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

Université d'Ottawa • University of Ottawa

PERMISSION DE REPRODUIRE ET DE DISTRIBUER LA THÈSE

PERMISSION TO REPRODUCE AND DISTRIBUTE THE THESIS

NOM DE L'AUTEUR / NAME OF AUTHOR:	SHAO, Ying
ADRESSE POSTALE / MAILING ADDRESS:	608-196 METCALFE STREER OTTAWA ON K2P1P8
GRADE / DEGREE:	**ANNÉE D'OBTENTION / YEAR GRANTED**
M.A.Sc. (Electrical Engineering)	2003
TITRE DE LA THÈSE / TITLE OF THESIS:	COMPARISON OF CLASSIFICATION TECHNIQUES FOR SPEECH / AUDIO APPLICATIONS

L'auteur permet, par la présente, la consultation et le prêt de cette thèse en conformité avec les règlements établis par le bibliothécaire en chef de l'Université d'Ottawa. L'auteur autorise aussi l'Université d'Ottawa, ses successeurs et cessionnaires, à reproduire cet exemplaire par photograhie ou photocopie pour fins de prêt ou de vente au prix coûtant aux bibliothèques ou aux chercheurs qui en feront la demande.

Les droits de publication par tout autre moyen et pour vente au public demeureront la propriété de l'auteur de la thèse sous réserve des règlements de l'Université d'Ottawa en matière de publication de thèses.

The author hereby permits the consultation and the lending of this thesis pursuant to the regulations established by the Chief Librarian of the University of Ottawa. The author also authorizes the University of Ottawa, its successors and assignees, to make reproductions of this copy by photographic means or by photocopying and to lend or sell such reproductions at cost to libraries and to scholars requesting them.

The right to publish the thesis by other means and to sell it to the public is reserved to the author, subject to the regulations of the University of Ottawa governing the publication of theses.

N.B. LE MASCULIN COMPREND ÉGALEMENT LE FÉMININ

Dec. 17. 2002 — Ying Shao

DATE (AUTEUR) SIGNATURE (AUTHOR)

Université d'Ottawa · University of Ottawa

Université d'Ottawa • University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

SHAO, Ying

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE - DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Comparison of Classification Techniques for Speech/Audio Applications

Martin Bouchard

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

R. Goubran

T. Aboulnasr

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

SIGNATURE

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Comparison of Classification Techniques for Speech / Audio Applications

By

Ying Shao

A thesis submitted to

The Faculty of Graduate and Postdoctoral Studies

in partial fulfillment of the requirements of the degree of

Master of Applied Sciences

in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

School of Information Technology and Engineering

Faculty of Engineering

University of Ottawa

Abstract

This classification of speech vs. noise or speech vs. music can be used in speech/audio signal processing applications as an important part to achieve a lower bit-rate or to enhance the performance of coding in multimedia communications. In recent years, some research on the classification has been published. In the previous work some traditional classification algorithms were used, such as linear Least Mean Squares, the Nearest Neighbour and the Quadratic Gaussian algorithms. Some of the previous work also used machine learning software and neural networks with basic training algorithms. This thesis provides an extensive experimental simulation of the speech classification problem. In this thesis, the Extended Kalman Filter algorithm is proposed to train a neural network classifier. Our experiments show that using neural networks in speech/noise/music classification produces a more robust and powerful classification than other traditional algorithms. Furthermore, the Extended Kalman Filter provides a fast convergence and gives results near the global optimal. Our results show that the learning algorithm chosen to train the neural network is very important to the final results. Therefore, the neural network classifier trained with the Extended Kalman Filter is compared with the traditional classification methods but also with other neural classifiers trained with different learning algorithms. The experiments performed for the thesis are clean speech vs. noise classification, clean speech vs. music classification and active/inactive speech classification of noisy speech.

Table of Contents

List of Figures

List of Acronyms

1-NN	Nearest Neighbour testing, 1 candidate
k-NN	Nearest Neighbour testing, k candidates
BP	Back Propagation learning
CNG	Comfort Noise Generator
CS-ACELP	Conjugate-Structure Algebraic CELP
DSP	Digital Signal Processing
ECM	Expected Cost for Misclassification
Ef	Full band Energy
El	Low band Energy
EKF	Extended Kalman Filter
LM	Levenberg-Marquadt learning
LMS	Least Mean Square
LPC	Linear Prediction Coefficient
LSF	Line Spectrum Frequency
PCM	Pulse Code Modulation
QG	Quadratic Gaussian
SNR	Signal to Noise Ratio
VAD	Voice Activity Detector
ZC	Zero Crossing rate

Acknowledgements

I would like to express my deepest gratitude to my thesis supervisor, Dr. Martin Bouchard for his constant advice, encouragement, and support throughout this thesis. Some new ideas in this thesis came from him, and his patience and rich knowledge made it a pleasure to have discussions with him.

I would also like to thank the professors in SITE for their great jobs in teaching the state-of-the-art knowledge to us. A special thanks is given to Dr. Peter Galko, Dr. Emil Petriu, Dr. Eric Dubois for the great courses that they offered.

Chapter 1

Introduction

1.1 Motivation:

This thesis focuses on the classification of speech, noise, and music. Classification is not a new subject in data processing. During the last couple of decades, its principles and methodologies have been widely and successfully used in the development of almost every field in science and engineering, such as artificial intelligence, weather forecast, even medical diagnosis. Its use has been especially accelerated by the developments in the computer technology. However, it is still a fascinating field for researchers. In the following paragraphs some applications of classification in modern speech/audio processing are presented.

1. One of the most well known applications of classification in speech processing is the VAD (Voice Activity Detector) in speech coding. This technique is widely used in recent ITU and ETSI speech coding standards [1] [2]. Although the detailed algorithms of these two groups are a little bit different, the general idea is the same, i.e. using different encoder for speech and noise, reducing the coding quality in noise/inactive periods, to achieve a lower average bit rate for multimedia transmission, such as wireless transmission or Voice over IP. In this technique, VAD is used to classify speech into two classes: active and inactive, which are processed using different encoders. ITU G.729B VAD is an example [1]. It selects 4 parameters describing the energy and spectral content of the signal. These 4 parameters construct a 4-dimensional vector space. It supposes that the parameters of active speech occupy a certain area in the vector space, and that the inactive/noise is

clustered in another hyper-volume. It uses linear four-dimension decision boundaries to separate one class from the other.

2. A similar application is needed in combined speech/audio coding [3]. It is important in many multimedia applications. Obviously to get a good quality of music we need more bits compared with that of speech. During a digital program of radio or over the Internet, speech and music occur alternately. This classifier discriminates speech from music, using lower bits speech codecs to treat speech, and using audio codecs to process music. The overall bit rate is reduced significantly.

3. Another application is in multi-mode coding. In the new proposed 4 kbit/s candidate for speech coding [4], it attempted to provide toll quality speech coding at 4 kbit/s for all conditions. To achieve this aim, the classification is used too. Instead of a 2-class classification as in ITU G.729 VAD, the speech is classified into 2 modes and 5 classes according to whether the signal is periodic or stationary. After that, according to different service requirements and the results of classification, it assigns different coding schemes.

4. Classification in speech coding is also sometimes used to classify different types of noise for good performance of CNG (Comfort Noise Generator). Usually, in speech coding, a CNG algorithm is added in the decoder accompanying a VAD in the encoder to compensate the abnormal silence inserted by the VAD during the conversations. CNG works according to the received SID (Silence Insertion Descriptor) signal. The decoder gets the information about the inactive voice from the SID, and generates an output signal which is similar to the background noise in the sender side, to make the transition between active and inactive voice imperceptible/comfortable. But a SID containing only power level and power spectrum information (LPC coefficients) may not be accurate to get good CNG quality. More accurate descriptions about the inactive voice (such as the types of noise or stationary/non-stationary) are preferred. Then in the decoder some pre-recorded/ pre-designed excitation signals can be used for each type. This needs to be done by classifying the inactive speech further [5].

1.2 Contributions:

A new method based on non-linear neural networks with fast convergent learning, the Extended Kalman Filter algorithms, was used. It was compared with the traditional classification algorithms (including Least Squares algorithm, Nearest Neighbour algorithm, and Quadratic Gaussian algorithm). The test was done on both speech vs. noise and speech vs. music classification, and using clean speech and noisy speech. In this thesis our results of classification using different algorithms under different situations are presented. The results show that neural networks are the most robust, powerful tool for speech classification.

The main contributions of this thesis are:

1. Compared classification results for not only clean speech but also noisy speech;
2. Compared classification results for both speech/noise and speech/music discrimination.
3. Compared classic classification methods with neural network methods, and different neural network training methods (Extended Kalman Filter, Levenberg-Marquadt, Back Propagation, etc.) for classification.
4. Compared the computational complexity of the different classification methods.

1.3 Thesis Layout:

The remaining chapters will be divided as following:

Chapter 2: Characteristics and features of speech, noise, and music

In this chapter, the differences among these three types of signals are compared, and described using basic DSP tools. Then the features used in our work are described and the reasons why they extract the characteristics of these signals are explained.

Chapter 3: Algorithms for Classification

In this chapter we briefly introduce the traditional classification algorithms first, then the basic structure of neural networks, with the standard Back Propagation learning. After that the Extended Kalman Filter algorithm is explained in details.

Chapter4: Experimental results for clean speech

In this chapter the experimental results of classification on clean speech vs. noise and clean speech vs. music are presented step by step, one by one. At the end, we get the detailed understanding of each algorithm, the parameters in each algorithm, the structures and the effect of features.

Chapter5 : Experiments for noisy speech

In this part the resulting structures from the previous chapter are used to classify the noisy speech under different SNR ratios.

Chapter 6: Computational complexity

In this chapter the computational complexities of all the algorithms are compared.

Chapter 7: Conclusion and future works

In the last chapter we conclude the thesis by discussing the main results and contributions of the thesis as well as the possible future work that may enhance the results presented here.

Chapter 2

Characteristics and Features of Speech / Noise / Music

To achieve a good classification performance, it is required to have some *a priori* knowledge of speech, noise, and music, and good features are needed to extract the characteristics of the signals. In this chapter, the production mechanisms of these signals are described. We use some basic digital signal processing tools to visually present the differences among them. Then, the features that were used in our work (LSFs, ZC, Ef, El) are explained. It shows that these features properly and significantly extract the main differences between the three types of signals.

2.1 Production mechanisms of these three types of signals:

Speech is produced by a vocal tract excited by the passing airflow from the lung [7]. When we want to speak, the brain orders the muscles of vocal organs to move and to force the air in the lung flow across the vocal tract. The vocal organs make up a vibration tube, and make some constraints on the airflow in some cases. This causes different sounds. The air source and the nature of its flow are referred to as the excitation signal. The vocal organs shape the frequency characteristics of the vibrating air traveling through the vocal tract. The excitation signal and the vocal tract shape decide how the speech sounds. There are voiced and unvoiced speech. Voiced sounds are generated by periodic excitations, whereas the unvoiced sounds are produced by turbulence, which is caused by forcing the air passing through the constrained vocal tract. In fact the vocal tract is a slowly time-varying system. However compared to the variation of the excitation, its variation is slower. So the vocal tract can be considered as a LTI system during a short interval 5~ 50 ms [8].

The next type of sound, noise, is produced by friction between materials, or a combination of a large number of un-correlated sounds (such as the office noise, babble noise, and so on). It is usually whiter and flatter. The energy is mostly uniformly distributed along the frequencies.

The third type of sound, music, is the creative product of the musicians. It is carefully controlled, a harmonic combination of different tones or different instruments. The physical properties of music in the DSP domain are mainly affected by the mathematic characters of the instruments [9]. A brief categorization of the instruments is the following: string instruments, tubes, and percussion instruments. Strings can be further classified as plucked, struck, and bowed. Drums are examples of percussion instruments. Music is produced by the vibrations of a string (wave traveling through the string), or acoustic tube resonances excited by the player's lip vibration or the breath stream, or the vibrations caused by striking some materials. For music to achieve beautiful rhythms, the tone durations are long, the envelopes of the spectrum are smoothly changed. Music rhythms are complex, not as fine structured as speech because they are usually combinations of a lot of elements. More energy is focused on the low frequency [10].

To sum up, all sounds can be treated as stochastic processes. From digital signal processing theory, we know that " a wide-sense stationary process can be represented as the output of a causal and causally invertible linear system excited by a white noise process. The condition that the system is invertible also allows us to represent the WSS random process by the output of the inverse system with a white noise as the input"[11]. An Auto-regressive filter (AR filter) is a suitable model for sound production [7].

2.2 Visual representation of the main characters of speech, noise, and music

From a digital signal processing viewpoint, frequency response and energy distribution are two important parameters in system analysis. In this section we use basic digital signal processing tool to give a straightforward presentation of the frequency response and energy distribution of these three types of signals one by one, processing normalized 8 kHz, 16 bits PCM digital speech, noise, and music. In particular we will consider:

- Time-domain waveform representation;
- Spectrum in frequency domain representation;
- Spectrogram representation.

1. Time domain waveform representation:

Time domain waveform presents the change of the amplitude along the time axis. It gives us a rough idea of the difference among the three types of signals.

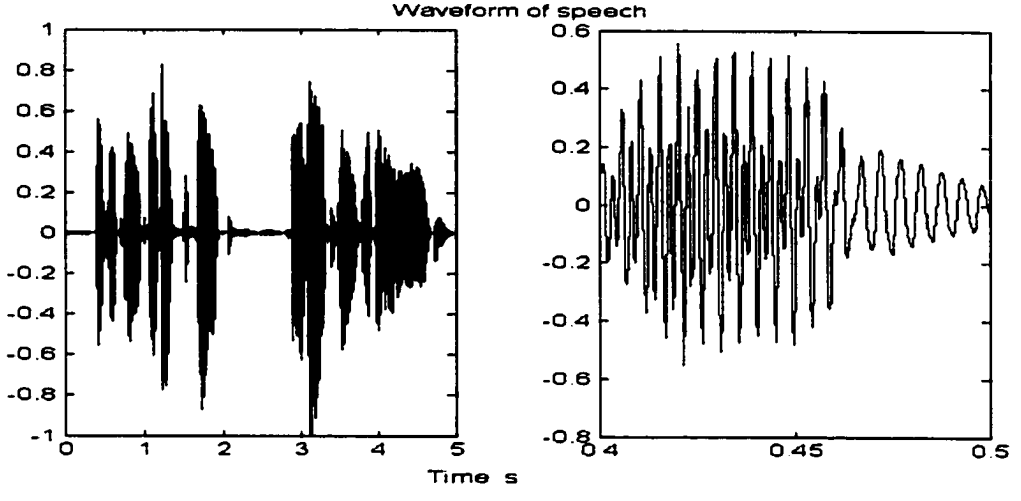


Figure 2-1: Waveform of a speech sentence and 0.1s short segment.

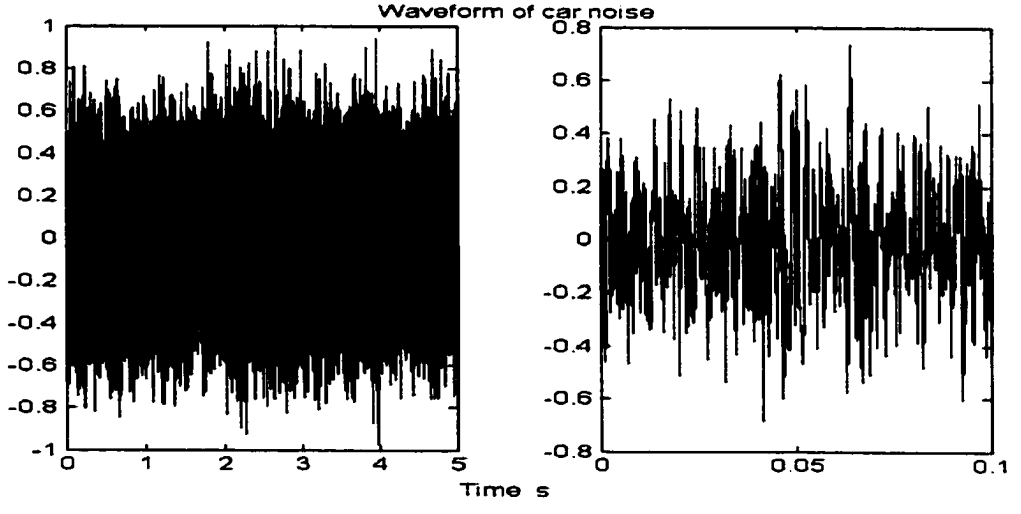


Figure 2-2: Waveform of car noise and 0.1s short segment

For noise, the values flip over zero randomly, the amplitudes are less-correlated with each other, as shown in Fig. (2-2). For voiced speech, there are apparent periodic

segments such as the segment in Fig. (2-1). Amplitude changes regularly. For music, there are obviously harmonic components and periods as in Fig. (2-3).

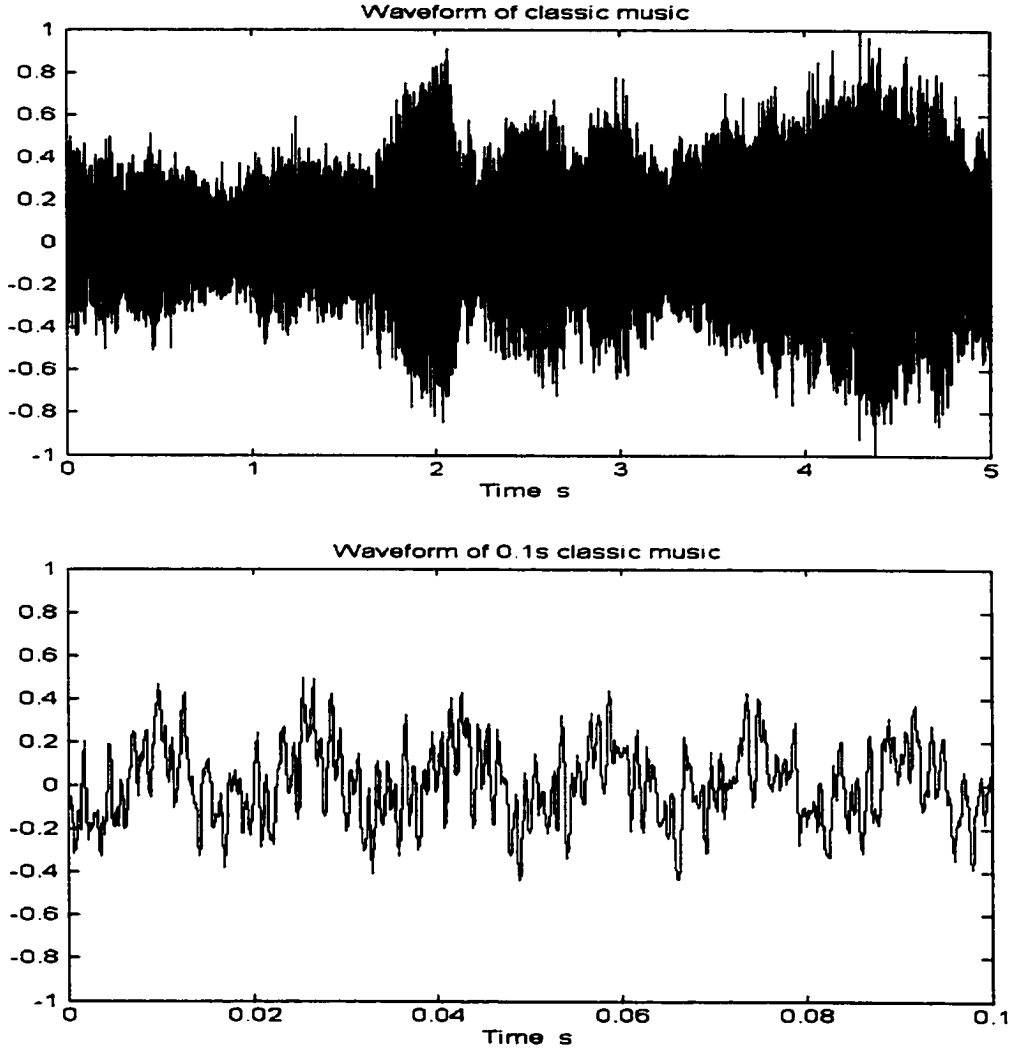


Figure 2-3: Waveform of classical music and 0.1s segment

2. Frequency response:

Frequency response is a powerful DSP analysis tool. It shows how the signal behaves in the frequency domain. As explained in a previous section, the Auto-Regressive (AR) filter is often used to model the vocal tract. The speech is generated by passing an excitation signal through the AR filter. In speech processing, this filter is estimated using linear prediction. The estimated filter is usually termed synthesis filter, denoted by 1/A(z) [12]. The frequency response of this filter indicates the formant envelopes, which are different

for each person because of the pronunciation. However, they are well structured. There are usually 3 ~ 4 formant peaks in active speech. From the spectrum graph, we can see that there are clear periodic components for voiced speech. However, it is a kind of pseudo-period. Especially, only in the short segments of voiced speech the periods are distinct in real conversations. For the most part of a speech conversation, the pitches are not obvious, and not regular.

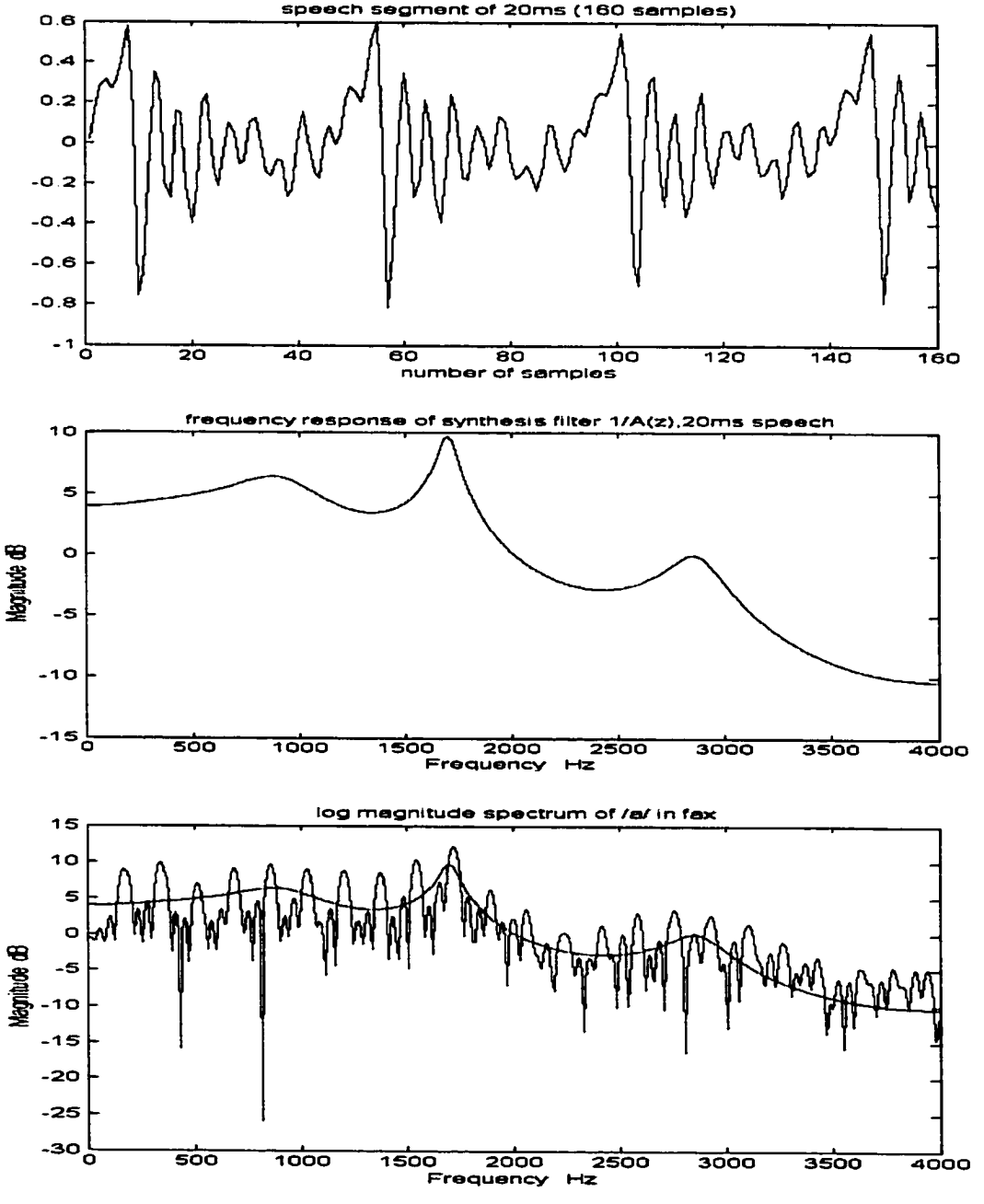


Figure 2-4: Power spectrum of speech, 20ms segment

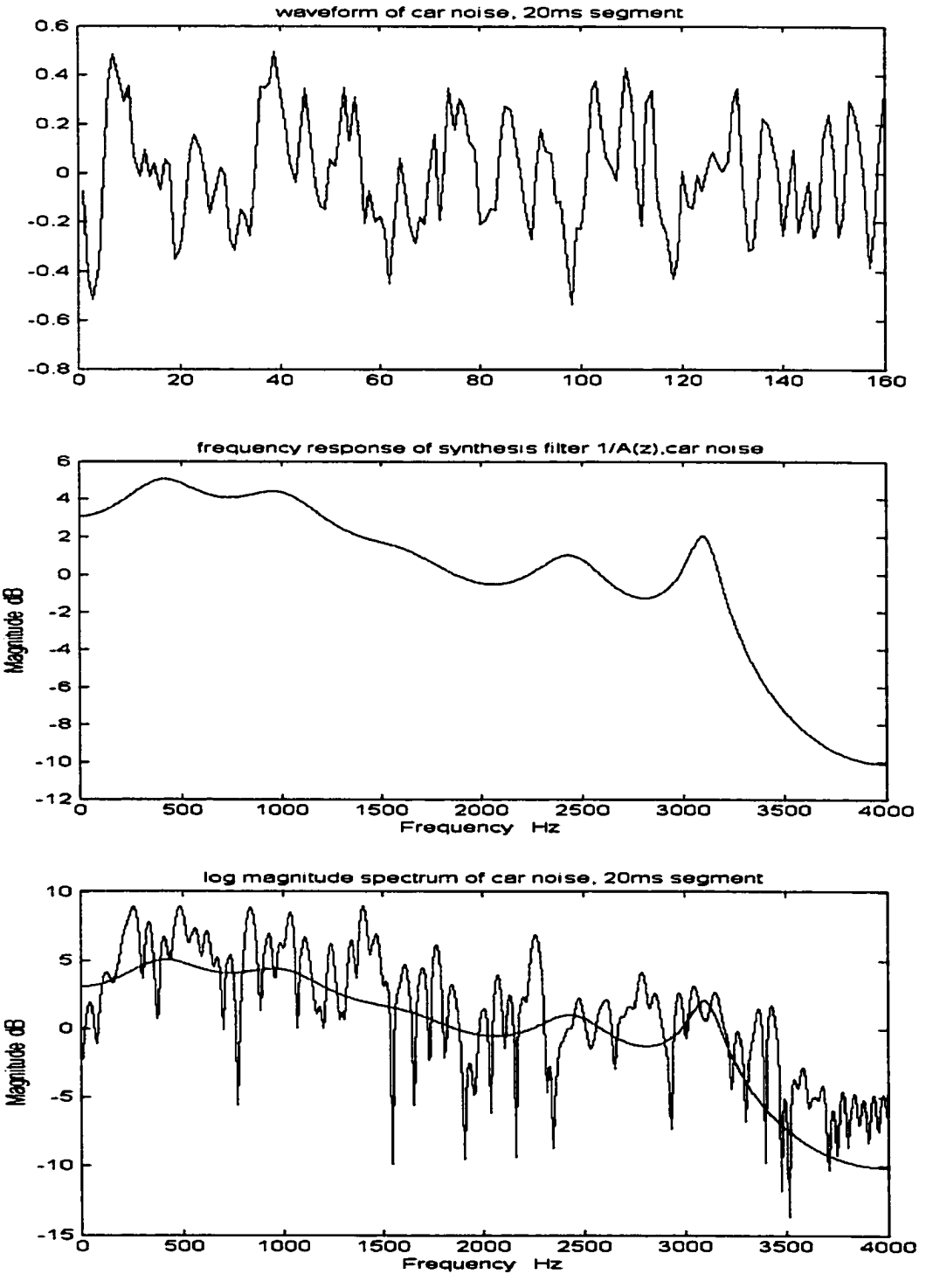


Figure 2-5: Power spectrum of car noise, 20 ms segment

From Fig. (2-5), it is clear that the frequency response of car noise is quite flat up to 3 kHz, almost an all pass filter and there is no regular and periodic pitch in the spectrum plot.

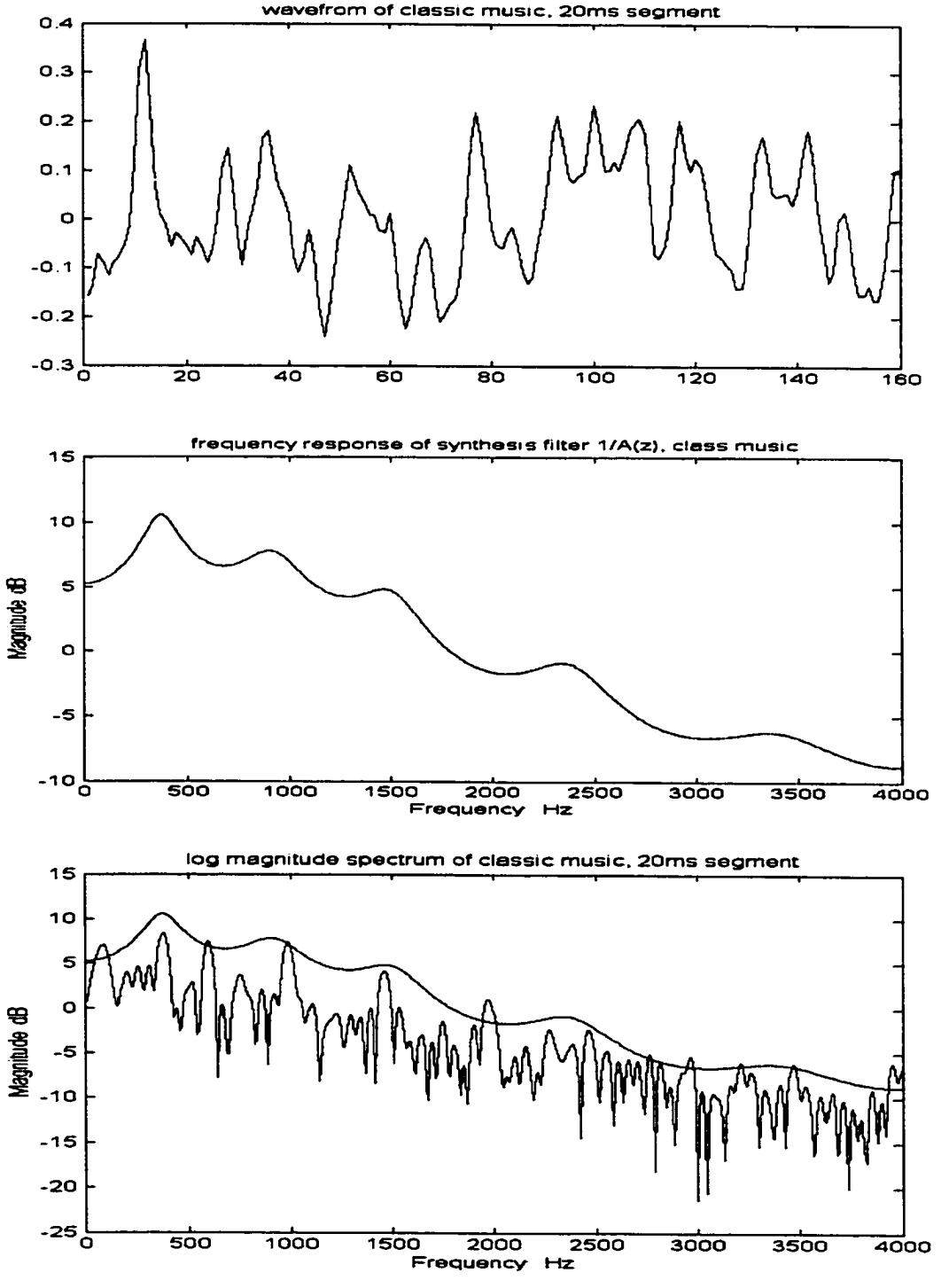
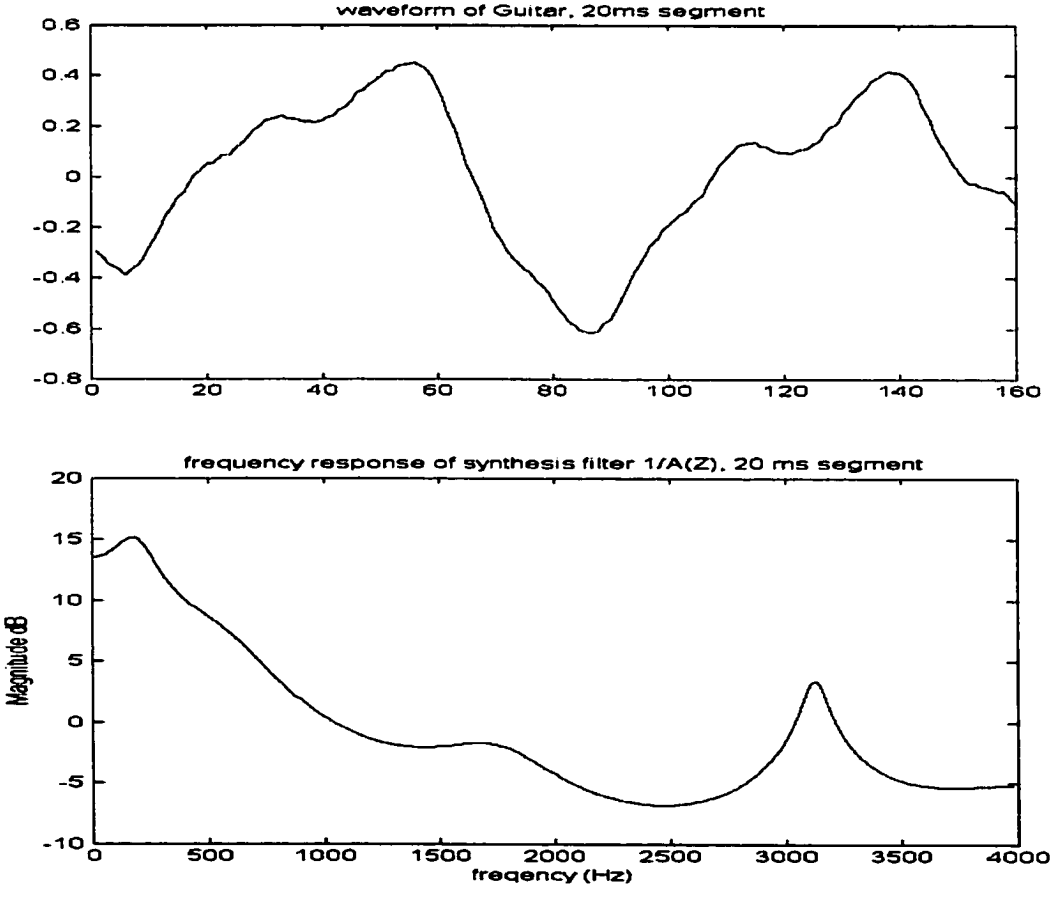


Figure 2-6: Power spectrum of classical music, 20ms segment

Music is a combination of lot of harmonic tones, so its frequency response changes smoothly. The numbers of peaks in the envelope are different under different situations. More instruments usually means smoother frequency changes. But it is still more or less a low-pass filter, not as flat as noise.

For the guitar signal in the following figure, no matter what rhythm it is performing, most of the energy is located in lower frequencies, and there are peaks at about 100,1600 and 3100 Hz. In the waveform plot, there is an exact pitch period. For a single instrument, this is always the case. But usually music is a combination of a large number of instruments. This makes things difficult for classification. In the following chapter we will show that for classification of speech vs. music the results are worse than that of speech vs. noise.



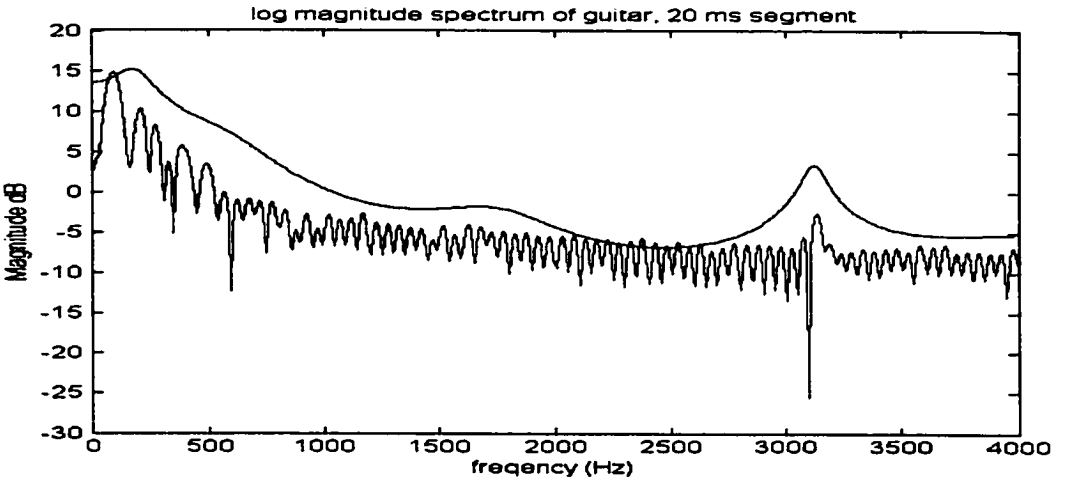
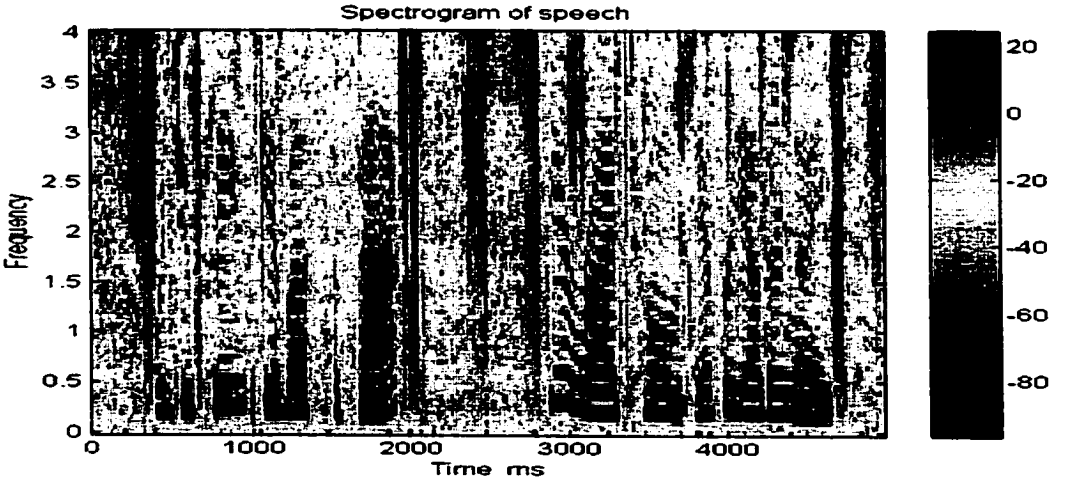


Figure 2-7: Power spectrum of guitar, 20ms segment

3. Spectrogram representation:

The spectrogram is a two-dimensional representation (time-frequency with different colors mapping the levels of energy). It presents the energy distribution along time and frequency. It is a good tool to give us a visual expression of the abstract frequency numbers. As shown in the following figure, the energy of noise is almost uniformly distributed along time and frequency axis. For classical music, the tones last very long. There are energy lines lasting as long as 1 s, while for voiced speech, the vowels last about 100 ms, obviously shorter than that of classical music. The energy changes from voiced vowels to unvoiced frictives.



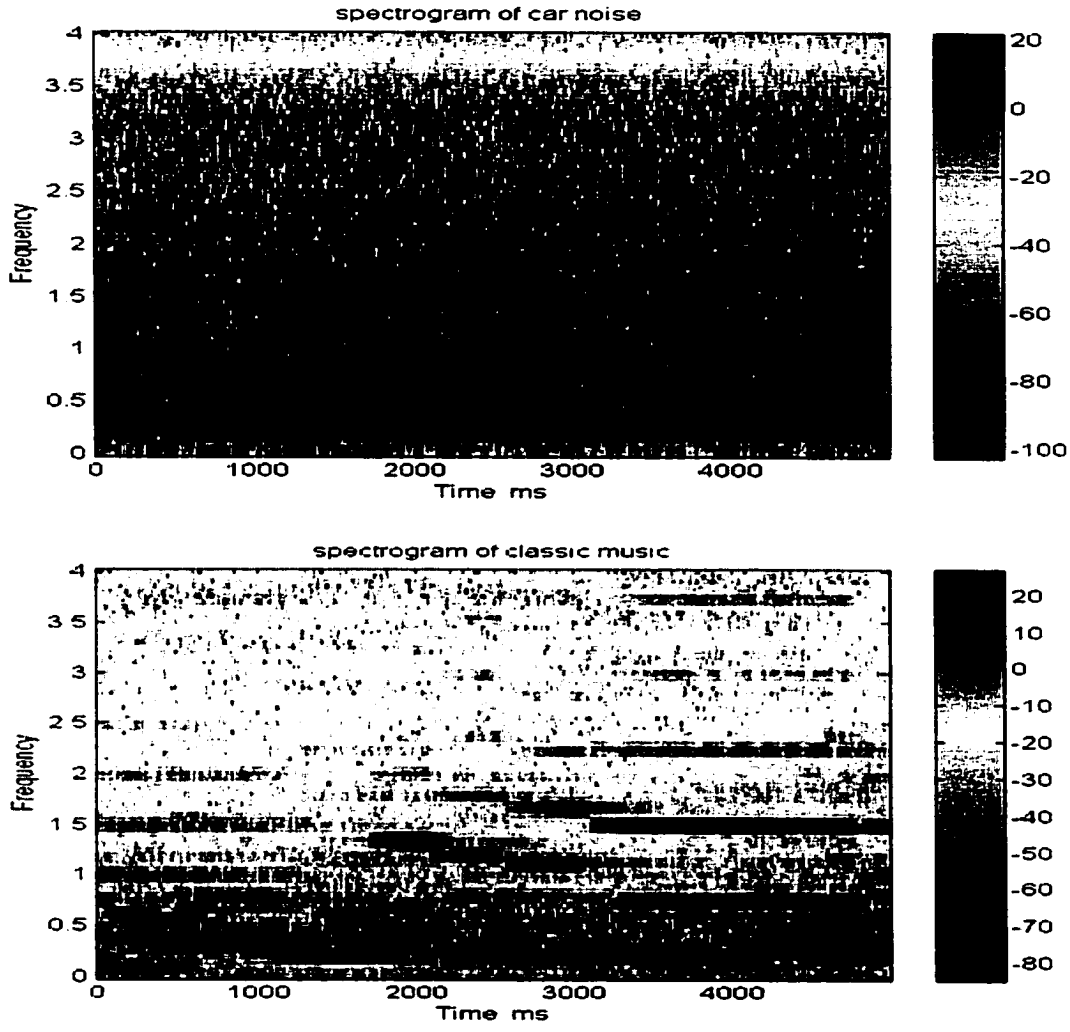


Figure 2-8: Spectrograms of three types of signal

In an orchestra, with more instruments combined, the energy is higher and widely distributed along frequency axis.

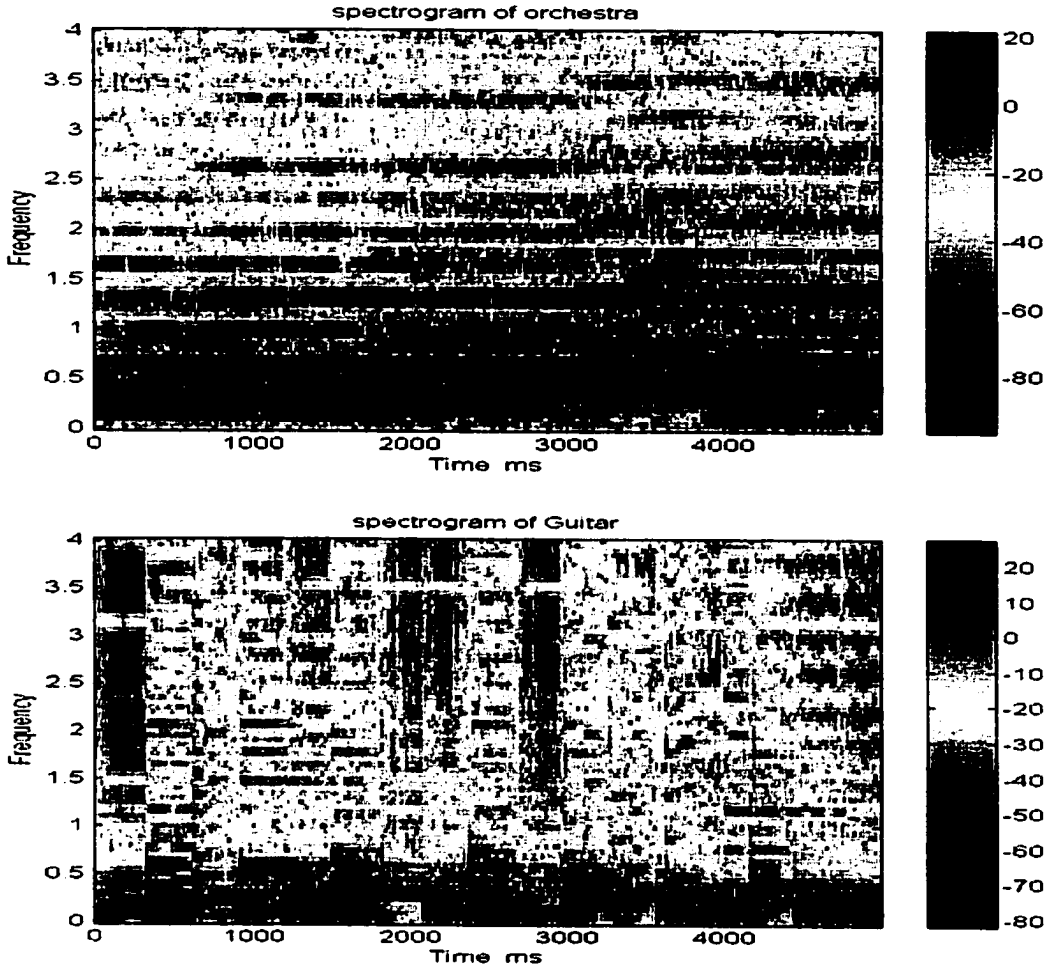


Figure 2-9: Spectrogram of orchestra and guitar

For a guitar, there are some continuous energies in lower band, and there is a lot of harmonic tones spared in the frequency axis. The tones last a short time, similar to the vowel in speech.

2.3 The features selected [1] [6]

1) Linear Spectrum Frequency (LSFs):

By the analysis of the speech production mechanism, we know that the sound can be modeled by an AR filter excited by some kind of periodic or non-periodic excitation. In the basic Digital Signal Processing theory, Linear Prediction is a powerful method to estimate the system (AR filter) producing a stochastic process [12]. We can use it in our analysis.

This problem can be solved by the Levinson-Durbin algorithm, a computationally efficient algorithm for linear prediction with time series. It minimizes the error between the original input and the reconstructed speech signal. Through the linear prediction algorithm we get the linear prediction coefficients (LPs), denoted by A, which model the structure of the vocal tract. Their values affect how the speech sounds. A 10^{th} order linear predictor was used to get the LP coefficients $A=\{a_1, a_2, \ldots, a_{10}\}$ for each frame. The corresponding AR filter is 1/A(z), where

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{10} z^{-10}.$$

For speech coding, experiments have showed that the LP coefficients are more sensitive to the quantization error, especially when the poles are located near the unit circle. There are some transformations of LP coefficients for better quantization, such as the reflection coefficients, arccosine reflection coefficients. Among them, the Linear Spectrum Frequencies (LSFs), denoted by ω_i, are among the best, and experiments have shown that they are more compact in their vector space, i.e. more separable [13]. A lot of research in speech classification uses LSF coefficients[1][14][15]. The algorithm for getting the LSFs (ω_i) from the LPs (A(z)) is the following.

$$F_1(z)=[A(z)+z^{-11} A(z^{-1})]/(1+z^{-1}) = \prod_{i=1,3,\ldots 9} (1 - 2q_i z^{-1} + z^{-2});$$

$$F_2(z)=[A(z)-z^{-11} A(z^{-1})]/(1-z^{-1}) = \prod_{i=2,4,\ldots 10} (1 - 2q_i z^{-1} + z^{-2});$$

The above polynomials are symmetric. Each polynomial has five conjugate roots q_i on the unit circle. The q_i are called Linear Spectral Pair (LSP) coefficients.

$$\omega_i = a\cos(q_i), \quad i=1,2,\ldots 10. \qquad (2\text{-}1)$$

The coefficients ω_i are defined as the Linear Spectral Frequencies coefficients (LSFs). They are responsible for formant peaks in the LP spectrum. It must be mentioned here that the LSF coefficients are in increasing order, i.e. they are ordered as $0<\omega_1 < \omega_2 \ldots < \omega_{10} < \pi$.

We have explored the statistical distribution of the LP coefficients and the LSF coefficients for different sounds. We evaluated the norm, mean, and standard deviation. It showed that LSF coefficients are more preferable than LP coefficients.

The norm is defined as $N = \frac{1}{L}\sqrt{\sum_{i=1:k} x_i^2}$, where x_i is the elements in the input feature vectors, k is the dimension of the feature vectors, and L is the total number of frames.

	LP (the 10th order)	LSF (the 10th order)
Speech	0.0404	0.0793
Noise	0.0525	0.0860
Classical music	0.0374	0.0725

Table 1: The Norm of LPs and LSFs

The mean vector is $\mu = (\mu_1, \mu_2, ..., \mu_k)$, where $\mu_i = \frac{1}{L}\sum_{i=1}^{L} x_i$,

The standard deviation is $\sigma_i = (\frac{1}{L}\sum_{i=1}^{L}(x_i - \mu_i)^2)^{1/2}$, where μ_i is the mean values.

mean		1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
	speech	-1.1372	0.5611	-0.2235	0.1264	-0.0783	0.0594	0.0624	0.0513	-0.1371	0.0764
LPs	noise	-1.1005	0.6323	-0.4258	0.3688	-0.2862	0.3007	-0.2292	0.1926	-0.1258	0.0793
	speech	0.1976	0.3532	0.5848	0.9280	1.2346	1.4995	1.8779	2.1525	2.4917	2.8048
LSF	noise	0.2228	0.4401	0.6963	0.9617	1.2645	1.5327	1.8365	2.1188	2.4431	2.7097

Table 2: The mean of the 10 coefficients in LPs and LSFs

STD		1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
	speech	0.9068	0.9049	0.7751	0.6194	0.6349	0.5250	0.5370	0.4720	0.3700	0.1821
LPs	noise	0.2781	0.3580	0.3800	0.4613	0.4257	0.4259	0.3329	0.3187	0.2234	0.1537
	speech	0.0757	0.1353	0.2152	0.2009	0.2232	0.2217	0.1797	0.1602	0.1558	0.1114
LSF	noise	0.0895	0.0610	0.0781	0.0720	0.0737	0.0641	0.0672	0.0656	0.0616	0.1178

Table 3: The standard deviations of the 10 coefficients in LPs and LSFs

Comparing the stochastic character of the LPC and LSF coefficients, we can find that the standard deviations of LSF coefficients are definitely smaller than that for LPC coefficients. It means that LSF coefficients are more concentrated around their average values. Also, considering the distance between the mean of speech and the mean of noise over the standard deviation, we found that for six out of ten coefficients the normalized distances of LSFs are bigger than the normalized distances of LPs, especially for the first three and the last two coefficients, where they are several times of those of LPs:

$$normalized\ distance = \left| \frac{mean\ of\ speech - mean\ of\ noise}{STD\ of\ noise} \right|.$$

The comparison is shown in Table 4.

	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
LPs	0.1320	0.1989	0.5324	0.5255	0.4884	0.5666	0.8759	0.4434	0.0506	0.0189
LSFs	0.2815	1.4246	1.4275	0.4681	0.4057	0.5179	0.6161	0.5137	0.7810	0.8073

Table 4: Comparison of the normalized distance of LPs and LFSs

This means that small distance of the same quantity accounts for a bigger proportion in the decision when we use LSF coefficients. In normalized space, the LSFs are more separated. So LSFs are more powerful in pattern classifications. The following is the result of 1-NN (Nearest Neighbour algorithm) speech vs. noise classification using 10^{th} order LPs and corresponding 10^{th} order LSF coefficients. For each class, there are 20 centroids for the prototype. (The detailed explanation of the algorithm will be given in Chapter 4.) It shows that using the LSFs, the performance is improved by about 50%. So in the rest of this thesis we will use LSFs as a main set of features in classification. More details on the experiments setup will be provided in Chapter 4.13

LPs			LSFs		
Error for speech	Error for noise	Total error	Error for speech	Error for noise	Total error
21.31	4.25	15.93	10.7	1.9	7.9

Table 5: Error rate of 1-NN classifying speech vs. noise (%)(20 centroids)

2) **ZC:**

ZC is the zero-crossing rate of the time domain waveform. It provides a measure of the average of the spectral energy distribution in the waveform, the spectral center. It has been widely used in practice as a strong feature to detect fricative (or noise) from voiced speech.

$$ZC = \frac{1}{2M}\sum_{i=0}^{M-1}[|\mathrm{sgn}(x(i)) - \mathrm{sgn}(x(i-1))|], \tag{2-2}$$

where M is the length of the frame.

3) **Full-band Energy:**

Energy is another important feature to classify the different acoustic signals. We know that vowels are usually higher energy events containing at lower frequencies. Noise-like signals have lower spectral energy distributed more uniformly and more towards the higher frequencies. Full band energy E_f is the logarithm of the normalized first autocorrelation coefficient R(0).

$$E_f = 10\log_{10}[\frac{R(0)}{N}] \tag{2-3}$$

4) **Lower-band Energy:**

Low band energy E_l is measured on the 0~1kHz band.

$$E_l = 10\log_{10}[\frac{1}{N}h^T Rh]; \tag{2-4}$$

where h is the impulse response of a 10^{th} order FIR filter with cutoff frequency at 1 kHz. R is the Toeplitz auto-correlation matrix with the auto-correlation coefficients on each diagonal. N is the number of samples in the frame.

2.4 How the features extract the most information about the differences

In the previous sub-sections, we provided the definitions of some features. We analyzed the differences among three types of signals too. Matching them with each other, we can see that the features that we used extracted main aspects of the differences.

1. LSFs: As explained in this chapter, this set of parameters gives us the information about the formant envelope, the formant peaks. From the analysis of the speech

production mechanism we know that the synthesis filter designs how the sounds sound. From the figures of the frequency response it shows that there are large differences. Between the three types of signals LSFs are the most important information about the sound.

2. ZC: it is a way to measure the time-domain waveform. It shows how often the amplitude flips around zero. It is a measure of the average frequency of a segment. It illustrates the average frequency in a short segment.

3. Energy: from the spectrograms we see that there are different energy distribution styles for the three types of signals. For noise, the energy is nearly uniform. For music more energy is concentrated in the low frequency part. So, we use full-band energy and lower-band energy, two parameters to extract the information about the energy distribution.

Chapter 3

Algorithms for Classification

In this thesis, four different kinds of classifiers are compared. Here the basic algorithms of each kind of classifiers are explained.

3.1 Linear Classifier [12]:

The linear classifier assumes that the class boundaries can be defined by a linear combination of the input features. It uses mean-square optimization to find the optimal estimation using a linear function. If the class boundaries are not a linear function of the input features, this method will find the nearest linear estimation to match them.

Assume the classifier output is a linear function. It can be expressed as:

$$y_i = w_1 x_{1i} + w_2 x_{2i} + ... + w_n x_{ni} + b$$

where $w_1, w_2, ..., w_n$ are the weights, b is a bias, $x_{1i}, x_{2i}, ...x_{ni}$ are the elements of the i^{th} the feature inputs, and y_i is the corresponding system output . The bias b can be treated as a constant input 1 with a weight w_0.

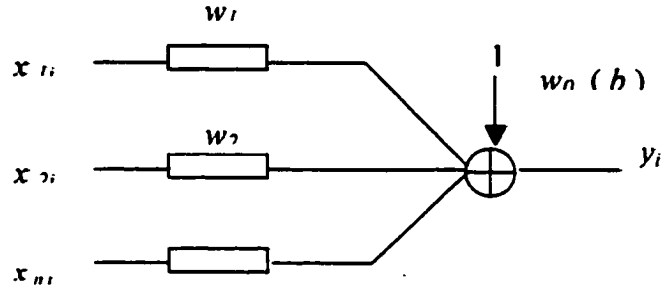


Figure 3-1: Structure of a linear classifier

We can also define a vector notation with $X_i=[\ 1\ x_{1i},\ x_{2i},\ \ldots,\ x_{ni}]^T$ and $W=[w_0, w_1, w_2, \ldots, w_n]^T$, then the system can be expressed in vector notation:

$$Y_i = X_i^T W = W^T X_i;$$

The linear classifier is tuned by finding the optimal weight vector W that minimizes the energy of the error e_i, distance between the classifier output y_i and a target signal d_i:

$$e_i = d_i - y_i$$

$$e_i^2 = (d_i - y_i)^2 = d_i^2 + W^T X_i X_i^T W - 2 d_i X_i^T W;$$

Assume that e_i, d_i, X_i are stationary over all input vectors,

$$E(e_i^2) = E((d_i - y_i)^2) = E(d_i^2) + W^T E(X_i X_i^T) W + 2E(d_i X_i^T) W;$$

Let $R = E(X_i X_i^T)$, $P = E(d_i X_i^T) = E[d_i x_{1i}\ d_i x_{2i} \ldots\ d_i x_{ni}]$, $E(e_i^2) = E(d_i^2) + W^T R W + 2PW$.

For the expected error $E(e_i^2)$, the minimum is achieved when the differential of the expected error $E(e_i^2)$ with respect to W is zero, i.e.

$$\nabla = \frac{\partial E}{\partial W} = [\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_k}] = 2RW - 2P = 0.$$

This holds when $RW^* = P$, i.e. $W^* = R^{-1}P$, (3-1)

where W^* is the optimal weight vector. Under this condition, the expected error $E(e_i^2)$ achieves the minimum, it means the overall distance from the actual outputs to the desired outputs is minimum. So the linear system described by W^* is the optimal estimation for the linear classification.

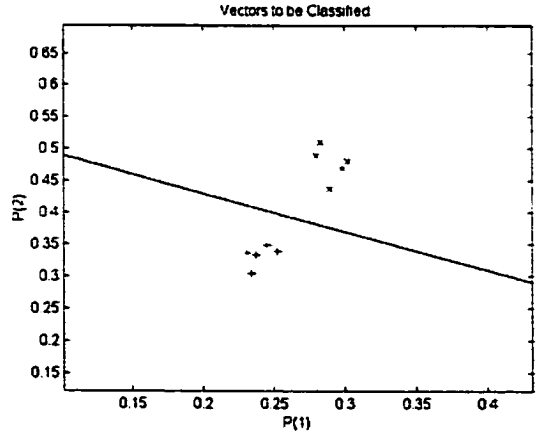


Figure 3-2: Two-dimensional illustration of the linear classification

In fact, W^* defines a hyper-plane to separate the feature space. Fig. (3-2) shows this in the two dimensions case. Because the features that we want to classify are not always linearly separable, what the linear classifier gives us is just the optimal linear approximation that it can get.

3.2 Nearest Neighbour Method

As the name implies, the Nearest Neighbour Method uses the distances between observations and the centroids of each class to classify them. Suppose that we have three classes 1,2, and 3. Each has one centroid--- cent1, cent2, and cent3. If x is nearer to cent 1, it will be classified into class 1. Many distances can be used to measure the proximity of a centriod. The Euclidean distance is the simplest one, and we have used it in the calculations.

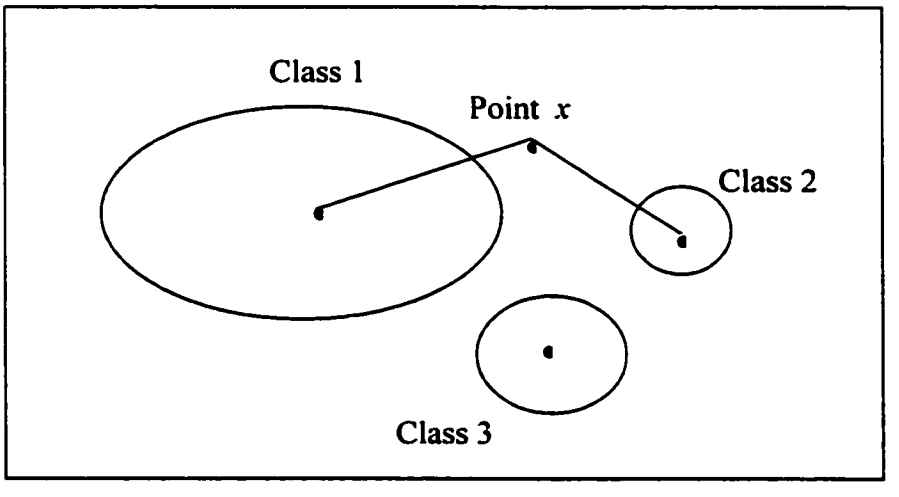


Figure 3-3: Nearest Neighbour with one prototype each class

When the classes are not normally distributed with equal variances in all directions, misclassification will occur. That is because a single point in each class does not represent the class very well. As shown in Fig. (3-3), class 1 spreads wider than class2 and 3. It is obvious that the point x is more likely to belong to class1. But if we check the distance to each centroid, x should be in class2.

So we introduce another term---prototype [16]. It means that if we have a priori knowledge that some points are in class *i*, we refer to them as the prototypes. If a new observation is near one prototype in class *i*, it is assigned to class *i*. The best way could be to use all the points in the training data as the prototypes. But it would need a lot of memory and computations to cover all probabilities. In practice, we divide classes into smaller pieces to make their distributions comparable, and use the centroids of subclasses to represent the prototypes. Then each class is represented by multi-prototypes instead of a single point.

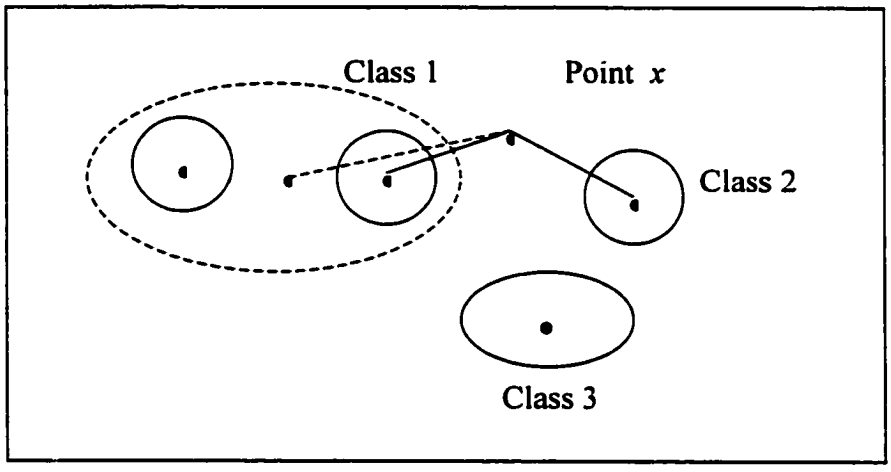


Figure 3-4: Nearest Neighbour with multi-prototypes

As shown in Fig. (3-4), by dividing the class 1 into pieces, we got two centroids to represent the class 1. This processing makes the distribution fair and comparable with the other classes. In the new system, the point x is classified into class 1.

The algorithm for Nearest Neighbour classification based on multi-prototypes is the following[14]:

$$x \in class_j \text{ if } D(x, cent_j^{m_j}) = \min_{all\ M\ cent} [D(x, cent_i^{m_i})],$$

where $cent_i^{m_i}$ is the m_i^{th} of centroids of class i, M_i is the number of sub-classes for class i, M is total number of centroids for all N classes, $M = \sum_{i=1}^{N} M_i$. $D(x, cent_i^{m_i}) = \left| x - cent_i^{m_i} \right|^2$, it is the distance from x to the centroid $cent_i^{m_i}$.

Arrange all the centroids from 1 to M, denoted in vector notation, the distance from x to the i^{th} centroid $cent_i$ is

$$d(i) = D(x, cent_i) = \left| x - cent_i \right|^2 = (X - C_i)^T (X - C_i)$$

$$= X^T X - 2X^T C_i + C_i^T C_i = X^T X - (2X^T C_i - C_i^T C_i),$$

where Ci denotes vector notation of the i^{th} centroid $cent_i$, X is the vector notation of the input signals. The first part is constant for all i. To seek $\min_i[D(x, cent_i)]$ is equivalent to seeking

$$\min_i[-2X^T C_i + C_i^T C_i] \text{ for i=1,2...}M.$$

Each class is separately clustered into sub-classes, M_i sub-classes for class i. The centroids are labeled according to the classes, the total number of centroids for all classes is $M = \sum_{i=1}^{N} M_i$, where N is the total number of classes. During testing, all the M centroids of all classes are checked, according to the label on the centroids to identify them.

K-mean clustering algorithm [17] from Vector Quantization can be used to separate each class into subclasses. The basic algorithm of K-mean is the following:

1. Decide the training data X and the number of subclasses M_i for each class i ; initialize the centroid for each subclass;
2. Calculate the distance between each training data and each centroid . If $d(j)$ is the smallest within the M_i values, assign $x(i)$ to subclass j; iterate to check all the training data.
3. Calculate the new centroids by averaging all the training data classified in each sub-class and update them.

4. Check the convergent condition. If the decrease in the overall distance (sum of all distance for all the training data) is less than 0.1%, stop. Else go back to step 2.

Because the K-mean algorithm only finds a local optimum, we need to train many times to get a better clustering, i.e. to get a smaller overall distance.

There are several ways to initialize the centroids.

1) Regular pick of the centroids from training data;
2) Random pick of the centroids from the training data;
3) Random values as the initial centroids.

Our simulations have shown that the first way can get a fast performance (smaller overall distance). But because it is fixed, it is possible that the training focuses on only a subset of the input data and it is usually not the best. The second way randomly chooses the initial centroids from the training data. The overall distance D is more or less around the average. But it can get a smaller D in several trials. As for the third, because the coefficients used have some stochastic distribution and various mean, this way needs more times of trial to get a good result. The first approach was used to get a threshold performance (on the overall distance), and then the second method was used to get a better solution.

The number of centroids affects the result of the classifier significantly. We will present our experimental results in chapter 4 and 5.

3.3 Quadratic Gaussian Method:

The Quadratic Guassian method is also called Bayesian Optimal classifiers, which are based on probabilistic information on the population of the training data. It is one of the major pattern recognition techniques. It assumes that the feature vectors of each class obey a multivariate gaussian distribution. In our case, according to the Large Number theory in statistics, this assumption is acceptable when we use a large number of data to train the classifiers.

Below is the basic theory of Bayesian optimal classifiers [18]. First, we define some terms that are used in the analysis.

1. $f(x \mid \omega_i)$, the conditional probability distribution function of observation x given that x is chosen from class i.
2. $p(\omega_i)$, the a priori probability that x is chosen from class i.
3. $p(x \in \omega_i \mid x)$, the conditional probability, showing the probability that x is in class i while given the observation x.
4. $c(i \mid j)$, cost of misclassification, which means the cost of misclassifying x from j to i. All the costs are set to positive values, and $c(i \mid i)=0$.
5. $L_{di}(x)$, cost (loss) of misclassification for one particular decision di given that $x \in \omega_j$, it is the cost of misclassifying x from other classes to decision i.
6. ECM, the expected loss of the misclassification for class i.
7. R_i, The optimal decision regions, it is defined by $R_i = \{x \mid d_i(x) \geq d_j(x)$ for all $j\}$ for class i, where $d(x)$ is the decision function of the classifier. Then if $x \in R_i$, the classifier assigns x to ω_i.

In our analysis, the misclassification of a decision includes two parts: (1) $x \in \omega_j$, but during classifying, it is assigned to other class ω_i; (2) x comes from other classes, but is classified as class j. As shown in Fig. (3-5), suppose the decision is $x \in \omega_1$, misclassifying occurs in the overlapping area for case (1) or for case (2). In other words, the purpose of this classifier is to set a suitable region Ri by a decision function such as to minimize the probability of misclassifying. So the next aim is to find the decision function $d(x)$.

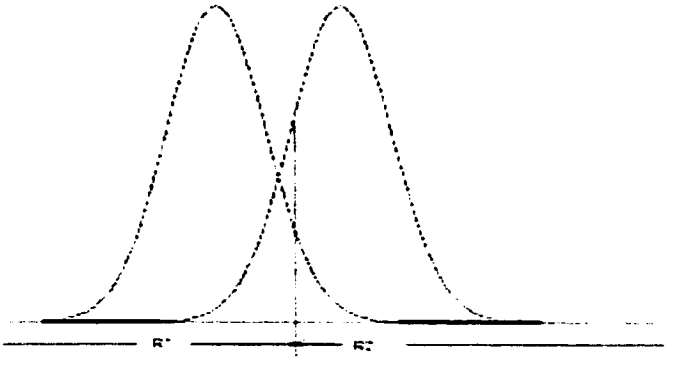


Figure 3-5: Quadratic Gaussian classification and the decision region

With the normal distribution assumption, according to Bayes' rule, the conditional cost of misclassification for one particular decision d_i given that x $\in \omega_j$ [19],

$$L_{di}(x)=c(i\,|\,j)p(x\in\omega_i\,|\,x) \quad = c(i\,|\,j)\frac{p(x\,|\,\omega_i)p(\omega_i)}{p(x)}$$

Assume that all the observations are chosen randomly in the universe X of the feature domain, then $p(x)$ is constant. The above equation can be further simplified as

$$c(i\,|\,j)p(x\,|\,\omega_{i)})p(\omega_i).$$

The expected cost $ECM_i(x)$ for class i (misclassify x from ω_j to d_i, or misclassify x from i to j , j=1:M, $j\neq i$) is

$$ECM_i(x)= \sum_{\substack{j=1:M\\ j\neq i}} c(i\,|\,j)\times p(x\in\omega_i\,|\,x)+ \sum_{\substack{j=1:M\\ j\neq i}} c(j\,|\,i)p(x\in\omega_j\,|\,x)$$

$$= \sum_{\substack{j=1:M\\ j\neq i}} c(i\,|\,j)\times\frac{p(x\,|\,\omega_i)p(\omega_i)}{p(x)} + \sum_{\substack{j=1:M\\ j\neq i}} c(j\,|\,i)\times\frac{p(x\,|\,\omega_j)p(\omega_j)}{p(x)}$$

As same as above, the equation can be further reduced to

$$ECM'_i(x)= \sum_{\substack{j=1:M\\ j\neq i}} c(i\,|\,j)p(x\,|\,\omega_i)p(\omega_i) + \sum_{\substack{j=1:M\\ j\neq i}} c(j\,|\,i)p(x\,|\,\omega_j)p(\omega_j) \qquad (3\text{-}2)$$

So the decision function of the classifier is the one that minimizes the $ECM'_i(x)$.

For example, suppose that we have two classes, 1 and 2. The universe X=R1 $\cup$ R2, R1 is the range of class 1 and R2 is the range of class2, they are exclusive and exhaustive.

The expected cost for misclassification of class1 is equal to

$$ECM'_1(x)= c(2\,|\,1)p(x\,|\,x\in\omega_1)p(\omega_1)+c(1\,|\,2)p(x\,|\,x\in\omega_2)p(\omega_2),$$

For the optimal classier, the $ECM'_1(x)$ should be minimized. Let's see under what condition this is achieved.

$$ECM'_1(x) \quad = c(2\,|\,1)\int_{R2} f(x\,|\,x\in\omega_1)dx\cdot p(\omega_1)+c(1\,|\,2)\int_{R1} f(x\,|\,x\in\omega_2)dx\cdot p(\omega_2);$$

Changing the integral region to the decision area R1,

$$ECM'_1(x) \quad = c(2\,|\,1)p(\omega_1)(1-\int_{R1} f(x\,|\,x\in\omega_1)dx)+c(1\,|\,2)p(\omega_2)\int_{R1} f(x\,|\,x\in\omega_2)dx$$

$$= c(2\,|\,1)p(\omega_1)-c(2\,|\,1)p(\omega_1)\int_{R1} f(x\,|\,x\in\omega_1)dx+c(1\,|\,2)p(\omega_2)\int_{R1} f(x\,|\,x\in\omega_2)dx$$

$$= c(2|1)p(\omega_1) - [c(2|1)p(\omega_1)\int_{R1} f(x|x\in\omega_1)dx - c(1|2)p(\omega_2)\int_{R1} f(x|x\in\omega_2)dx]$$

$$= \underbrace{c(2|1)p(\omega_1)}_{(1)} - \int_{R1} \underbrace{[c(2|1)p(\omega_1)f(x|x\in\omega_1) - c(1|2)p(\omega_2)(f(x|x\in\omega_2)]}_{(2)}dx$$

The first part of above equation is positive. The minimization of ECM_1' is achieved by ensuring that the integrants (the second part) in the above equation is a positive quantity and as large as possible, i.e.

$$c(2|1)p(\omega_1)f(x|x\in\omega_1) - c(1|2)p(\omega_2)(f(|x\in\omega_2) \geq 0;$$

$$\text{or} \quad c(2|1)p(\omega_1)f(x|x\in\omega_1) \geq c(1|2)p(\omega_2)(f(x|x\in\omega_2). \qquad (3\text{-}3)$$

This criterion equation sets the optimal region to minimize the misclassification loss.

If we assume that the costs of each misclassification are the same, the criterion function is reduced to:

$$p(\omega_1)f(x|x\in\omega_1) \geq p(\omega_2)f(x|x\in\omega_2). \qquad (3\text{-}4)$$

The decision function is thus $d_i(x) = p(\omega_i)f(x|x\in\omega_i)$.

In general, $x \in \omega_j$ when $p(\omega_j)(f(x|x\in\omega_j) = \max_{i=1:M} p(\omega_i)f(x|x\in\omega_i)$ (3-5)

If μ_i and $\sum_i$ are the mean and covariance matrix of the class i, the distribution function of x can be expressed as:

$$f(x|\omega_i) = \frac{1}{(2\pi)^{n/2}\left|\sum_i\right|^{1/2}}\exp[-\frac{1}{2}(x-\mu_i)^T(\sum_i)^{-1}(x-\mu_i)] \qquad \text{i=1:M} \quad (3\text{-}6)$$

where n is the dimension of vector x, then Eq. (3-4) can be evaluated.

Trying to classify speech and noise or speech vs. music using the above equation directly, only very poor results were initially obtained. Analyzing the stochastic character of the LSF coefficients for speech, noise, and music, it was found that all ten coefficients overlap badly. As shown in Fig. (3-6), speech scatters widely in the feature domain, the standard deviation of speech is twice that of noise, and the means are very near to each other. Speech overlaps with music and noise in a lot of areas.

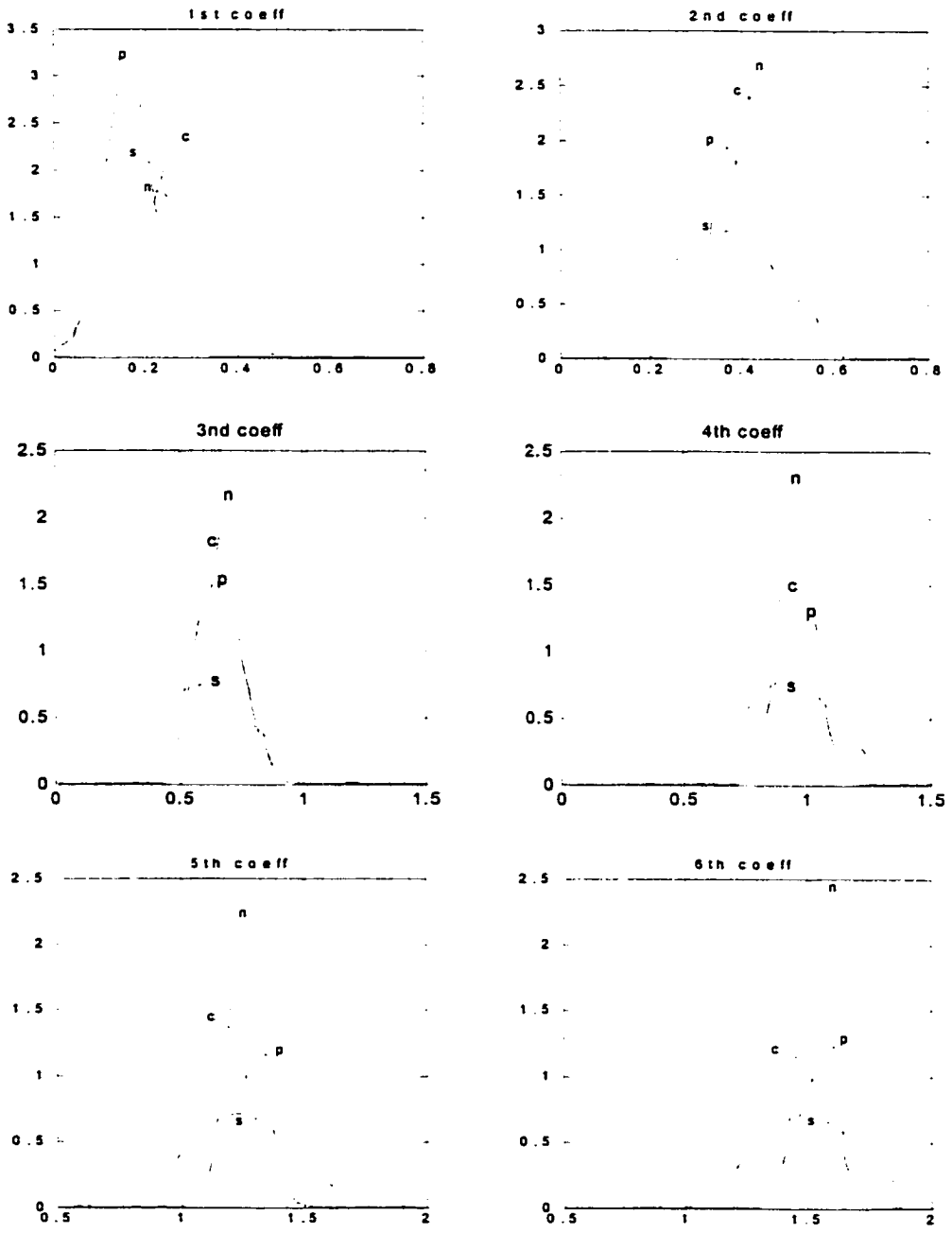


Figure 3-6: Stochastic distribution of some LSF coefficients of speech (s), noise (n), classical music (c), and pop music (p)

It was thus chosen to sub-divide the speech or both speech and noise / music to make the standard deviation similar and comparable, and make the means separable. Borrowing the idea from the Nearest Neighbour method, the speech and noise / music were divided into

subclasses, then the stochastic parameter μ_i and $\sum_i$ were extracted from each subclass. A count was made of the number of training data that were found in each subclass, and normalizing the counts by the total number of training vectors, we obtained the probability of $p(\omega i)$. In classifying, the probability function was calculated using Eq. (3-6), and then multiplying by $p(\omega_i)$. If the largest resulting value was obtained with μ_i and $\sum_i$ and $p(\omega_i)$ then x was classified to class i.

3.4 Classification using neural networks:

3.4.1 General introduction of neural networks

A neural network is an artificial intelligent network with massive parallel processing units working together. It is inspired by the human brain and nervous system, using linear or non-linear parallel computation to learn from the environment knowledge, to store the information, or to perform a particular task. In short, the neural network models human brain. The potential benefits of an artificial neural network are that it provides a higher computation ability and a great degree of robustness. It is a powerful tool for pattern recognition and parameter classification [20].

The basic unit of a neural network is the neuron. In an artificial neural network, like the neurons in the human brain, they are the basic information processing units. A neural network consists of typically a large number of neurons. The structure of a McCullon- Pitts neuron is shown in Fig. (3-7) [21]:

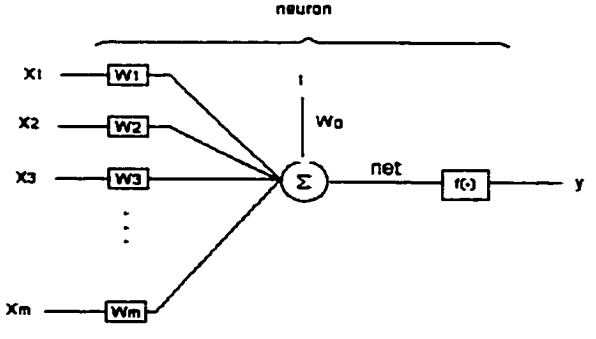


Figure 3-7: The structure of McCullon-Pitts neuron

The elements of the neuron include the following:

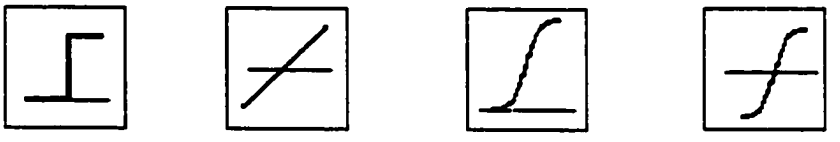
1. Synapse(--w--) : It is a connecting link between inputs and the adder with a weight.
2. Summing junction (adder, Σ): It collects all the input information with respect to the weights.
3. Activation function ($f(\cdot)$): It maps the output of the neuron with the summation of weighted inputs of the same neuron.
4. The bias: an offset, it can be seen as unit input and weight w_0.

In mathematical terms, we can express it as the following equation:

$$net_j = \sum_i w_{ji} x_i + b_j$$

$$y_j = f(net_j) = f(\sum_i w_{ji} x_i + b_j)$$

The activation function can be linear or non-linear. The commonly used activation functions are shown in Fig. (3-8).



hardlim purelin logsig tansig

$$y = \begin{cases} 0 & net < thresh \\ 1 & net \geq thresh \end{cases} \qquad y = net \qquad y = \frac{1}{1+e^{-net}} \qquad y = \frac{e^{net} - e^{-net}}{e^{net} + e^{-net}}$$

Figure 3-8: Activation Functions used in neural networks

A number of neurons connected together construct a neural network. The structure of a neural network is intimately linked with the learning rules used in training. There are fixed weight networks, supervised networks, and unsupervised networks [21]. The commonly used one is supervised network. For supervised networks, each observation vector is associated with a desired response from the network's output neurons. The network updates the weights according to the error between the actual output and the desired

output. For speech classification, we know the training data and its corresponding class. We can use the supervised structure.

In general, the different types of supervised neural network architectures are single-layer feed-forward networks, multi-layer feed-forward networks, and recurrent networks. Among them, the multi-layer feed-forward networks are the most popular artificial neural network structure being used today. The figure of a multi-layer neural network is shown in the Fig (3-9), each "o" indicates a neuron having the inner structure as that in Fig (3-7).

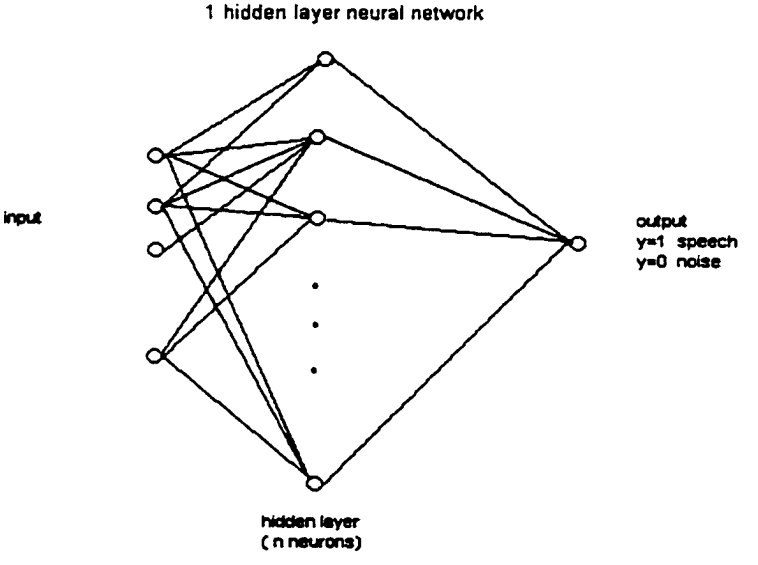


Figure 3-9: The structure of a multi-layer neural network (1 hidden layer)

Learning is the process in which the networks modify the parameters (weights) of the networks according to the environment information and particular learning laws. The next sub-section describes some learning algorithms.

3.4.2. The Standard Back Propagation Algorithm [17] :

Back Propagation is the most widely used learning law for multi-layer neural networks. It provides a computationally efficient method for the training of multi-layer neural networks. It is an approximate steepest descent algorithm in which the cost function is the mean square error E, i.e. it updates the weights to minimize the mean square error E and

modifies the weights in the direction opposite to the measured gradient vector $\frac{\partial E}{\partial w_{ji}}$. For a single-layer neural network, the derivatives with respect to the weights can be computed easily. In the multi-layer neural networks the calculation is more complex. We need to use chain rule of calculus to alter the weights backwardly. This is why the algorithm is named Back Propagation.

Let w_{ji} denote the synaptic weight connecting the output of the neuron i to the input of neuron j while i and j indicate different neurons in the network. Then we can express the basic idea of steepest descent algorithm in a mathematic equation:

$$w_{ji}^{new} = w_{ji}^{old} - \eta \frac{\partial E}{\partial w_{ji}} = w_{ji}^{old} + \Delta w_{ji}, \qquad \text{where } E = \frac{1}{2}\sum_j e_j^2,\ e_j = d_j - y_j$$

y_j is the output of j^{th} neuron in the output layer, d_j is the target, desired output for neuron j, e_j is the corresponding error signal.

1. For the single-layer network:

$$\frac{\partial E}{\partial w_{ji}} = \underbrace{\frac{\partial E}{\partial e_j} \cdot \frac{\partial e_j}{\partial y_j} \cdot \frac{\partial y_j}{\partial net_j}}_{\frac{\partial E}{\partial net_j}} \cdot \frac{\partial net_j}{\partial w_{ji}} = e_j \cdot (-1) \cdot f'(net_j) \cdot x_i$$

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = \eta \cdot \delta_j \cdot x_i,\ x_i \text{ is the } i^{th} \text{ input.}$$

We define δ_j the local gradient of neuron j, as $\delta_j = -\frac{\partial E}{\partial net_j}$.

2. For multi-layer network:

1). For the neurons in the output layer the situation is similar with the previous case.

$$\Delta w_{ji} = -\eta \cdot e_j \cdot (-1) \cdot f'(net_j) \cdot y_i = \eta \cdot \delta_j \cdot y_i$$

where y_i is the output of the previous hidden layer.

2) . For the neurons in the hidden layer, the derivatives are complex.

$$\Delta w_{ji} = -\eta \cdot \frac{\partial E}{\partial w_{ji}} = -\eta \cdot \frac{\partial E}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ji}} = \eta \cdot \delta_j \cdot y_i$$

with $\delta_j = -\frac{\partial E}{\partial net_j} = f'(net_j) \cdot \sum_k \delta_k w_{kj}$, k indicate the neurons in the upper layer.

In general, the modification Δw_{ji} applied to the synaptic weight w_{ji} is defined by the delta rule:

$$\begin{pmatrix} \text{weight} \\ \text{modification} \\ \Delta w_{ji} \end{pmatrix} = \begin{pmatrix} \text{learning} \\ \text{rate} \\ \eta \end{pmatrix} * \begin{pmatrix} \text{local} \\ \text{gradient} \\ \delta_j \end{pmatrix} * \begin{pmatrix} \text{input signal} \\ \text{of the neuron j} \\ y_i \end{pmatrix}$$

input signal or output signal of the lower layer neurons

Although the standard Back Propagation algorithm works well for small networks or simple problems, there are many parameters to guess in its initialization, and the convergence is poor if the problems become complex. Alternative fast learning algorithms in the training are needed.

3.4.3 Levenberg-Marquadt Learning (LM) [22][23]:

Although the Back Propagation is the most common learning algorithm, it converges very slowly. It needs hundreds, even thousands of epochs. In practical application some fast algorithms are developed, most of which are modified Back Propagation. There are some fast convergence learning algorithms, such as conjugate gradient algorithm, quasi-Newton, and Levenberg-Marquardt used in Matlab Neural Networks Toolbox. Matlab toolbox compared these faster learning algorithms. In the conclusion, it claims that Levenberg-Marquadt learning is the fastest in most of cases [23].

LM algorithm is based on the Gauss-Newton algorithm. It combines the speed of the Newton algorithm with the stability of the steepest descent method. The LM algorithm uses the following formula to update the weights of the network.

$$w_{k+1} = w_k - (J_k^T J_k + \mu I)^{-1} J_k^T e \qquad (3\text{-}7)$$

where μ is the learning rate, J is the Jacobian matrix of the output errors with respect to the weights of the neural network, $J = \frac{\partial E}{\partial W}$. For μ=0, it becomes the Gauss-Newton method. For large μ the LM algorithm becomes the steepest descent or BP algorithm.

3.4.4 The Extended Kalman Filter (EKF) [24] :

The problem of computing the weights of a feed-forward network to accomplish a desired input/output mapping can be viewed as a high dimensional non-linear system identification problem. In this area, the Kalman Filter is an interesting approach. Unlike the gradient technique, it computes the optimal value of the weights based on all the previous observed data. It relies on the state space representation and can cope with non-stationary and time-varying signals.

Standard Kalman Filters work on linear systems [25]. Extended Kalman Filters extend the power of Kalman Filters to non-linear situations. Extended Kalman Filters linearize the estimation around the current estimate using Taylor series, to get a set of recursive equations for non-linear systems. Here the Extended Kalman Filter is briefly explained.

The state space representation of a non-linear finite state discrete time system is the following:

$$x[n+1] = f(x[n]) + g(x[n]) \cdot u[n] + w[n] \tag{3-8}$$

$$y[n] = h(x[n]) + v[n] \tag{3-9}$$

while $u[n]$ is the input vector at time n, $y[n]$ denotes the corresponding output vector at the same time, w[n] is the process error, $v[n]$ is the measurement error, x[n] is the state of the system at time n. $f(\cdot),g(\cdot),h(\cdot)$ are non-linear functions of the states. Assume that the initial state is x[0] and sequences { $w[n]$ },{$v[n]$} are independent and Gaussian distributed with $v[n] \sim N(0,R[n])$ and $w[n] \sim N(0,Q[n])$, where $R[n]$ and $Q[n]$ are the standard deviation of $v[n]$ and $w[n]$ respectively. Our problem of identifying the system is to find an estimate $\hat{x}[n+1]$ of $x[n+1]$ given $u[j]$, $y[j]$, $0 \le j \le n$. We denote this estimate by $\hat{x}[n+1|n] = E(x[n+1]|U^n, Y^n)$, while U^n is the sequence {u[0], u[1],...u[n]}, Y^n is the

sequence of $\{y[0],y[1],..y[n]\}$. If the non-linear functions $f(\cdot)$, $g(\cdot)$, $h(\cdot)$ are differentiable, using the Taylor series we can expand them around the state a posteriori estimate $\hat{x}[n]$ and a priori estimate $\hat{x}[n|n-1]$.

$$f(x[n]) = f(\hat{x}[n]) + F(n)\cdot(x[n]-\hat{x}[n]) + ... \tag{3-10}$$

$$g(x[n]) = g(\hat{x}[n]) + ... = G(n) + ... \tag{3-11}$$

$$h(x[n]) = h(\hat{x}[n|n-1]) + H^T(n)\cdot(x[n]-\hat{x}[n|n-1]) + ... \tag{3-12}$$

where $F(n) = \frac{\partial f}{\partial x}|_{x=\hat{x}[n]}$, $H(n) = \frac{\partial h}{\partial x}|_{x=\hat{x}[n|n-1]}$, $G(n) = g(\hat{x}[n])$.

Neglecting higher order terms and assuming knowledge of $\hat{x}[n]$ and $\hat{x}[n|n-1]$, the system can be approximated as:

$$\begin{aligned} x[n+1] &= f(\hat{x}[n]) + F[n]\cdot(x[n]-\hat{x}[n]) + G[n]u[n] + w[n] \\ &= F[n]x[n] + G[n]u[n] + f(\hat{x}[n]) - F[n]\hat{x}[n]) + w[n] \\ &= F[n]x[n] + G[n]u[n] + w'[n] \end{aligned} \tag{3-13}$$

The estimate is $\hat{x}[n+1] = F[n]\hat{x}[n] + G[n]\hat{u}[n]$ and $w'[n] = f(\hat{x}[n]) - F(n)\hat{x}[n]) + w[n]$

$$\begin{aligned} y[n] &= h(\hat{x}[n|n-1]) + H^T[n]\cdot(x[n]-\hat{x}[n|n-1]) + v[n] \\ &= H^T[n]x[n] + v[n] + h(\hat{x}[n|n-1]) - H^T[n]\hat{x}[n|n-1] \\ &= H^T[n]x[n] + v'[n] \end{aligned} \tag{3-14}$$

The estimate $\hat{y}[n] = H^T\hat{x}[n]$, and $v'[n] = h(\hat{x}[n|n-1]) - H^T[n]\hat{x}[n|n-1]$.

Just like linear systems, we can get the recursive equations of the Extended Kalman Filter as the following:

The desired estimate $\hat{x}[n|n-1]$ can be obtained by the recursion:

$$\hat{x}[n|n-1] = f(\hat{x}[n-1]) \tag{3-15}$$

$$\hat{x}[n] = \hat{x}[n|n-1] + K[n]\cdot(y[n]-h(\hat{x}[n|n-1]) \tag{3-16}$$

$$K[n] = p[n|n-1]\cdot H[n]\cdot(R[n] + H^T[n]p[n|n-1]H[n])^{-1} \tag{3-17}$$

$$P[n|n-1] = F[n]P[n-1]F^T[n] + Q[n] \tag{3-18}$$

$$P[n] = p[n|n-1] - K[n]H^T P[n|n-1] \tag{3-19}$$

K[n] is called the Kalman gain. P[n] is the conditional error covariance matrix.

Next we will show how the multi-layer feed-forward network can be cast into the form of the extended Kalman algorithm. [26]

Suppose that the multi-layer neural network has a structure as in Fig (3-9) with L layers, and $N(k)$ neurons in the k^{th} layer . The input layer is layer 0 with N_0 dimensional inputs. The total number of weights in the system is $M = \sum_{k=1}^{L}(N(k-1)+1)\cdot N(k)$. The system inputs (or outputs of k^{th} layer neurons) is denoted as u^k[n] at time n. Let the weights in the system constitute the state x of the nonlinear system, thus consisting of all the weights arranged in a linear array with the dimension equal to the total number of weights M in the system. The new system state space representation is the following:

$$x[n+1] = x[n] \tag{3-20}$$

$$d[n] = h(x[n], u[n]) + e[n] = y[n] + e[n] \tag{3-21}$$

where $u[n]$ is the input vector, $d[n]$ is the corresponding desired outputs, $y[n]$ is the actual output of the designed system, $h(\cdot)$ denotes the non-linear relationship between the inputs, weights, and outputs of the net. Compared with the system that is denoted by Eq. (3-8) and (3-9), here the net has $f(\cdot)=1$, $g(\cdot)=0$.

The initial state x[0] of the network is defined as the Guassian random variable with an $N(\bar{x}[0], P[0])$ distribution, where mean and standard deviation $\bar{x}[0], P[0]$ reflect the *a priori* knowledge about the net. In the absence of any knowledge, an $N(0, \frac{1}{\varepsilon} I)$ distribution can be used, while ε is a small number (we set ε=0.001) and I is the identity matrix.

For the system in Eq. (3-20) and (3-21), the Extended Kalman Filter simplifies to the following equations (Here $\hat{x}[n]$ is equal to $\hat{x}[n|n-1]$ in previous equations):

$$\hat{x}[n+1] = \hat{x}[n] + K[n]\cdot(y[n] - h(\hat{x}[n], u[n])) \tag{3-22}$$

$$K[n] = P[n]H[n]\cdot(R[n] + H^T[n]P[n]H[n])^{-1} \tag{3-23}$$

$$P[n+1] = P[n] - K[n]H^T[n]P[n] \tag{3-24}$$

In feed-forward nets, h (·) is the activation function, usually one of those in Fig. (3-8). Then $H_{ji} = \frac{\partial y_i}{\partial x_j}$ is a M^*N matrix, M is the total number of neurons in the network, N is the number of neurons in the output layer.

The final set of weights $x(\infty)$ is obtained as the limit of a search $\lim_{n\to\infty} x[n] = x(\infty)$, updated at iteration step n with knowledge of $[U^n, Y^n]$.

Suppose that $R[n] = \frac{1}{\lambda} I_1$, where I_1 is a N^*N identity matrix, Eq.(3-23) is

$$K[n] = \lambda^{-1} P[n]H[n] \cdot (I_1 + \lambda^{-1} H^T P[n]H[n])^{-1} \quad (3\text{-}25)$$

$$P[n+1] = \lambda^{-1} P[n] - \lambda^{-1} K[n]H^T[n]P[n] \quad (3\text{-}26)$$

Set $P[0] = \frac{1}{\varepsilon} I_2$, and ε=0.001 where I_2 is a M^*M identity matrix.

We get the set of equations to update the network weights using the extended Kalman filter algorithm, exchanging X to W, we have:

$$P[0] = \frac{1}{\varepsilon} I_2 \qquad \varepsilon=0.001 \quad (3\text{-}27)$$

$$K[n] = \lambda^{-1} P[n]H[n] * [I_1 + \lambda^{-1} H^T[n]P[n]H[n]]^{-1} \quad (3\text{-}28)$$

$$W[n+1] = W[n] + K[n] \cdot e[n] \quad (\text{learning rate } \eta \text{ is } 1) \quad (3\text{-}29)$$

$$P[n+1] = \lambda^{-1} P[n] - \lambda^{-1} K[n]H^T[n]P[n] \quad \text{where } \lambda=0.999, \quad (3\text{-}30)$$

These four equations were used in our implementation.

The Extended Kalman filter algorithm computes the optimum values of the system parameters as each new input/ output data is seen. It updates parameters consistently with all previously seen data. Usually it converges very fast, and achieves a better local optimum than the solution found by standard Back Propagation. Thus it can prove to be a better classifier as it will be reported in subsequent chapters.

Following is a detailed explanation of how to compute the H[n] matrix.
For the output of the neuron k in the output layer

$$e_k = d_k - y_k \qquad k=1,2...N$$

Error vector $e = [e_1, e_2 ... e_N]$, N is the number of neurons in the output layer.

Counting all the neurons in the entire network from the first layer to the output layer, the total number of weights is M; then

$W=[w_1, w_2, ..., w_M]^T$, M is the total number of weights.

$$H = \frac{\partial Y}{\partial W} = [\frac{\partial y_1}{\partial W}, \frac{\partial y_2}{\partial W}, ..., \frac{\partial y_N}{\partial W}] = [H_1, H_2, ...H_N] \ , \qquad (3\text{-}31)$$

where H_i is a column vector, $H_i = [\frac{\partial y_i}{\partial w_1}, \frac{\partial y_i}{\partial w_2}, ..., \frac{\partial y_i}{\partial w_M}]^T$.

Whatever the architecture of the network is, it is possible to arrange the weights W in the above way so that the columns of $H(n)$ contain the derivative of the global outputs with respect to all the synaptic weights. Eq. (3-27) to (3-30) were used to adapt the weights sequentially until the goal of minimum error energy was achieved. A significant benefit on top of the fast convergence and better achieved minimum is that there are less parameters to guess in this algorithm.

It is possible to show that this algorithm is consistent with the steepest descent method.
In the steepest descent method, the modifications to the weights are the negative of the gradient vectors of the cost function.

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = \underbrace{-\eta \underbrace{\frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial net_j}}_{\delta_j} \frac{\partial net_j}{\partial w_{ji}}}_{H}.$$

In the Back Propagation we define the local gradient $\delta_j = -\frac{\partial E}{\partial net_j}$, $\Delta w_{ji} = \eta \cdot \delta_j \cdot y_i$,

while in the Extended Kalman Filter algorithm (3-23) (3-31) a new global symbol H is defined.

$$H = \frac{\partial Y}{\partial W} \qquad \Delta W = -\eta \cdot H \cdot e \ .$$

The Extended Kalman Filter algorithm then uses the de-correlated version of H, $K[n]$, in the calculations to achieve a fast convergence.

Chapter 4

Experimental Results for Classification of Clean Speech

In this chapter the testing environment will be explained first. Then the results obtained using this environment to test all classifiers/ algorithms in classification of clean speech vs. noise will be presented one by one, followed by the results on clean speech vs. music. The training and testing were performed using feature vectors extracted on 20ms frames. All test results provided here are hard decisions based on a single frame only. In practice, some hangover algorithms will be used to get a soft decision based on the hard decision of the current input frame and decisions from several previous frames, for example as in the ITU-T G.729b standard. Still, as it will be shown in this thesis, good classification results can be obtained only based on a hard decision from a single 20 ms frame.

4.1 Testing Environment

Before any algorithm could be used, a proper training and testing data set had to be found. In this section, the pre-processing of the sound files before the feature extraction is introduced, and then the database is introduced.

The original sound files are *.SRC*(speech and noise) or *.MP3* (music) files. They were converted into 16 bits PCM format *.wav* files with sampling frequency of 8kHz, band-limited between 0~ 4000 Hz. In clean speech classification, all files, including speech, noise, and music, were normalized by setting the highest magnitude in time domain to the same level. The files were divided into frames and then features were extracted frame-by-frame based on 20ms frame size. Because our experiments focused on extracting the characteristics of sound, there was no use of look- ahead windowing, each frame was non-

overlapping, and processed by a 160 coefficients (20ms under 8kHz sampling frequency) Hamming window. The LP coefficients were calculated frame-by-frame using a 10^{th} order Linear Prediction algorithm. The Levinson-Durbin algorithm was used for the linear prediction. Then the LP coefficients were converted to LSF coefficients according to Eq. (2-1). The zero-crossing rate ZC, full-band energy Ef, and low-band energy El were calculated frame-by-frame as in Eq. (2-2), (2-3), (2-4). For each frame there is thus the 10 LSF coefficients, 1 ZC, 1 Ef, and 1 El, which construct a 13-dimension vector. In the future experiments some dimensions or all dimensions will be used.

10 female speech files and 10 male speech files were used for the experiments, along with files for the most common occurring noise: car noise, babble noise, street noise, and factory noise. The silence was discarded from the speech sequences manually, and only the active speech portion of the files was kept. All the resulting files were grouped into *speech.wav*, which was made up by all the active voices (female and male), whereas the *noise.wav* was made up by the four common noises. To be fair among all kinds of noises, we used almost the same number of frames of each noise to construct the database. For each frame of the sound files the desired output of the classification was known.

For the classification of speech vs. music, the speech files without silence were used same as in classifying speech vs. noise. The classical music files were obtained from the orchestra *Swan Lake* and pop music files were cut from some pop songs with music only parts selected, not including the vocal parts. These two groups were trained and tested separately because the characteristics of music are more complex, the classical music and pop music are totally different in some aspects. The classical music is a combination of lots of instruments. The tones last longer. The formant envelope changes smoothly. The pop music is usually made up of guitars and drums. The rhythms change fast.

In total, 4300 frames of speech, 3600 frames of noise, 4900 frames of classical music, and 4800 frames of pop music were used. The features were extracted for each frame. Then the orders of the feature vectors were randomized. The first 1800 frames of each class were set as the training data, the others were set as the testing data. During testing, speech and noise

were chosen as 40% : 60% ratio (this percentage comes from the general occurrence of active/ inactive speech during practical conversations) from the testing database, speech and music were chosen on a 50% : 50% ratio from the testing database. Here all the testing data were new for the classification system, it is referred to as out-group testing, whereas the in-group testing is the test using the training data.

In the following sections the results are presented by error rate percentage in tables. In the tables the "error for speech" means the error rate that speech is misclassified as noise or music.

$$e_{speech} = \frac{\text{number of frames of speech that is misclassified}}{\text{total number of frames of speech used in the testing}} \qquad (4\text{-}1)$$

The "error for noise or music' is the error rate that noise or music is misclassified as speech.

$$e_{noise} = \frac{\text{number of frames of noise(or music) that is misclassified}}{\text{total number of frames of noise (or music) used in the testing}} \qquad (4\text{-}2)$$

The "total error rate" is the overall error rate, calculated by

$$e_{total} = 1 - \frac{\text{correctly classified speech} + \text{correctly classified noise (or music)}}{\text{total number of the testing data}}. \qquad (4\text{-}3)$$

4.2 Experimental results for clean speech vs. noise classification:

In this section the schedule of experiments for different classifiers is explained in detail. The parameters are adjusted to achieve the best results. The results of classification for clean speech vs. noise using different classifiers/ algorithms with different parameters are presented. At the end, the performance of these algorithms is compared.

4.2.1 Linear Classifier:

As mentioned in Chapter 3, the linear classifier assumes that the feature space can be separated by a linear function of the features. It designs a linear continuous hyper-plane in multi-dimensional vector space, which separates the vector space into left and right for two

classes respectively. In our case the two classes are not linearly separable, they are overlapping in many areas. So the results obtained from the linear classifier for classifying speech vs. noise are very weak compared with the other methods. On the other hand, just because of its weakness, the effects of each set of features are enlarged. This helps to understand about the roles of each feature in the classification, and the degree of overlapping and the separability of the two classes.

Before the details of the results are presented, some comments about the desired output of the linear classifier will be provided. As used in many papers on linear classifiers or neural networks, there are several pairs of possible desired outputs, such as -1 and +1, -0.5 and 0.5, 0.99 and 0.01, 0 and 1, and so on [28][29].

From our experiments it was found that setting the output to a symmetric pair around zero, such as (-1, +1), (-0.5 0.5), gave us the same results because they all set a threshold at 0. If the desired output is 0 or 1, it means that the threshold is set for the output at ½. It is equal to shift the line in Fig. (3-2) right with respect to the (-1,+1) pair. In fact all of pairs separate the plane into two parts. They gave us similar results under large training and testing database. It was thus decided to set the desired output for speech as 1, for noise as 0.

Another thing which needs to be explained is the bias. Without bias, the algorithm forces the hyper-plane to pass through the origin of the multi-dimensional vector space. In the two dimensions case, like that in Fig. (3-2), it means that the line must pass through the origin. As shown in Fig. (3-2), in most cases this is not a good solution. Setting a bias gives the system more freedom in finding an optimal result. Suppose that the dimension of the feature vectors is L, we add one dimension to the vectors and set it to 1, and then the bias can be treated as a constant input 1 with a weight, like the input 1 and w_0 shown in Fig. (3-1).

Now there is enough context introduced so that the results using the linear classifier can be presented. The two classes were trained together to get an optimal weight vector W^*. In

testing, after calculating the actual outputs, the distances from the actual output to 1 and 0 were compared. If the actual output was closer to 1, the input signal was classified as speech, otherwise it was classified as noise. Following are the experimental results (error rate percentage).

Feature vectors used	error for speech (%)	error for noise (%)	total error rate (%)
ZC+Ef+El	43.667	16.722	27.50
10^{th} order LSF	24.833	22.0556	23.1667
10^{th} LSF +ZC	21.667	13.722	16.90
10^{th} LSF+ZC+Ef	19.0833	6.667	11.633
10^{th} LSF+ZC+Ef+El	12.5833 ✓	1.5 ✓	5.933 ✓
Notes: training set: speech : noise =1800 : 1800 frames testing set : speech : noise = 1200 : 1800 frames testing method : Linear classifier;			

Table 6: Experimental results of linear classifiers for classifying clean speech vs. noise

Comparing the data in the Table 4, the improvement is significant. The first two rows show that using ZC+Ef+El only, an acceptable result in noise is obtained, but it is almost a random choice in speech, while using the 10^{th} order LSFs comparable result in both speech and noise are obtained. This shows that LSFs extract most of the important information for both speech and noise, whereas ZC+EF+El are auxiliary. The next three rows show that using more features plus the 10^{th} order LSFs the performance on the noise side is improved more than twice while the speech side is also improved.

Of course even using all the four sets of features in the linear classifier, the results were still far from ideal. We will see that using other methods, even with the 10^{th} order LSFs only, we can surpass the best performance of the linear classifiers.

4.2.2 Nearest Neighbour Classification:

The Nearest Neighbour algorithm uses the prototypes and the distances to the prototypes to classify the inputs. It separates the vector space into several hyper-spheres, some of them belong to class 1, the others belong to class2. They may overlap in some degree. The centroid of each sphere represents a prototype. The key target in this algorithm is to find the sufficient and proper prototypes for each class.

The number of centroids has a significant influence on the performance: increasing the number of centroids for speech improves the performance on the speech side. However, because of the overlapping in the vector space, this increase is not unlimited. When it passes some threshold the overall error rate actually increases with the number of centroids. At the threshold situation, the volumes of the subclasses for speech and noise in the feature space are mostly similar. On the other hand, increasing the number of centroids significantly increases the computation load in testing, making it not suitable for real-time applications. Another issue in training is the initialization. In the algorithm, at the first step the initial centroids are set randomly. This causes a problem of local optimization. So the system needs to be trained several times to get a good learning result, i.e. the smallest mean distance from all the training data to the nearest centroids. Usually this needs to be done 5 ~ 10 times. The Nearest Neighbour algorithm uses unsupervised learning in training, groups the features into sub-classes, and uses the centroids of the sub-classes as the prototypes. So speech and noise are trained separately, and the centroids for each class are labelled for future testing. In testing, the distances to all the centroids are calculated and compared, and the class of the closest centroid is chosen.

The following table shows the results for different numbers of centroids tested by 1-NN. Because most of the noises are compactly distributed compared with speech, the number of centroids for noise is set to a constant, 30, and the number was changed for speech. From the table it can be seen that 50~70 centroids (or a ratio of about 2.0) is a suitable range for speech. Here only the 10^{th} order LSFs were used. Compared with the linear classifier the results are very impressive. The next experiments of clean speech vs. noise are all based on the 10^{th} order LSFs only, which will give us a fair competition among the Nearest

Neighbour classifiers, the Quadratic Gaussian classifier, and the neural network classifier. Of course using all the features (13 dimensions of the vectors), the results would be even better. For noisy speech classification more features are needed as shown in Chapter 5, where the results of using 13 dimensions will be presented for both clean and noisy speech.

number of centroids speech + noise	error for speech (%)	error for noise (%)	total error rate (%)
30 + 30	8.9167	1.8889	4.7
50 + 30	6.1667	2.5556	4.00
60 + 30	4.00	2.90	3.6667 ✓
70 + 30	4.4167	3.889	4.10
Notes: training set: speech : noise =1800 : 1800 testing set : speech : noise = 1200 : 1800 testing method : 1-NN; features: the 10^{th} order LSFs			

Table 7: Experimental results of Nearest Neighbour classifiers using 1-NN testing for classifying clean speech vs. noise

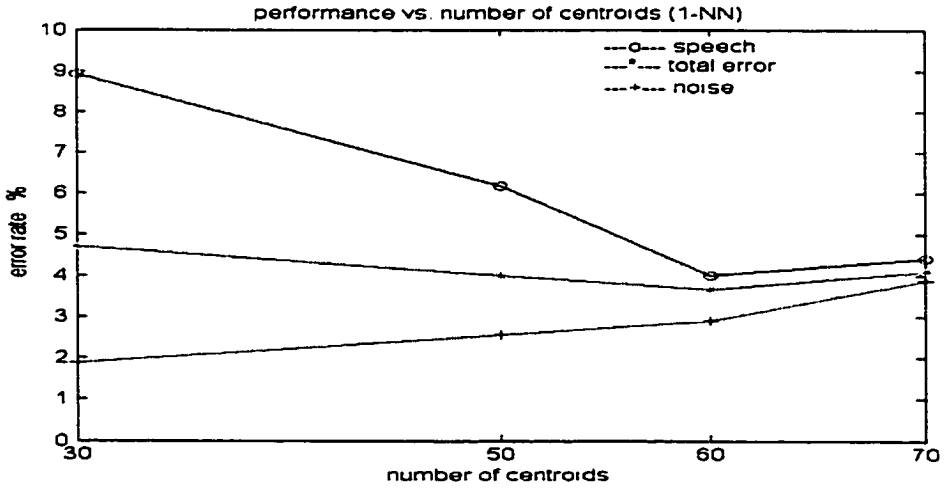


Figure 4-1: Performance of 1-NN under different number of centroids

Some more discussion about the Nearest Neighbour classification is provided here. There are two ways to perform the classification. One is called 1-NN, which was used in the previous table, the other one is called *K*-NN. 1-NN chooses the closest centroid to classify, i.e. it calculates the distance between the input and all the centroids, both speech and noise,

then finds the closest one. If this centroid is a prototype of speech, the input is classified as speech. If the centroid is in the noise class, the input is classified as noise. *K*-NN finds the *k* nearest neighbours first, then chooses the most frequently chosen one among the *k* or *k*% centroids as the result. Here one thing that has to be explained is the frequency (probability). The frequency is based on the training set. During training, the training data are clustered into M_i sub-classes for each class *i*. Each sub-class may have a different number of training points, the percentage with respect to the total number of the training data is the probability that the point occurs in this sub-class. In the *k* nearest neighbour method, the most frequently chosen (higher probability) sub-class is chosen as the result even if it was not the nearest one. If the numbers of centroids for two classes were the same, the frequency can be obtained from the following equation:

$$\text{probability}_{\text{i}}^{\text{j}} = \frac{\text{number of points in the subclass}_{\text{i}}^{\text{j}}}{\text{total number of the training data of class i}}. \qquad (4\text{-}4)$$

where subclass_i^j denotes the j^{th} subclass of class *I*, and probability_i^j denotes the probability that x is in the j^{th} subclass of class *i*. However, if the number of centroids for two classes are different, such as 50 for speech and 30 for noise, the same number of training data distribute among more sub-classes for speech compared with that of noise. To compensate this unbalance between the two classes, the frequency can be modified by the number of centroids in each class.

$$\text{normalized probability}_{\text{i}}^{\text{j}} = \frac{\text{number of points in the subclass}_{\text{i}}^{\text{j}}}{\text{total number of training data of class i}} * \left(\frac{\text{number of centroids}}{\text{of class i}}\right) \qquad (4\text{-}5)$$

For example, if 50 centroids are used for speech and 30 centroids are used for noise, then the normalized probability that a point falls in the subclass *j* of speech is:

$$\text{normalized probability}_{\text{speech}}^{\text{j}} = \frac{\text{number of points in the subclass}_{\text{speech}}^{\text{j}}}{\text{total number of the training data of speech}} * 50 \qquad (4\text{-}6)$$

and the normalized probability that the input is in the subclass *j* of noise is:

$$\text{normalized probability}_{\text{noise}}^{\text{j}} = \frac{\text{number of points in the subclass}_{\text{noise}}^{\text{j}}}{\text{total number of the training data of noise}} * 30 \qquad (4\text{-}7)$$

Applied strictly as defined above, the result of *K*-NN was not better compared with 1-NN. Some modifications had to be made. 3-NN was investigated in details. One method tested was that *k*-NN is separately used for each class, denoted as 3-NN-sep. In this testing method, two candidates are chosen first, one for each class, then the probabilities of these two candidates are compared, and the most frequently chosen one is selected as the result. Another method tested was to choose the majority among 3 nearest neighbours, denoted as 3-NN-major. It favours the nearest one, and adjusts the decision only when the frequency of the second or the third chosen candidate is significantly greater than the nearest one. After comparing all the distances, 3 nearest neighbours among all the sub-classes of both classes are found. They are ordered in increasing order according to the distance, as 1,2,3. If the three candidates are all in the same class, it is an easy case. Otherwise, if the nearest one and the second (or the nearest one and the third) are in the same class, the decision is set to the nearest unless the probability of the third (or the second) is larger than the sum of the nearest and the second (or the nearest and the third). The result does not need to be flipped when the nearest one is in a different class from both the second and third, because in some cases this is very likely occurring in a remote area. The following is the result using different testing methods and under different numbers of centroids. Again only the 10^{th} order LSFs were used. The errors are presented in percentage.

centroids	testing method	error for speech (%)	error for noise (%)	total error (%)
30+30	1-NN	8.9167	1.8889 ✓	4.7
	3-NN	18.50	5.667	10.80
	3-NN-sep	16.91	2.94	8.533
	3-NN-major	8.416	0.333	3.5667
50+30	1-NN	6.1667	2.5556	4.00
	3-NN	6.667	6.556	6.60
	3-NN-sep	10.9167	2.722	6.00
	3-NN-major	5.6667	0.333 ✓	2.4667
60+30	1-NN	4.0 ✓	2.9	3.3667 ✓
	3-NN	8.167	5.778	6.733
	3-NN-sep	9.833	2.722	5.567
	3-NN-major	3.25 ✓	0.50	1.60 ✓
70+30	1-NN	4.417	3.889	4.10
	3-NN	6.417	8.556	7.70
	3-NN-sep	8.00	3.889	5.533
	3-NN-major	4.250	0.778	2.1667
Notes:	training set: speech : noise =1800 : 1800 testing set : speech : noise = 1200 : 1800 testing method : several types of Nearest Neighbour features: the 10^{th} order LSFs			

Table 8: Experimental results of Nearest Neighbour using different testing methods for classifying clean speech vs. noise

As shown in the table, the number of centroids for speech and noise affects the overall performance significantly. They present the same changing trend. 60 for speech and 30 for noise is a good choice. It shows that in most cases 3-NN-major testing has better performance than 1-NN.

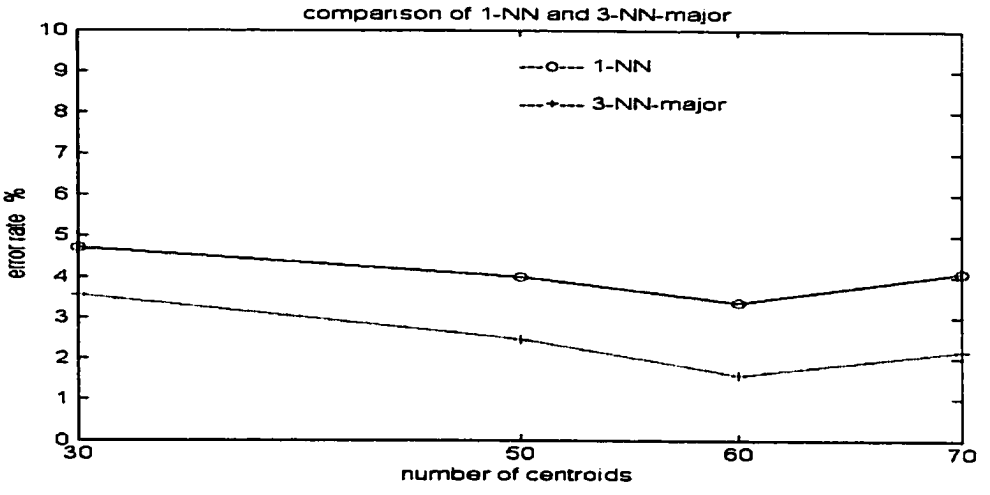


Figure 4-2: Comparison of 1-NN and 3-NN

4.2.3 Quadratic Gaussian Classifier:

Quadratic Gaussian classifier is a remarkable method in pattern recognition. It assumes that the features are Gaussian distributed in the vector space. As explained in Chapter 3, the decision function is Eq. (3-4). However in practice, the classes can be highly overlapping, and the standard deviations of the distributions are highly different. The prototype idea can be imported from the Nearest Neighbour algorithm. First the whole volume of the feature area is divided into subclasses with comparable standard deviations. It assumes that the feature points in each sub-class are Gaussian distributed. Each sub-class has a mean and a standard deviation. Knowing these conditions the probability can be calculated using Eq. (3-6) and the decision function Eq. (3-5) is used to find the biggest probability as the result, i.e. $x \in \omega_j$ when $p(\omega_j)f(x|x\in\omega_j)=\max_{i=1:M} p(\omega_i)f(x|x\in\omega_i)$, in which $p(\omega_i)$ is the a priori probability that we choose x from class i. $p(\omega_i)$ is calculated in section 3.3. If the resulting maximum is in a speech area, the input is classified as speech. Otherwise, it is set as noise.

In fact, through the decision function of the QG classifier, the method designed a set of hyper-volumes, bounded by the interface of several Gaussian distributions. Similar to the Nearest Neighbour method, the training data are clustered into sub-classes first. However, in QG the ratio of the numbers of sub-classes for speech and noise is different from the

Nearest Neighbour method because the decision mechanisms are different between these two algorithms. For a Gaussian distribution, the probability distribution function is related with the standard deviation, if a standard deviation is smaller, the corresponding $f(x)$ for the same x will be bigger as shown in the following figure. Even if the distance to one centroid is small it still will be classified as another class. This is the reason that QG classifiers were found to produce a better performance for speech, whereas the Nearest Neighbour classifiers usually produce a better result for noise. So the preferred clustering would sub-classify the data to make $p(\omega_i)f(x|x \in \omega_i)$ comparable (not significantly distinct).

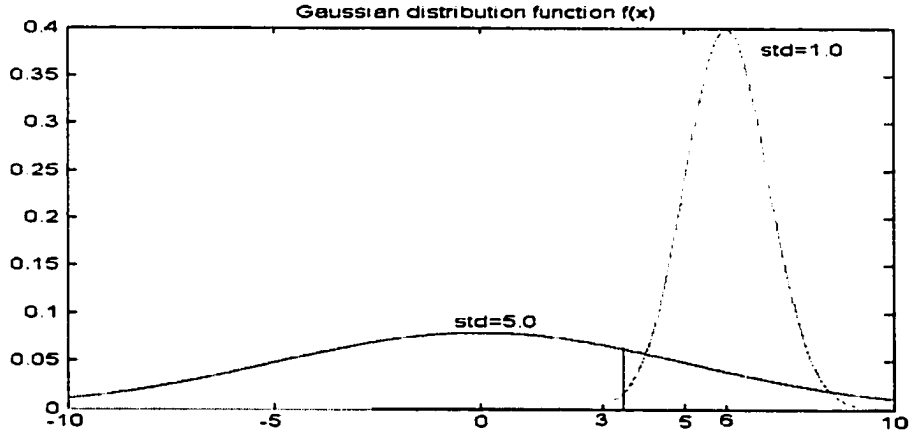


Figure 4-3: Figure of Gaussian distribution functions

The testing results are shown in the following table. Using QG a better performance was usually obtained for speech.

number of centroids speech + noise	error for speech (%)	error for noise (%)	total error rate (%)
20+20	2.25	3.167	2.80
30+30	1.333 ✓	3.50	2.633
20+30	1.50	2.44 ✓	2.067 ✓
20+40	2.167	3.722	3.10
30+40	1.583	4.44	3.30
Notes: training set: speech : noise =1800 : 1800 testing set : speech : noise = 1200 : 1800 testing method : Quadratic Gaussian; features: the 10^{th} order LSFs			

Table 9: Experimental results of Quadratic Gaussian classifiers for classifying clean speech vs. noise

4.2.4 Classification Using Neural Networks:

Neural networks are powerful tools for classification. Some algorithms of learning for neural networks were explained in Chapter 3. In practical implementations, there are still some details that need to be addressed. The main problems are convergence, generalization, and speed. Convergence properties come from the learning procedure. During learning, the weights are updated in the neural network according to the steepest descent deviation or some other algorithms. Usually a local optimal is found. In complex cases the distribution surfaces (cost function as a function of the weights) are wigged, mixed of curves and peaks. If the estimation falls in a local optimum, it is difficult to get out for some learning algorithms. This causes the estimated result to be far from the global optimum. Generalization is referred as the capacity for the designed network to produce reasonable outputs for unseen new observations of inputs. It is an important point in practical applications. Speed is also an issue in the experiments. It is measured by the number of epochs used to achieve the convergence goal, where one epoch indicates one cycle where all of the training data and targets are presented to the network to update the weights. From our experiments some learning algorithms needed thousands of epochs to reach the goal, while some others needed only less than 10 epochs. In the following part

we will explain the details of our neural networks results, using the standard Back Propagation algorithm, the Levenberg-Marquadt learning, and the Extended Kalman Filter learning.

1. The activation function:

The data of speech and noise are overlapping in the feature space as shown in Fig. (4-4). It is not easy for a linear classifier to separate the two classes as has been shown in the linear classification results. Non-linear activation functions are preferred. Using non-linear activation functions the neural network can fit the random shaped curves or circles. We can use *logsig* or *tansig* function as in Fig. (3-8). Although it is traditional for Back Propagation network to use *logsig* with activation within [0, 1], there is theoretical and experimental evidence that [-1, 1] activation functions give better results [30]. Using *tansig* as the activation function the output of i^{th} neuron y_i can be either positive or negative. This gives more accurate effects on the modifications. The *tansig* functions were used in the hidden layers, the *purlin* functions were used in the output layer, and a *hardlim* function was also used at the output layer to check the final status of the testing. The activation function and its first derivative are the followings. The structure of neuron used is the same as in Fig. (3-7).

$$y_i = f(net_i) = tansig(net\ i) = \frac{e^{net_i} - e^{-net_i}}{e^{net_i} + e^{-net_i}},$$

$$f'(net_i) = \frac{\partial f(net_i)}{\partial x} = 1 - y_i^2$$

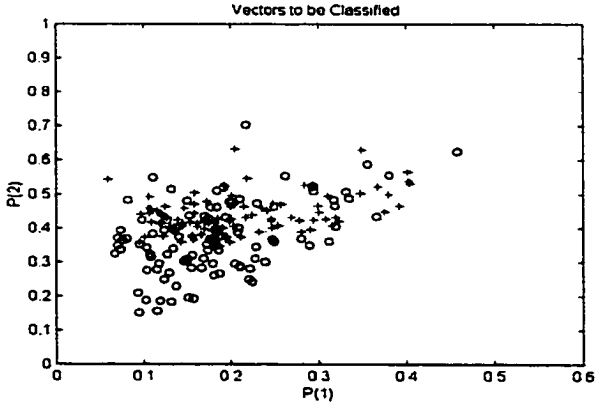


Figure 4-4: The first two dimensions of LSF for speech (o) and noise (+)

2. The stop condition (the goal): In practice, for a complex training data, it is almost impossible for the network to get to the minimum of the error surface. On the other hand a good system does not need a complete zero error rate for in-group data. We should set a condition to let the calculation stop properly. The stop condition was set to be the average of the mean square error E over one epoch. If the E is small enough (0.025), the iterations were stopped.

3. Number of hidden layers and neurons in the network: A one hidden layer structure was tried first. In theory, a one hidden layer neural network structure is enough to track arbitary shape functions. The performance was found acceptable. On the other hand, as shown in Fig. (4-5) for a curve-fitting problem in two dimensions space, although the result for in-group data may be perfect in the right figure, if the fitting line is extreme, then for out-group data the performance will be bad. For a good generation, we usually want a smoother fitting curve.

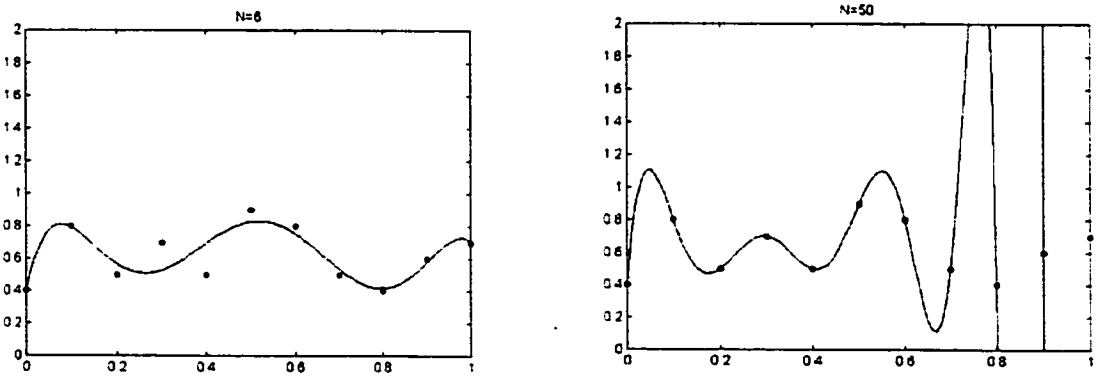


Figure 4-5: An example of curve fitting

The simplest function means the smoothest one that approximates the actual mapping. Therefore, a smaller number of neurons in the hidden layer should be tried and increased gradually until the goal is achieved.

4. The training data: To achieve a good generalization it is hoped that the training data evenly covers all the possible situations including female and male, young and old. So the training data was randomly chosen in the experiments. For the neural

network training, it is preferred to normalize the training data first before sending it into the network, i.e. so that the data fits in the [0, 1] interval. The two classes were trained together with different desired outputs, 1 and 0.

Our results of classification using neural networks are presented next.

Although the Back Propagation is the most common and basic learning algorithm, it converges very slowly. It needs thousands of epochs. In the Matlab Neural Network Toolbox, some faster algorithms are compared with Back Propagation. In the conclusion, it is mentioned that "in general on networks which contain up to a few hundred weights the Levenberg-Marquadt algorithm will have the fastest convergence. This advantage is especially noticeable if very accurate training is required."[22]. So on top of the standard Back Propagation algorithm, the Levenberg-Marquadt algorithm will be used to represent the traditional neural network learning and compared to the Extended Kalman Filter algorithm that we suggest. The results obtained from these algorithms are compared in training speed and generalization that they can achieve.

1. Standard Back Propagation:

Standard Back Propagation learning is the basic neural network learning algorithm. But it converges very slowly, often falls in a local optimal, and needs many trials to get a better result with the same number of epochs. Even with the number of epochs increased up to 1000, the error rate is still around 12%. The results of 200 and 1000 epochs were similar.

number of neurons in the hidden layer	error for speech (%)	error for noise (%)	total error rate (%)
20	20.50	7.1667	12.50
Notes: training set: speech : noise =1800 : 1800 testing set : speech : noise = 1200 : 1800 testing method : Neural Network learning by standard BP features: the 10^{th} order LSFs;			

Table 10: Experimental results of Neural Network classifiers with BP for classifying clean speech vs. noise

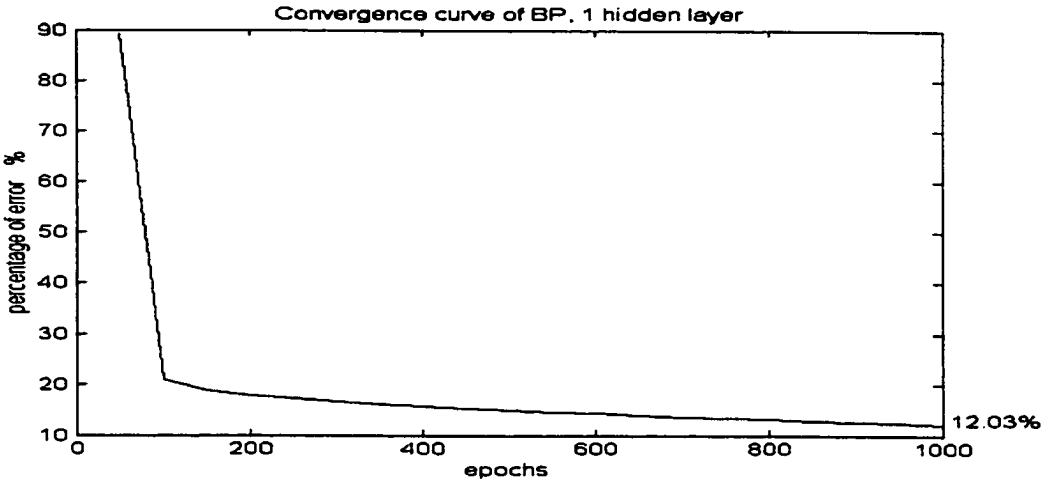


Figure 4-6: Convergence curve of standard Back Propagation learning

So in the future experiments, the standard Back Propagation learning will not be used.

2. Levenberg-Marquadt:

The Levenberg-Marquadt method is a modified learning algorithm based on the standard Back Propagation. It converged fast at the beginning. But after some threshold the convergence curve became almost flat, and increasing the number of epochs a lot did little to improve the performance.

number of neurons in the hidden layer	error for speech (%)	error for noise (%)	total error rate (%)
10	6.5833	1.833	3.733
15	5.25	2.11	3.3667
20	5.167 ✓	1.833	3.167 ✓
25	6.25	1.944	3.667
30	5.917	1.611 ✓	3.333
Notes: training set: speech : noise =1800 : 1800 testing set : speech : noise = 1200 : 1800 testing method : Neural Network learning by Levenberg-Marquadt features: the 10^{th} order LSFs;			

Table 11: Experimental results of Neural Network classifiers with LM for classifying clean speech vs. noise

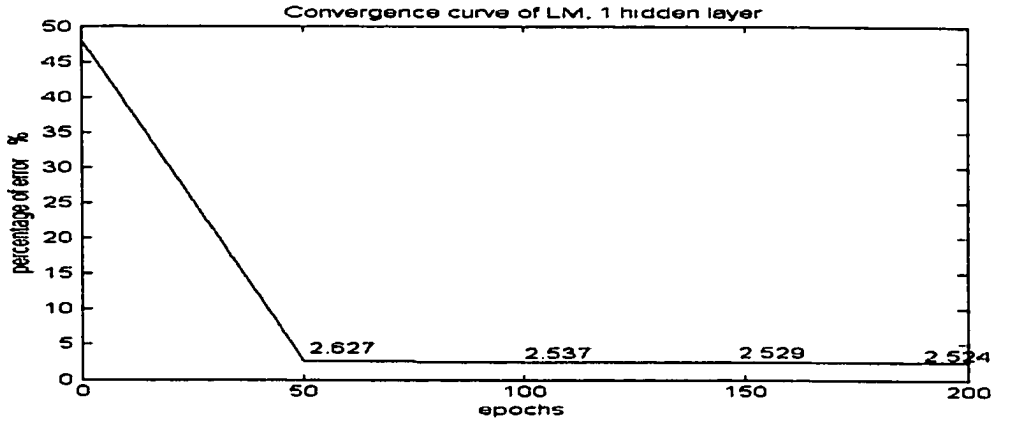


Figure 4-7: Convergence curve of Levenberg-Marquadt learning

3. Extended Kalman Filter:

Using the algorithm explained in Chapter3, the following results were obtained. Clearly, the convergence is fast and reaches a very lower goal. It needs at most 5~10 epochs to achieve the goals.

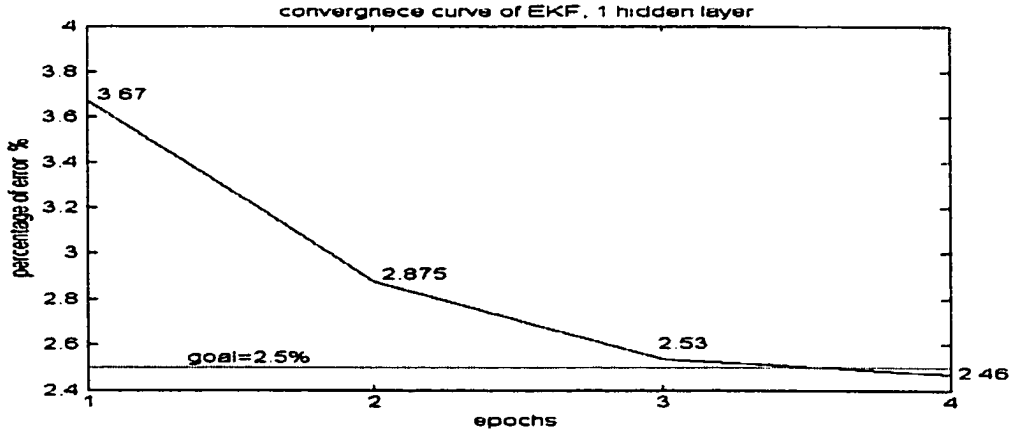


Figure 4-8: Convergence curve of Extended Kalman Filter

number of neurons in the hidden layer	error for speech (%)	error for noise (%)	total error rate (%)
10	4.1667	1.833	2.767
15	4.083	1.3889	2.4667
20	4.1667	1.1111	2.333
25	4.000	1.222	2.333
30	3.833 ✓	0.944 ✓	2.100 ✓
35	3.9167	1.0566	2.200
40	4.6667	0.9444	2.4333

Notes: training set: speech : noise =1800 : 1800
testing set : speech : noise = 1200 : 1800 features: 10^{th} LSF's
testing method : Neural Network learning by Extended Kalman Filter.

Table 12: Experimental results of Neural Network classifiers with EKF for classifying clean speech vs. noise

The experimental results previously presented illustrate that using the same number of neurons in the hidden layer, the Extended Kalman Filter that we propose achieved better results than Levenberg-Marquadt learning in all cases.

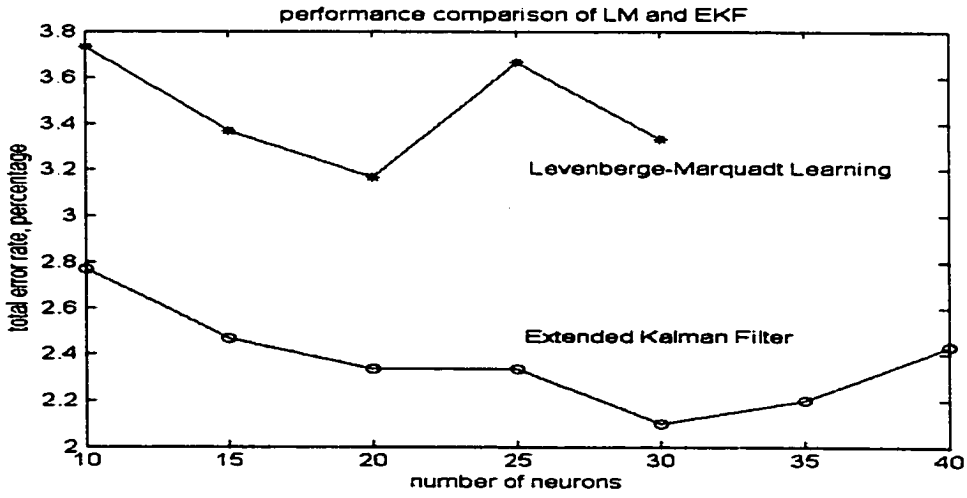


Figure 4-9: Performance comparison of Levenberg-Marquadt and Extended Kalman Filter learning

4.2.5 Comparison of the classification results of all the methods

Comparing all the algorithms in classification of clean speech vs. noise, it is clear that both Nearest Neighbour and QG work well, but the performance of Neural Network classifiers is better. In general, Neural Network classifiers give results in the range 2.1~2.767 under a large range of number of neurons in the hidden layer, while QG has most results above 2.633 % error rate although there was a good result in 20+30 case. Additionally, both Quadratic Gaussian and Nearest Neighbour performance are highly dependent on the number of sub-classes (or centroids). This means that Neural Network classifiers are more robust, less sensitive to the system structure compared with the other algorithms. However, it was also observed that the algorithm used for training the neural networks was critical and high performance algorithms were required. Algorithms such as Levenberg-Marquadt or Extended Kalman Filter algorithm had to be used.

4.3 Experimental results for clean speech vs. music:

The characteristics of different kinds of music are widely different. In this section the classification of clean speech vs. music was tested separately according to the types of music. Classical music and pop music were used in this section. Furthermore, in this section the classification was tested both for the 10^{th} order LSF coefficients and for 13 dimension features. The results showed that the 10^{th} order LSF coefficients are not enough for speech-music classification. The implementations of the classifiers have been explained in detail in section 4.2. In this section, most paragraphs focus on the results only. At the end, a comparison of the performance is presented.

4.3.1 Classification of speech vs. classical music

In this part the experiments of the linear classifiers, Nearest Neighbour, Quadratic Gaussian, and Neural Network classification were tested based on 10^{th} order LSF coefficients or using all 13 features. Because the norms of the energy features are significantly larger than those of LSF and ZC, for the test the parameters were normalized when all the 13 dimension features were used, so that all the features had a similar weight in classification.

4.3.1.1 Linear classifiers:

Using the same method as expressed in section 4.2.1, the classifier performance was evaluated for different dimensions of the feature vectors. The desired output for speech is set to 1, for classical music it is set to 0. In this section, the training data is 1800 frames for each class, and the testing set is 2500 frames per class, in 50%: 50% ratio.

Feature vector used	error for speech (%)	error for classical music (%)	total error rate (%)
ZC+Ef+El	38.86	16.04	27.45
The 10^{th} order LSF	24.5217	21.3043	22.9130
LSF +ZC	24.3478	19.2174	21.7826
LSF+ZC+Ef+El	19.8696 ✓	9.2174 ✓	14.5435 ✓
Notes: training set: speech: classical music =1800 : 1800 frames testing set : speech : classical music = 2500 : 2500 frames testing method : Linear classifier;			

Table 13: Experimental results of linear classifiers for classification of clean speech vs. classical music

4.3.1.2 Results from the Nearest Neighbour classifiers:

First, the 10^{th} LSF coefficients were used as the only features for classification. Then the experiments were done using the 13-dimension feature vectors.

	1-NN			3-NN-major		
number of centroids	error for speech (%)	error for classical music (%)	total error rate (%)	error for speech (%)	error for classical music (%)	total error rate (%)
20+20	7.72	7.44	7.58	8.8	3.0	5.90
30+30	8.04	5.76	6.9	7.28	2.36	4.82
40+40	7.48	4.16 ✓	5.82	7.16	2.36	4.76
50+50	7.44	5.04	6.24	5.52 ✓	2.0 ✓	3.76✓
50+40	6.12	6.80	6.46	6.40	2.44	4.42
60+40	4.80 ✓	5.96	5.38✓	5.80	2.36	4.08

Notes: training set: speech : classical music =1800 : 1800 frames
testing set : speech : classical music = 2500 : 2500 frames
testing method : Nearest Neighbour; features: 10th order LSF coefficients

Table 14: Experimental results of Nearest Neighbour classifiers for classification of clean speech vs. classical music using 10th order LSFs

	1-NN			3-NN-major		
number of centroids	error for speech (%)	error for classical music (%)	total error rate (%)	error for speech (%)	error for classical music (%)	total error rate (%)
20+20	7.12	4.84	5.98	10.44	2.52	6.48
30+30	5.24	4.08	4.66	5.24	1.44	3.34
40+40	4.52	3.44	3.98	6.40	1.24 ✓	3.82
50+50	5.12	2.24 ✓	3.68 ✓	4.16	1.40	2.78 ✓
60+60	4.44	2.48	3.46	5.12	1.64	3.38
50+40	4.60	3.44	4.02	3.96 ✓	2.24	3.10
60+40	4.04 ✓	3.96	4.0	5.12	2.8	3.96

Notes: training set: speech : classical music =1800 : 1800 frames
testing set : speech : classical music = 2500 : 2500 frames
testing method : Nearest Neighbour; features: 10th order LSF +ZC+Ef+El

Table 15: Experimental results of Nearest Neighbour classifiers for classification of clean speech vs. classical music using 10th order LSFs , ZC, Ef, and El

The results showed that using all the 13 dimension features, the classification results were improved a lot.

4.3.1.3 Results from the Quadratic Gaussian classifiers:

The results of Quadratic Gaussian classification are presented in the following table:

	The 10^{th} order LSF			10^{th} order LSF+ZC+Ef+El		
number of centroids	error for speech (%)	error for classical music (%)	total error rate (%)	error for speech (%)	error for classical music (%)	total error rate (%)
20+20	3.88	3.52	3.70	1.12	6.88	4.00
30+30	3.32 ✓	4.32	3.82	1.48	8.92	5.20
40+40	3.80	4.40	4.10	1.60	18.76	10.18
50+50	3.64	3.04 ✓	4.34	/	/	/
20+30	3.68	3.64	3.66 ✓	0.88 ✓	10.88	5.88
30+20	3.64	4.92	4.28	1.88	4.26 ✓	3.32 ✓
Notes: training set: speech : classical music =1800 : 1800 frames testing set : speech : classical music = 2500 : 2500 frames testing method : Quadratic Gaussian;						

Table 16: Experimental results of Quadratic Gaussian classifiers for classification of clean speech vs. classical music

They show that the distributions of the features of classical music, especially the energy features, are not quite Gaussian. For the classification of speech vs. classical music the Quadratic Gaussian classifiers do not work very well.

4.3.1.4 Neural Network classifiers:

The classification results using neural networks show that using the 10^{th} order LSF coefficients only the results obtained through the Neural Network classifiers are not as good as those from the Nearest Neighbour classifiers. The following table shows that the

results of using all the 13 dimension features improved a lot. In general they were better than the results from the Nearest Neighbour using 13 dimension features.

	Levenberg-Marquadt learning			Extended Kalman Filter		
number of neurons	error for speech (%)	error for classical music (%)	total error rate (%)	error for speech (%)	error for classical music (%)	total error rate (%)
10	9.44	5.12	7.48	7.65	7.19	7.42
15	8.04	4.40	6.22	6.72	5.16	5.94
20	6.36	5.16	5.76	7.04	4.72	5.88
25	7.36	3.56 ✓	5.46	6.04	4.48	5.26
30	5.90	4.20	5.06 ✓	4.24 ✓	4.0 ✓	4.12 ✓
40	5.28 ✓	4.92	5.10	5.32	4.28	4.80
Notes: training set: speech : classical music =1800 : 1800 frames testing set : speech : classical music = 2500 : 2500 frames testing method : Neural Networks; features: 10^{th} order LSF coefficients						

Table 17: Experimental results of Neural Network classifiers for classification of clean speech vs. classical music

	Levenberg-Marquadt learning			Extended Kalman Filter		
number of neurons	error for speech (%)	error for classical music (%)	total error rate (%)	error for speech (%)	error for classical music (%)	total error rate (%)
10	6.36	4.8	5.58	3.64	2.80	3.22
20	4.60	2.24 ✓	3.42	3.76	3.44	3.60
30	3.20 ✓	2.44	2.82✓	3.28 ✓	3.04	3.16✓
40	4.16	2.32	3.24	3.68	2.64 ✓	3.16
50	3.64	2.80	3.22	3.88	2.68	3.28
Notes: training set: speech : classical music =1800 : 1800 frames testing set : speech : classical music = 2500 : 2500 frames testing method : Neural Networks; features: 10^{th} order LSF +ZC+Ef+El						

Table 18: Experimental results of Neural Network classifiers for classification of clean speech vs. classical music using LSF, Zc, Ef and El

4.3.2 Experimental results for clean speech vs. pop music:

This sub-section presents the results of classification for speech vs. pop music.

4.3.2.1 Linear classifiers:

Using the same method as expressed in section 4.2.1, the classifier performance was evaluated for different dimensions of the feature vectors. The desired output for speech is set to 1, for the pop music it is set to 0. In this section, the training data is 1800 frames for each class, and the testing set is 2500 frames per class, in 50% : 50% ratio.

Feature vector used	error for speech (%)	error for pop music (%)	total error rate (%)
ZC+Ef+El	36.32	13.01	24.68
The 10^{th} order LSF	22.48	23.44	22.96
LSF +ZC	23.56	23.32	23.44
LSF+ZC+EF+El	12.44 ✓	8.36 ✓	10.40 ✓
Notes: training set: speech : pop music =1800 : 1800 frames testing set : speech : pop music = 2500 : 2500 frames testing method : Linear classifier;			

Table 19: Experimental results of linear classifiers for classification of clean speech vs. pop music

4.3.2.2 Results from the Nearest Neighbour classifiers:

The results of the classification using the 10^{th} LSF coefficients were presented, followed by the results of using all the 13 dimension features.

	1-NN			3-NN-major		
number of centroids speech+pop	error for speech (%)	error for pop music (%)	total error rate (%)	error for speech (%)	error for pop music (%)	total error rate (%)
20+20	16.88	10.28	13.58	14.60	4.16	9.38
30+30	14.92	9.00	11.96	12.32	4.08	8.20
40+40	14.00	8.32	11.16	11.52	3.36 ✓	7.44
50+50	12.44	8.60	10.52 ✓	9.24 ✓	4.0	6.62 ✓
60+60	13.68	7.52 ✓	10.60	11.40	3.44	7.42
30+20	13.16	12.56	12.86	8.56	8.04	8.30
40+30	11.64 ✓	10.16	10.90	9.92	5.16	7.54

Notes: training set: speech : pop music =1800 : 1800 frames
testing set : speech : pop music = 2500 : 2500 frames
testing method : Nearest Neighbour; features: 10^{th} LSF coefficients

Table 20: Experimental results of Nearest Neighbour classifiers for classification of clean speech vs. pop music using LSFs only

	1-NN			3-NN-major		
number of centroids speech+pop	error for speech (%)	error for pop music (%)	total error rate (%)	error for speech (%)	error for pop music (%)	total error rate (%)
20+20	8.92	2.92 ✓	5.92	7.36	1.20 ✓	4.28
30+30	6.92	4.96	5.94	5.88	2.44	4.16
40+40	6.60	3.52	5.06 ✓	4.12	2.00	3.06
50+50	7.28	3.00	5.14	5.00	3.24	4.12
30+20	5.72	4.36	5.04	4.52	2.76	3.64
40+30	5.52 ✓	4.64	5.08	3.40 ✓	2.56	2.98 ✓

Notes: training set: speech : pop music =1800 : 1800 frames
testing set : speech : pop music = 2500 : 2500 frames
testing method : Nearest Neighbour features: 10^{th} order LSF +ZC+Ef+El

Table 21: Experimental results of Nearest Neighbour classifiers for classification of clean speech vs. pop music using LSFs, Zc,Ef, and El

4.4.3 Results from the Quadratic Gaussian classifiers:

The results of Quadratic Gaussian classification are presented in the following table:

	The 10th order LSF			10th order LSF+ZC+Ef+El		
number of centroids	error for speech (%)	error for pop music (%)	total error rate (%)	error for speech (%)	error for pop music (%)	total error rate (%)
10+10	2.84 ✓	37.76	20.30	0.72	10.40	5.56
20+20	6.68	8.88	7.74 ✓	0.68	10.72	5.70
30+30	10.68	8.16 ✓	9.42	0.84	11.52	6.18
40+40	12.04	8.52	10.28	0.64 ✓	23.12	11.88
20+10	6.40	11.00	8.70	1.44	7.44 ✓	4.44 ✓
30+20	6.52	9.20	7.86	1.16	7.92	4.54
Notes: training set: speech : pop music =1800 : 1800 frames testing set : speech : pop music = 2500 : 2500 frames testing method : Quadratic Gaussian;						

Table 22: Experimental results of Quadratic Gaussian classifiers for classification of clean speech vs. pop music

Again, the features of pop music, especially the energy features, are not quite Gaussian distributed. The results obtained were not as good as those of the Nearest Neighbour classifiers.

4.4.4 Neural Network classifiers:

The neural network classification results appear in the following tables for the 10th order LSF features only and for the 13-dimensions features.

	Levenberg-Marquadt learning			Extended Kalman Filter		
number of neurons	error for speech (%)	error for pop music (%)	total error rate (%)	error for speech (%)	error for pop music (%)	total error rate (%)
10	15.03	9.24	12.12	11.24	12.86	12.04
15	12.14 ✓	9.24	10.68	14.53	8.87	11.72
20	14.03	6.89 ✓	10.44	13.1833	8.36	10.76
25	12.46	9.16	10.81	10.76 ✓	10.56	10.66
30	13.30	7.40	10.35 ✓	12.90	6.85	9.90
40	13.746	7.52	10.62	11.41	6.97	9.18 ✓
50	12.14	9.3153	10.72	12.54	6.53 ✓	9.52
Notes: training set: speech : pop music =1800 : 1800 frames testing set : speech : pop music = 2500 : 2500 frames testing method : Neural Networks; features: 10^{th} LSF coefficients						

Table 23: Experimental results of Neural Network classifiers for classification of clean speech vs. pop music using LSFs only

	Levenberg-Marquadt learning			Extended Kalman Filter		
number of neurons	error for speech (%)	error for pop music (%)	total error rate (%)	error for speech (%)	error for pop music (%)	total error rate (%)
10	6.15	3.50	4.82	4.38	2.71	3.54
20	3.86 ✓	3.18	3.52	2.89 ✓	2.82	2.86✓
30	4.42	2.31	3.36✓	3.82	3.18	3.50
40	4.78	2.03 ✓	3.40	4.30	2.35 ✓	3.32
50	6.35	4.06	5.20	5.27	2.91	4.08
Notes: training set: speech : pop music =1800 : 1800 frames testing set : speech : pop music = 2500 : 2500 frames testing method : Neural Networks; features: 10^{th} LSF+ZC+Ef+El						

Table 24: Experimental results of Neural Network classifiers for classification of clean speech vs. pop music using LSFs, Zc,Ef, and El

4.3.3 Comparison:

4.3.3.1 Comparison of the performance for different sounds:

From the results presented it appears that the classification of speech vs. music is more complex than that of speech vs. noise. The formant envelope changes in a larger range. The 10^{th} order LSF coefficients were not enough to extract the differences among the speech and music. The energy features and zero-crossing feature helped a lot to improve the performance.

4.3.3.2 Comparison of the performance of classifiers:

Comparing the results obtained from different classifiers/ algorithms, it can be seen that in general the Neural Network classifiers are superior to the other classifiers. Also they are not very sensitive to the number of neurons in the hidden layer, whereas the Nearest Neighbour classifiers and Quadratic Gaussian classifiers are all very sensitive to the number of centroids/ subclasses. It means that the Neural Network classifiers are more powerful and robust in classification for clean speech vs. noise/music. Furthermore, the performance of Neural Network classifiers is related to the different learning algorithms. In general the Extended Kalman Filter performs better than both Back Propagation, and the fast learning Levenberg-Marquadt algorithm.

To conclude, the following graph provides a general comparison of the classification among the different classifiers for different sound classifications. It shows clearly the better performance of Neural Network classifiers.

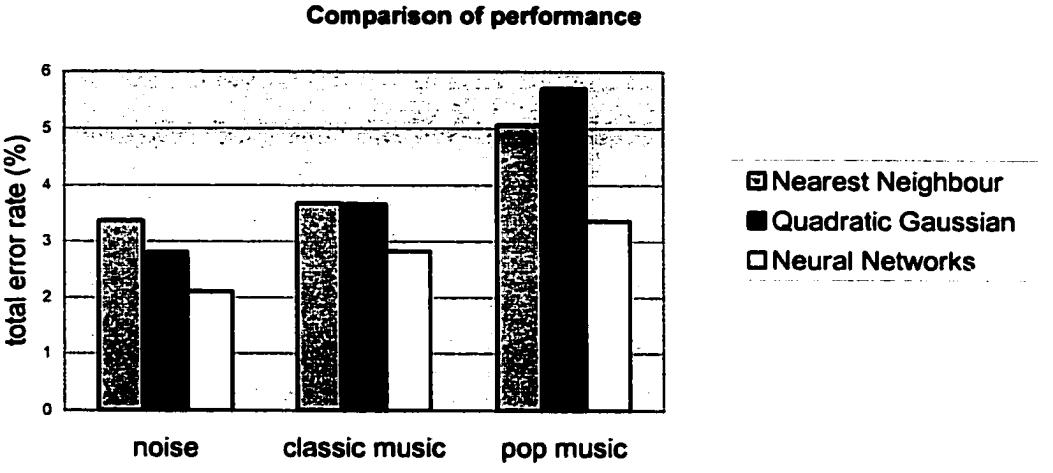


Figure 4-10: Comparison of performance of different classifiers

Chapter 5

Experimental Results for Noisy Speech Classification

In this chapter, the results for classification of noisy speech are presented. The main purpose of this classification is to classify active speech periods from inactive speech/ noise parts in noisy speech under different SNR ratio (Signal to Noise ratio). The experimental environment and database used will be described first, and then the results of all algorithms for noisy speech will be presented. As in Chapter 4, the training and testing were performed using the feature vectors extracted on a frame by frame basis. All test results provided here are hard decisions based on a single frame only. In practice, hangover algorithms are often used to get a soft decision based on the hard decision from the current input frame and results from several previous frames.

5.1 Experimental Environment:

In this section, the method used for obtaining the noisy speech is described. Then the training and testing environments are explained.

The noisy files were obtained by mixing the clean files with noise. As in Chapter 4, 10 female and 10 male clean speech files were used. The original files are in PCM 16 bits format. All the files were converted into 8 kHz 16 bits wave files *.wav* and normalized. Each clean speech file was originally made up with the active and the inactive parts, where active speech is talkspurts and inactive is either the silence or the background noise occurring during the intervals of active speech. In this chapter, the inactive parts were cut from the clean speech files manually. They were almost silence and were used to construct

another file, *inactive.wav*, while all the active parts of clean speech constructed *active.wav*. Next, the clean files, both *active.wav* and *inactive.wav* , were mixed with three kinds of noise: car noise, street noise and classical music, under different *SNR* ratio (20dB, 10dB, 0dB) to construct the noisy files.

The features were extracted using the same methods explained in Chapter 4 using Eq (2-1) to Eq. (2-4). The database was grouped into several sets according to the different *SNR* ratio and the noise types, denoted as:

1. for the active part of noisy speech: active_clean,
   active _car_20dB, active _car_10dB, active _car_00dB;
   active _street_20dB, active _street_10dB, active _street_00dB;
   active _music_20dB; active _music_10dB, active _music_00dB;

2. for the inactive part of noisy speech: inactive_clean,
   inactive_car_20dB, inactive _car_10dB, inactive _car_00dB;
   inactive _street_20dB, inactive _street_10dB, inactive _street_00dB;
   inactive _music_20dB, inactive _music_10dB, inactive _music_00dB.

There were about 4300 frames for each speech set, and 3800 frames for each noise set. For all sets in the database, the order of the feature vectors was randomized. During training, 600 frames from each set of clean, 20dB, and 10dB SNR are chosen to construct the training data, all the remaining frames are testing set. The 0dB SNR data was not used in training because from the experiments we found that they did not improve the performance, and it made the classification more complex. During testing, as in Chapter 4, speech and noise were chosen as 40%: 60% ratio for every *SNR* ratio. In total there were 4200:4200 frames for training, 8000:12000 frames for testing. In the next section the results are presented in error rate with percentage.

5.2 Experimental Results:

In this section, the classification of noisy speech is tested. As previously mentioned, the speech was mixed with car noise, street noise and classical music under *SNR* ratio 20dB, 10dB, and 00dB, respectively.

5.2.1 Linear classifier:

Using the same procedure as in Chapter 4, the noisy speech classification was experimented using different dimensions of feature vectors. During training, clean, 20dB, 10dB *SNR* data were used, with both active and inactive speech. Then during testing, tests were done separately according to the different *SNR*. Note again that no 0dB *SNR* data was used during the training, the results of 0dB SNR as expected are very poor.

features used	testing (SNR)	error for active speech (%)	error for inactive part (%)	total error rate (%)
ZC+Ef+El	clean	35.8	0	14.32
	20dB	29.87	0.022	11.96
	10dB	5.15	78.38	49.10
	00dB	10.83	100	64.33
10^{th} LSF only	clean	29.05	21.03	24.24
	20dB	28.61	24.6	26.21
	10dB	39.88	23.75	30.21
	00dB	43.19	27.60	33.83
10^{th} LSF+ZC	clean	20.95	16.26	18.14
	20dB	23.72	24.51	24.13
	10dB	37.97	24.3	29.77
	00dB	39.23	25.88	31.22
10^{th} LSF + ZC+Ef+El	clean	23.05	0.433	9.48 ✓
	20dB	23.23	1.399	10.13 ✓
	10dB	13.3	39.489	29.01 ✓
	00dB	7.25	97.486	61.39
Notes: training set: active:inactive=600:600 per class (clean, 20dB,10dB) testing set: active : inactive =2000:3000 (40% : 60%) each *SNR* testing classifier: linear classifier				

Table 25: Experimental results of linear classifiers for classification of active speech vs. inactive speech

Using linear classifiers, the performance was complex to analyze. In general, the 10^{th} order LSF coefficients still hold a lot of important information for the classification. But under lower SNR (higher noise level), the spectral envelopes were mixed with noise became flatter, especially the unvoiced speech. The characteristics of active speech were faded. So in lower SNR, the results for active speech were not good, while the results for inactive were good. On the other hand, energy measures had a good effect in higher *SNR* cases,

they improved the performance a lot. But in lower *SNR* cases, because usually active speech has higher energy, when the energy of inactive parts increases with the noise level, the energy measures will produce misclassification for inactive parts. Furthermore, the absolute value of the energy measures are several times higher than LSFs and ZC, so they degrade the performance of 10^{th} order LSFs with high noise levels, when using this simple linear algorithm. Based on the analysis of above experiments, we can see that the effect of the energy measures would be fairer with respect to other features if the energy measures were normalized before they are sent to train when the linear combination were used in algorithms. In section 5.2.4, when we did experiments on Neural Networks using 13 dimensional features all the features were normalized. On the other hand, from the total error rate, it can be seen that the performance still improved when using 13 dimension features in above experiments. So in the following sub-sections, the 13 dimension vectors are used. In section 5.2.4 on Neural Networks with Extended Kalman Filter learning, a test using the 10^{th} order LSFs only will be presented. It shows again that the 10^{th} order LSFs only were not enough for noisy speech classification. So in the most part of the following sub-sections, the 13 dimension vectors are used.

5.2.2 The Nearest Neighbour classifiers:

The algorithms of Nearest Neighbour classifiers, both training and testing, were explained in detail in Chapter 3 and Chapter 4. In this chapter, the Nearest Neighbour classifiers were tested using 1-NN and 3-NN-major. Here different numbers of centroids were tested.

Number of centroids active+inactive	testing (SNR)	error for active speech (%)	error for inactive part (%)	total error rate (%)
50+50	clean	22.85	3.2	11.06
	20dB	30.53	8.47	17.29
	10dB	29.97	17.78	22.65 ✓
	00dB	11.1	97.7	62.9
75+50	clean	21.4	2.9	10.3
	20dB	24.82	7.32	14.31 ✓
	10dB	22.03	32.45	28.28
	00dB	8.93	98.24	62.52
100+50	clean	18.8	2.9667	9.3 ✓
	20dB	22.32	16.08	18.12
	10dB	21.93	22.83	22.93
	00dB	7.63	97.71	62.75

Notes: traing set: active:inactive=600:600 per class (clean, 20dB,10dB)
testing set: active : inactive =2000:3000 (40% : 60%) each *SNR* testing;
testing algorithm: 1-NN.
features: LSF+ZC+Ef+El

Table 26: Experimental results of 1-NN Nearest Neighbour classifiers for classification of active speech vs. inactive speech

Number of centroids active+inactive	testing (SNR)	error for active speech (%)	error for inactive part (%)	total error rate (%)
50+50	clean	32.00	2.367	14.32
	20dB	30.55	1.99	13.41
	10dB	22.92	22.32	22.56
	00dB	10.07	99.87	63.95
75+50	clean	20.3	2.667	9.72
	20dB	22.43	1.478	9.86 ✓
	10dB	14.22	27.62	22.26 ✓
	00dB	41.5	66.62	56.72
100+50	clean	15.10	2.833	7.74 ✓
	20dB	16.367	8.54	11.67
	10dB	8.23	51.68	34.3
	00dB	39.6	66.56	55.83
Notes: traing set: active:inactive=600:600 per class (clean, 20dB,10dB) testing set: active : inactive =2000:3000 (40% : 60%) each *SNR* testing; testing algorithm: 3-NN-major. features: LSF+ZC+Ef+El				

Table 27: Experimental results of 3-NN-major Nearest Neighbour classifiers for classification of active speech vs. inactive speech

It is thus shown that increasing the number of centroids for active speech, the performance for active speech is improved, but at the same time the performance in inactive part is deteriorated.

5.2.3 Results from the Quadratic Gaussian classifiers:

In this part, the results of Quadratic Gaussian classifiers are presented.

Number of centroids active+inactive	testing (SNR)	error for active speech (%)	error for inactive part (%)	total error rate (%)
30+30	clean	1.3	3.8	2.8
	20dB	8.82	8.52	8.64
	10dB	17.82	17.55	17.65
	00dB	2.72	99.75	60.93
50+50	clean	2.55	2.3667	2.44 ✓
	20dB	5.45	14.09	10.6
	10dB	8.95	34.83	24.47
	00dB	1.95	100	60.78
50+30	clean	2.8	2.3	2.5
	20dB	7.38	10.37	9.18 ✓
	10dB	14.7	24.13	20.36 ✓
	00dB	33.3	66.61	53.3
Notes: traing set: active:inactive=600:600 per class (clean, 20dB,10dB) testing set: active : inactive =2000:3000 (40% : 60%) each *SNR* testing; testing algorithm: Gaussian classifier. features: LSF+ZC+Ef+El				

Table 28: Experimental results of Quadratic Gaussian classifiers for classification of active speech vs. inactive speech

Using Quadratic Gaussian classifiers, the performance was improved a lot compared with Nearest Neighbour classifiers. They had better results for active speech. At the same time they had similar results for inactive speech.

5.2.4 Classification using Neural Networks:

Levenberg-Marquadt learning and Extended Kalman Filter algorithm were used for Neural Network classifiers in this section. A one-hidden-layer structure was shown to be enough to deal with this classification. The results showed that both Levenberg-Marquadt and

Extended Kalman Filter learning performed well under different *SNR* ratio. At the end, the performance of the classifier using only the 10^{th} order LSF coefficients is also presented.

Number of neurons in the hidden layer	testing (SNR)	error for active speech (%)	error for inactive part (%)	total error rate (%)
10	clean	4.5	4.0	4.20 ✓
	20dB	12.95	2.26	6.54
	10dB	20.38	15.277	17.32
	00dB	7.48	80.83	51.50
20 ✓	clean	5.85	3.8	4.62
	20dB	12.27	1.57	5.82 ✓
	10dB	19.37	12.85	15.46
	00dB	10.98	69.00	45.79
30	clean	7.15	5.33	6.06
	20dB	13.17	1.73	6.31
	10dB	18.37	12.68	14.95 ✓
	00dB	8.33	79.58	51.08
Notes: traing set: active:inactive=600:600 per class (clean, 20dB,10dB) testing set: active : inactive =2000:3000 (40% : 60%) each SNR testing; testing algorithm: Neural Network, Levenberg-Marquadt learning. features: LSF+ZC+Ef+El				

Table 29: Experimental results of Neural Network classifiers with LM for classification of active speech vs. inactive speech

Number of neurons in the hidden layer	testing (SNR)	error for active speech (%)	error for inactive part (%)	total error rate (%)
10	clean	7.35	3.033	4.76 ✓
	20dB	13.28	1.52	6.23
	10dB	18.62	16.42	17.30
	00dB	7.767	84.04	53.512
15	clean	6.70	4.967	5.66
	20dB	13.61	1.32	6.24
	10dB	17.38	17.87	17.68
	00dB	8.18	69.15	44.76
20 ✓	clean	6.50	4.00	5.00
	20dB	12.48	1.42	5.85 ✓
	10dB	18.26	16.31	17.09
	00dB	9.517	78.76	51.07
30	clean	8.85	3.33	5.54
	20dB	13.55	2.19	6.73
	10dB	17.27	16.00	16.51 ✓
	00dB	5.87	95.11	59.41

Notes: : traing set: active:inactive=600:600 per class (clean, 20dB,10dB)
testing set: active : inactive =2000:3000 (40% : 60%) each SNR testing;
testing algorithm: Neural Network, Extended Kalman Filter learning.
features: LSF+ZC+Ef+El

Table 30: Experimental results of Neural Network classifiers with EKF for classification of active speech vs. inactive speech

Using Neural Network classifiers, a good performance was obtained for most cases, clean speech and higher *SNR* ratio noisy speech such as *SNR*=20dB, 10 dB. For very low *SNR* ratio (0dB), most of inactive parts were still misclassified as active speech (recall that 0dB *SNR* data is not used in training). However compared with other classifiers, the performance is improved.

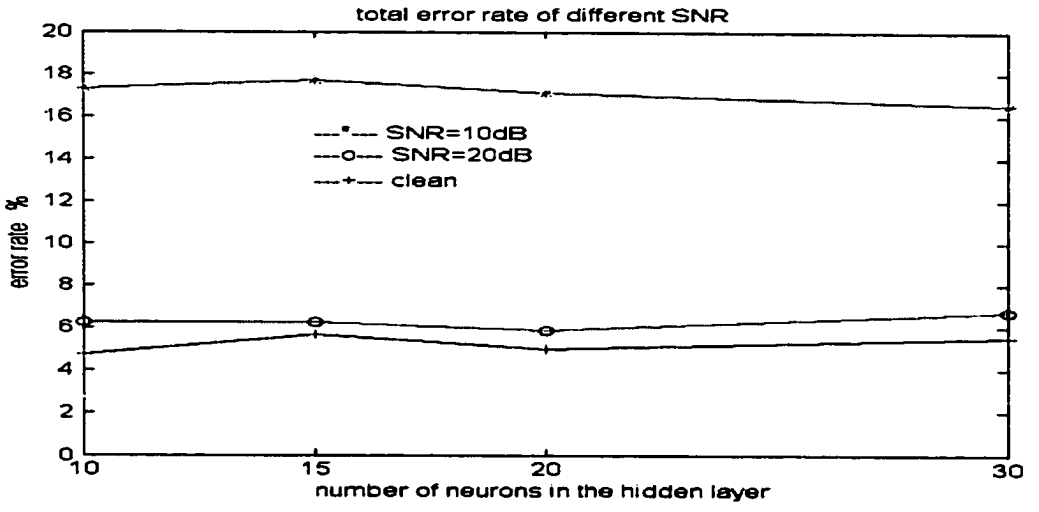


Figure 5-1: Performance of Neural Networks with different number of neurons in the hidden layer (EKL)

For clean speech and high *SNR* noisy speech, the performance of Neural Networks with different number of neurons was compared and shown in Fig. (5-1). For clean speech and 20dB SNR noisy speech, a 20 neurons structure works better, while for 10 dB *SNR* ratio noisy speech, it is preferred to use more neurons in the hidden layer. However the overall performance were not very different.

The following is the result from the Neural Network classifier using the 10^{th} order LSF coefficients only. Compared with the above two tables, it is clear that for classifying noisy speech, using only the 10^{th} order LSF coefficients is not enough. They performed badly in clean speech and 20dB SNR ratio noisy speech. For 0dB noisy speech, although the total error rate was lower, it was obtained by having a higher error rate for active speech.

Number of neurons in the hidden layer	testing (SNR)	error for active speech (%)	error for inactive part (%)	total error rate (%)
10	clean	7.65	19.86	14.98
	20dB	15.82	11.24	13.07
	10dB	34.00	9.6	19.36
	00dB	46.48	11.16	25.29
Notes: traing set: active:inactive=600:600 per class (clean, 20dB,10dB) testing set: active : inactive =2000:3000 (40% : 60%) each *SNR* testing; testing algorithm: Neural Network, Extended Kalman Filter learning. features: the 10^{th} LSF coefficients				

Table 31: Experimental results of Neural Network classifiers with EKF for classification of active speech vs. inactive speech using 10^{th} order LSFs only

5.3 Results Analysis:

The next table compares the performance of all the classifiers. Although Quadratic Gaussian classifier produced the best performance for clean speech, when more realistic data was used in testing (20dB SNR and 10dB SNR), the performance of Quadratic Gaussian classifiers degraded severely. In these noisy situations, Neural Network classifiers with LM and EKF produced a much better performance than all the others. It is clear from the results that Neural Network is the most powerful and robust technique tested. It has a better performance for noisy speech. And it is not too sensitive to the structure of the network, the number of neurons in the hidden layer does not affect the performance greatly. For noisy speech classification using Neural Networks, we found that the performance gain of the extended Kalman Filter over the Levenberg-Marquadt algorithm disappeared (at least for the final solution found by the algorithms). Both algorithms produced a very similar performance.

classifiers	testing (SNR)	error for active speech (%)	error for inactive part (%)	total error rate (%)
Linear classifier (10^{th} LSF+ ZC+Ef+El)	clean	23.05	0.433	9.48
	20dB	23.23	1.399	10.13
	10dB	13.3	39.489	29.01
	00dB	7.25	97.486	61.39
Nearest Neighbour classifiers (100+50,1NN)	clean	18.8	2.9667	9.3
	20dB	22.32	16.08	18.12
	10dB	21.93	22.83	22.93
	00dB	7.63	97.71	62.75
Nearest Neighbour Classifers (75+50, 3_NN)	clean	20.3	2.667	9.72
	20dB	22.43	1.478	9.86
	10dB	14.22	27.62	22.26
	00dB	41.5	66.62	56.72
Quadratic Gaussian classifiers (50+30)	clean	2.8	2.3	2.5 ✓
	20dB	7.38	10.37	9.18
	10dB	14.7	24.13	20.36
	00dB	33.3	66.61	53.3
Neural Network classifiers (20 neurons with EKF)	clean	6.50	4.00	5.00
	20dB	12.48	1.42	5.85 ✓
	10dB	18.26	16.31	17.09 ✓
	00dB	9.517	78.76	51.07 ✓
Neural Network classifiers (20 neurons with LM)	clean	5.85	3.8	4.62
	20dB	12.27	1.57	5.82 ✓
	10dB	19.37	12.85	15.46 ✓
	00dB	10.98	69.00	45.79 ✓
Notes: : traing set: active:inactive=600:600 per class (clean, 20dB,10dB) testing set: active : inactive =2000:3000 (40% : 60%) each SNR testing; features: LSF+ZC+Ef+El				

Table 32: Performance comparison of all the classifiers for classification of active speech vs. inactive speech

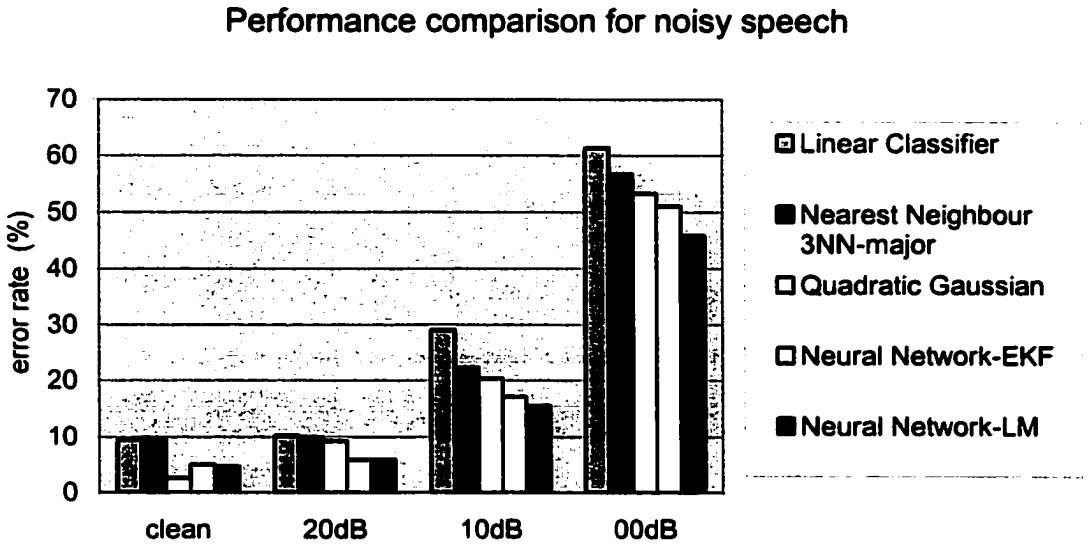


Figure 5-2: Performance comparison for noisy speech

Fig. (5-2) also compares the performance of the methods with the best produced classification results showing the best performance of Neural Networks for noisy speech.

Chapter 6

Computational Complexity

In the previous chapters, several classification algorithms and their performance were analyzed. In this chapter the computational complexity of the algorithms in the testing phase is compared. The testing phase corresponds to the normal or practical use of the classifiers mentioned in this thesis.

Let N denote the total number of sub-classes or centroids for the two classes, and M denote the dimension of the feature vectors. The computational complexity is estimated for one input vector.

1. Linear Classifiers:

The linear classifier gets an optimal weight vector W^* from training. In testing, it uses a dot product $X^T \cdot W^*$ to get the actual output, then compares it with 1 and 0. Here X is an (M+1) by 1 vector (one dimension of the input vector is for the bias), W^* is a (M+1) by 1 vector too. The computational load for each input vector is only a vector dot product. So the complexity C = $\begin{cases} (M+1) \text{ multiplies} \\ M \text{ additions.} \end{cases}$

This load is very low. For example if M=13 as in Chapter 5, then only 14 multiplies per input vector are required. However, the performance of the linear classifiers is very poor.

2. Nearest Neighbour classifiers:

During testing of the Nearest Neighbour classifier, it needs to calculate the distances from the input vector to all the centroids, then compare them. The main computation focuses on computing the distances. It was mentioned in Chapter 3 that the distance from the current input X to one of the centroids, *cent*, can be reduced to an expression:

distance=$2X \cdot cent' + cent \cdot cent'$,

where X is an M dimensional vector, and *cent* is also a M dimensional vector. There are two dot products to be computed for each centroid. So the overall computation load is

C= N (2(M multiplies + (M-1) additions) +1 multiplies + 1 additions)

$$= \begin{cases} (2M+1)N \text{ multiplies} \\ N(2M-1) \text{ additions} \\ \text{plus the comparison among } N \text{ values (and their probabilities in the k-NN algorithm)} \end{cases}$$

As an example, if M=13 and N=150 as in Chapter 5, the number of multiplies per input vector would be 4050 multiplies.

3. Quadratic Gaussian Classifiers:

The discriminant function of Quadratic Gaussian classifiers is $f(x|\ \omega_i\ )p(\omega_i)$. Most of the computation is in calculating the probability $f(x|\ \omega_i\ )$. For each sub-class ω_i, the classifier calculates $f(x|\ \omega_i\ )p(\omega i)$.

$$f(x \mid \omega_i) = \frac{1}{(2\pi)^{M/2}\left|\sum_i\right|^{1/2}} \exp[-\frac{1}{2}(x-\mu_i)^T(\sum_i)^{-1}(x-\mu_i)]$$

Σ_i^{-1} is a M by M matrix, and μ_i is M dimensional vector.

Suppose all the μ_i, Σ_i^{-1}, $\frac{1}{(2\pi)^{M/2}\left|\sum_i\right|^{1/2}}$, and $p(\omega i)$ are pre-calculated and stored in the software (which would require a very significant amount of memory) . Then the calculation complexity is the following:

C= N (M additions +M* (M multiplies + (M-1) additions)+ M multiplies+(M-1) additions+3 multiplies +1 exponential)

$$= \begin{cases} N(M^2+M+3) \text{ multiplies} \\ N(M^2+M-1) \text{ additions} \\ N \text{ exponentials, look-up table.} \\ \text{Plus the comparison among } N \text{ numbers} \end{cases}$$

This computation load is more than M times greater than for the Nearest Neighbour classifiers assuming the same number of sub-classes N, which is not necessarily in the case.

For example if M=13 and N=80 as in Chapter 5, the number of multiplies per input vector would be 14,800 multiplies, on top of a very large amount of memory required.

4. Neural Networks:

Neural networks find a good combination of the weights during the training. In the testing phase, they compute a linear combination before every activation function. Suppose that all the activation functions are *tansig*, and can be designed as a single look-up table, then most of the computational complexity is the linear combinations. Suppose again that M is the dimension of the input feature vectors, and that:

n_1 is the number of neurons in the first hidden layer,

n_2 is the number of neurons in the second hidden layer,

no is the number of neurons in the output layer, with a linear activation function (not *tansig*) (we used *no*=1)

Then for the one hidden layer structure neural networks,

C= n_1 ((M+1) multiplies + M additions)+ ((n_1+1) multiplies + n_1 additions)

$$= \begin{cases} n_1M+2\,n_1+1 \text{ multiplies} \\ n_1\,(M+1) \text{ additions} \\ n_1 \text{ activation functions (look-up table)} \end{cases}$$

For two hidden layers structure neural networks,

$$C = n_1((M+1)\text{ multiplies} + M\text{ additions}) + n_2((n_1+1)\text{ multiplies} + n_1\text{ additions})$$
$$+((n_2+1)\text{ multiplies} + n_2\text{ additions})$$
$$= \begin{cases} n_1(M+n_2+1)+2n_2+1 & \text{multiplies} \\ n_1M+n_2n_1+n_2 & \text{additions} \\ (n_1+n_2) & \text{activation functions (look-up table)} \end{cases}$$

For example, if the one hidden layer structure with M=13 and n1=20 is used (as in Chapter 5), then the number of multiplies per input vector would be 301 multiplies. This is significantly less than the Nearest Neighbour or Quadratic Gaussian classifiers (respectively 4050 multiplies and 14,800 multiplies) on top of producing better classification results.

The computational complexity of the different classifiers is summarized in the following table. It also gives an example of the complexity assuming that the dimension of the feature vector is 13. It is again clear that the complexity of the Quadratic Gaussian method is the highest, above M times those of Nearest Neighbour and Neural Network classifiers. Except for the linear classifiers, the load of the Neural Network classifiers is the lowest, and at the same time, they usually have a better performance with higher robustness.

Parameter used in the example column: M=13			
classifier	computational complexity		example
Linear Classifier	multiply	M+*1*	14
	addition	M	13
	others	one comparison	1
Nearest Neighbour Classifier	multiply	$(2M+1)N$	$27N$
	addition	$(2M-1)N$	$25N$
	others	N number comparisons	N
Quadratic Gaussian Classifier	multiply	$N(M^2+M+3)$	$185N$
	addition	$N(M^2+M-1)$	$181N$
	others	N number comparisons, N look-up table search	N N
Neural Network Classifier (1 hidden layer)	multiply	Mn_1+2n_1+1	$15\ n_1+1$
	addition	$(M+1)\ n_1$	$14\ n_1$
	others	n_1 look-up table search	n_1
Neural Network Classifier (2 hidden layers)	multiply	$n_1\ (M+n_2+1)+2\ n_2+1$	$n_1(14+\ n_2)\ +2n_{2+1}$
	addition	$n_1M+\ n_2\ n_1+\ n_2$	$n_1(13+\ n2)+\ n_2$
	others	$n_1+\ n_2$ look-up table search	$n_1+\ n_2$

Notes(Parameter ranges):

N denotes the total number of sub-classes/ centroids for the two classes, 50 ~ 100;

M denotes the dimension of the feature vectors, 10 ~ 13;

n_1 denotes the number of neurons in the first hidden layer, 10 ~ 30

Table 33: Comparison of the computational complexity of different classifiers

Chapter 7

Conclusion and Future Work

In this thesis, the classification of speech, noise, and music was explored, in particular using the Extended Kalman Filter training algorithm for Neural Network classifiers. The performance was compared with the traditional classification algorithms: the linear classifiers, the Nearest Neighbour classifiers, and the Quadratic Gaussian classifiers. It was also compared with other Neural Network learning algorithms--- standard Back Propagation learning and fast learning algorithm, Levenberg-Marquadt learning. The experimental results presented in the thesis illustrate that in general the Neural Network classifiers found using the Extended Kalman Filter learning produce a remarkable and robust performance for classification of both clean speech vs. noise/music and active speech vs. inactive speech in noisy situation.

7.1 Summary

The applications of classification in speech signal processing and audio signal processing were introduced. Good classification can improve the performance of speech coding and audio coding in reducing the overall bit-rate of codecs or in improving the quality of speech signal processing. These applications require classification of speech vs. noise or speech vs. music in clean or noisy situations. In this thesis these cases were explored.

Chapter 2 provided an introduction of the sound production mechanisms. Through the different production mechanisms, speech, noise, and music have different characteristics. The features that extract these characteristics were explained. In Chapter 3, the algorithms for several types of classifiers were explained. The Extended Kalman Filter was introduced in detail at the end. The algorithm is extended from the standard linear Kalman Filter

algorithm. It extends the power of the Kalman Filter to non-linear situations. The parameters in the algorithm were explained.

Chapter 4 and Chapter 5 presented the experimental results of several types of algorithms in different situations. It showed that Neural Networks with the Extended Kalman Filter learning are usually substantially more robust then the other classifiers. In the noisy speech classification, they also work very well.

Chapter 6 provided a comparison of the computational complexity among the classifiers. Except for the linear classifiers, the Neural Network classifiers have the lowest computational load. The Quadratic Gaussian classifier has the highest computational cost.

To sum up, the Neural Network classifiers were found to be very powerful for classification of speech/noise/music and have a lower computational complexity at the same time. To achieve the best results, using all the features (the 10^{th} order LSF coefficients, zero-crossing rate, full band energy and lower band energy) is preferred, especially in the noisy case.

7.2 Future work:

1. In this thesis most of the work focused on the algorithms. From the experiments we found that besides the algorithms, the features used also highly affected the final erformances. Here we used the 10^{th} order LSF coefficients, ZC, Ef and El features. But the actual effects of each feature were not explored in detail. It is possible that some of them have more weight in extracting the important information, and have more influence in the classification. Classification using other features would also be of interest

2. Another aspect that could be improved is the classification under very noisy conditions (0dB *SNR*). Our results using 0dB *SNR* noisy speech during training were not conclusive, which led to training without 0dB *SNR* speech and poor performance during the testing phase for the 0dB *SNR* noisy speech. Neural

Networks are likely to be a good solution to this problem, but different configurations would have to be tested. There are some other complex algorithms used in related areas, such as Hidden Markov Models (HMM), which may be interesting alternatives for this classification problem.

3. Multimode classification is interesting in practical applications. It classifies the input into several classes (for example male speech, female speech, classic music, pop music, car noise, white noise, etc.). This classification problem could be done by one single classifier, or it could require several classifiers with pair-wise classification in each one (hierarchical architecture of classifiers). Neural Networks would still be a good candidate for this problem.

References:

[1] ITU-T Rec. G.729 Annex B, "A Silence Compression Scheme for G.729 Optimized for Terminals Conforming to ITU-T V.70." 1996, International Telecommunication Union.

[2] ETSI GSM06.32 "Full Rate Speech VAD for Full Rate Speech Traffic Channel", 1998, European Telecommunication Standards Institute.

[3] Khaled El-Maleh, Mark Klein, Grace Petrucci, and Peter Kabal, "Speech / Music Discrimination for Multimedia Application", ICASSP 2000, vol. 4, pp. 2445-2448, Istanbul, Turkey.

[4] Jes Thyssen et. "A Candidate for the ITU-T 4 kbit/s Speech Coding Standard", ICASSP 2001, vol. 2, pp. 681-684, Salt Lake City, USA.

[5] Khaled El-Maleh, Ara Samouelian, and Peter Kabal, " Frame-Level Noise Classification in Mobile Environments", ICASSP 1999, vol. 2, pp. 237-240, Phoenix, USA.

[6] Adil Benyassine, *et al.* " ITU-T Recommendation G.729 Annex B: a Silence Compression Scheme for Use with G.729 Optimized for V.70 Digital Simultaneous Voice and Data Applications", IEEE Communication Magazine, pp. 64-73, September 1997.

[7] Randy Goldberg, Lance Riek, *A Practical Handbook of Speech Coders*, CRC press, 2000.

[8] Course Notes of Digital Signal Processing, ELG 5376, 2001, Dr. Martin Bouchard, School of Information Technology and Engineering, University of Ottawa.

[9] Ben Gold, Nelson Morgan, *Speech and Audio Signal Processing, Processing and Perception of Speech and Music*, John Wiley & Sons, Inc., 2000.

[10] Chris Dobrian, "Music and Artificial Intelligence", http://music.arts.uci.edu/dobrian/CD.music.ai.html.

[11] John G. Proakis and Dimitris G. Manolakis, *Digital Signal Processing, Principles, Algorithms, and Applications*, third edition, Prentice Hall press, 1996.

[12] Bernard Widrow, Samuel D. Stearns, *Adaptive Signal Processing*, Englewood Cliffs, NJ: Prentice Hall, 1985

[13] Jean-Yves Tourneret, "Statistical Properties of Line Spectrum Pairs", Signal Processing, 65, 1998

[14]K.K. Paliwal, B.S. Atal, "Efficient Vector Quantization of LPC Parameters at 24 bits/frame", IEEE Trans. Speech Audio Process, January 1993

[15] J.Y. Tourneret, M. Ghogho, "Line Spectrum Pairs in Pattern Recognition", Proc. EUSIPCO'96, Italy, 10-13 September 1996

[16] Robert M. Gray, "Vector Quantization", IEEE ASSP Magazine, pp. 4-29, April 1984.

[17] John Makhoul, Salim Roucos, and Herbert Gish, "Vector Quantiztion in Speech Coding", Proceedings of The IEEE, vol. 73, No. 11, pp. 1551-1588, November 1985

[18] Sing-Tze Bow, *Pattern Recognition and Image Preprocessing*, Marcel Dekker, Inc. 1992

[19] Evangelia Micheli-Tzanakou, *Supervised and Unsupervised Pattern Recognition, Feature Extraction and Computational Intelligence*, CRC Press, LLC, 2000

[20] S. Haykin, *Neural Networks a Comprehensive Foundation*, Englewood Cliffs, NJ: Prentice-Hall, 1999

[21] Course notes of " Neural Networks and Fuzzy Logic", ELG 5196, 2001, Dr. Emil Petriu, School of Information Technology and Engineerubg, University of Ottwa.

[22] "Matlab Toolbox of Neural Network Reference Guide", 1995, The Math works Inc.

[23] B.M. Wilamoski, "Neural Network Design", http://nn.uidaho.edu/cs/nnn.html, 2001

[24] Sharad Singhal and Lance Wu, "Training Feed-Forward Networks with the Extended Kalman Algorithm", Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 1989, vol. 2, pp. 1187-1190, Glasgow, Scotland.

[25] Greg Welch and Gary Bishop, "An Introduction to the Kalman Filter", TR 95-041, University of North Carolina at Chapel Hill.
http://www.cs.unc.edu/~welch/kalman/kalman_filter/kalman.html

[26] Samir Shah, Francesco Palmieri, and Michael Datum, "Optimal Filtering Algorithm for Fast Learning in Feedforward Neural Networks", IEEE Neural Networks, vol. 5, pp. 779-787,1992.

[27] Simon Haykin, *Adaptive Filter Theory*, Englewood Cliffs, NJ: Prentice Hall,1991.

[28] Thea Ghiselli Crippa *et al.* "A Fast Neural Network Training Algorithm and its Application to Voiced-Unvoiced-Silence Classification of Speech", vol. 1, pp. 441-447, IEEE ICASSP 1991

[29] Jotaro Ikedo, "Voice Activity Detection Using Neural Network", IEICE Trans. Communication, vol. E81-B, No. 12 Dec. pp. 2509-2513, 1998

[30] Stephen I Gallant, *Neural Network Learning and Expert System*, The MIT press, 1994