jul. 1999

[42] Robert V. Shannon, Fan-Gang Zeng, Vivek Kamath, John Wygonski, Michael Ekelid *"Speech Recognition with Primarily Temporal Cues"* Science, Vol.270, No.13, pp303-304, 1995

[43] Yaser S. Abu-Mostafa, Amir F. Atiya *"Introduction to The Special Issue on Neural Networks in Financial Engineering"* IEEE Trans. on Neural Networks, Vol.12, No.4, pp653-655, 2001

[44] Marcelo C. Medeiros, Alvaro Veiga *"A Hybrid Linear-Neural Model for Time Series Forecasting"* IEEE Trans. on Neural Networks, Vol.11, No.2, pp1402-1412, Nov. 2000

[45] Ken-ichi Funahashi *"Multilayer Neural Networks And Bayes Decision Theory"* Neural Networks 11, pp209-213, 1998

[46] Lawrence R. Rabiner *"A Tutorial on Hidden Markov Models And Selected Applications in Speech Recognition"* Proc. IEEE Vol.77, pp257-285, 1989