uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Luc Lamarche

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering (S.I.T.E.)

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Adaptive Environmental Classification System for Hearing Aids

TITRE DE LA THÈSE / TITLE OF THESIS

T. Aboulnasr

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

W. Gueaieb and C. Giguère

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Professor H. Dajani

Professor R. Goubran

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Adaptive Environmental Classification System for Hearing Aids

Luc Lamarche

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Master of Applied Science in Electrical Engineering

Faculty of Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

School of Information Technology and Engineering

University of Ottawa

Your file Votre référence
ISBN: 978-0-494-41672-3
Our file Notre référence
ISBN: 978-0-494-41672-3

Abstract

This thesis develops an adaptive environmental classification system for hearing aids. Two types of classifiers, minimum distance and Bayesian, are modified to include an adaptive layer that allows the classes to split or merge based on changes in the environment. In order to test the adaptability to the environmental change, both systems were first trained using two classes: Speech and Noise, followed by a testing period where, in addition to Speech and Noise, samples from a third class Music were introduced. Both systems were successful in detecting the presence of the new class Music and estimated its parameters obtaining an accuracy that is as high as the accuracy obtained through a non-adaptive supervised learning. In addition, the accuracy for the merging algorithm in both systems also met that of a non-adaptive system. In comparing the two systems, the adaptive Bayesian classification system resulted in a higher accuracy in classifying the environment into the three classes: Speech, Noise, and Music, following the adaptation process.

Acknowledgments

I would like to thank my thesis supervisors, Dr. Tyseer Aboulnasr, Dr. Christian Giguère, and Dr. Wail Gueaieb for their guidance and support throughout this research.

I would like to thank the SPOT group for their help.

I would also like to give special thanks to my parents, Helene and Guy Lamarche, my sisters Lynne and Lianne, and my brother, Denis, for their support and encouragement.

Table of Contents

List of Tables

List of Figures

Acronyms

DSP	Digital Signal Processing
SNR	Signal-to-noise Ratio
BTE	Behind-the-ear
ITE	In-the-ear
ITC	In-the-canal
CIC	Completely-in-the-canal
RIE	Receiver-in-the-ear
RMS	Root mean squared
FFT	Fast Fourier Transform
DFT	Discrete Fourier Transform
ACF	Autocorrelation Function
SSCL	Self-Splitting Competitive Learning
EM	Expectation Maximization
GMM	Gaussian Mixture Model
PDF	Probability Density Function
APV	Asymptotic Property Vector
CPV	Center Property Vector
DPV	Distance Property Vector
HR	Hit Rate
OH	Overall Hit rate
FA	False Alarm rate
PS	Post-Splitting

Chapter 1
Introduction

1.1 Background

The introduction of digital signal processing (DSP) to hearing aids has lead to many advanced features that aim to improve the listening experience for those who have a hearing impairment. These features include noise reduction algorithms, feedback cancellation, directional microphones and datalogging for learning user preferences. Despite all of these advanced features, there is still room for improvement. A report published by Kochkin [1] in 2005 on user satisfaction for hearing aids, issued after the year 2000, showed that 29% of hearing aid users are dissatisfied with their hearing aid devices and 10% stopped wearing the devices altogether.

Much of this dissatisfaction may be caused by the difficulty in selecting the best settings for the hearing aid in the laboratory, during the fitting process [2]. During this process, a general gain-frequency response model based on hearing loss measurements, such as hearing thresholds and discomfort levels, are used to prescribe the gain function of the hearing aid for each user. However, in practice, the optimum gain response depends on many other factors such as the type of environment, the listening situation and the personal preferences for the user. A study by Munkstedt in [3] shows that although some users are satisfied with a general prescription model, other users prefer a customized prescription that does not conform to any basic model. In addition, Munkstedt also shows that even the users that are

satisfied with a general prescription model in a particular environment may not be as satisfied in other environments. Therefore, there is a need for changing hearing aid prescription for different environments. In addition to the gain-frequency prescription, other settings such as noise reduction algorithms, directional microphone, and volume control are also sensitive to the environment. That is, settings that are beneficial for one environment may have no or even negative impact on other environments. For example, the noise reduction algorithm is beneficial while listening to speech that is accompanied by background noises. However if the noise reduction setting is operating while listening to music, the sound quality may be reduced. As the number of settings that can be changed on a hearing aid increases with the new features being added, changing them manually may overextend the mental and motor abilities of the users [4].

New hearing aids are now being developed with automatic environmental classification systems which are designed to automatically detect the current environment and adjust their parameters accordingly [5,6]. This type of classification typically predefines its classes according to the environments nature (speech, noise, music, etc.). A drawback with having predefined classes is that the classes cannot accommodate all environments for each specific user. For example, an automatic classification system having the 4 predefined classes: Speech, Speech in noise, Noise and Music, will most likely fail in situations not exactly conforming to these environments such as listening to music from a car radio while driving or conversing in a café with background music [4]. In these types of environments, the hearing aid settings selected by the automatic classification system may have a negative impact resulting in the user having to change the settings manually.

A more flexible solution is to tune the classes to each particular user's preferences through adaptive means. An adaptive layer can be added to the automatic classification system to allow for new classes to be added or deleted based on the environments that the specific user encounters.

1.2 Thesis Objectives

The objective of this thesis is to develop a classification system framework that can adaptively split and merge its classes based on changes observed in the ongoing environment. With an adaptive system the following issues must be considered: (1) a classifier needs to be selected to accurately classify the environment into a set of classes; (2) a set of features, used to capture the characteristic information in the environment, need to be strategically selected to carry relevant environmental information; (3) a split criterion must be in place to detect if the user is consistently entering a new and different environment than any existing class; (4) a splitting process must be in place to estimate the new class parameters with an accuracy that is comparable to a non-adaptive system trained on the same data; (5) a merge criterion must be in place to detect if two classes should merge together to make room for new classes; and (6) a merge process must be in place to select two classes to merge and accurately estimate the merged class parameters. The rate at which the adaptive layer is updated must also be taken into consideration. The adaptive system should not be sensitive to new environments that are only visited for a short period of time or infrequently. Therefore, the monitoring system that looks for changes in the environment needs to be designed to run at a much slower rate than the classifier itself. For example, a classifier that classifies the environment once per second may only check for changes once every 30 seconds or more.

The thesis topic was chosen based on the limitations that current automatic classification systems have on user specific class selection. This work can be viewed as the first stepping stone to developing a fully adaptive classification system, where many classes can be added or removed to adjust to the specific needs of the hearing aid user. As previously mentioned, not all users encounter the same day to day environments, and therefore may prefer a different set of classes than those selected for a general user. For example, a user that frequently attends hockey games may enjoy the experience of having the noise from the crowd cheering and thus will prefer that noise reduction algorithm be turned off. At the same

time the same user may want the noise reduction algorithm turned on for other types of noise. Thus ideally an adaptive classification system would be able to detect the hockey game environment as a new class allowing a separate set of hearing aid settings to be automatically activated for this specific environment. This class, however, would not be useful and practical for an individual who does not attend hockey games.

1.3 Thesis Outline

The remainder of this thesis is organized as follows.

Chapter 2 provides a review of current hearing aid technology. First, the components that impact hearing loss are described. Different methods used to combat user specific hearing loss are then described followed by the different hearing aid styles that are available. Lastly, the features available on two of the more advanced hearing aids are summarized.

Chapter 3 provides a description of how a basic automatic classification system functions. The topics covered include: a description of acoustic features that may be used to extract characteristic information from audio samples, the basic pattern classifiers that are used to classify the features, the different supervised and unsupervised algorithms for training the classifier, and finally a comparison between features for different classifiers.

Chapter 4 provides a framework for the proposed adaptive classification system. First, the adaptive layout is introduced, followed by the requirements that an adaptive algorithm must meet. The method used for testing the algorithm is then presented. Finally, the measure used for assessing the performance is summarized.

Chapter 5 provides an initial simple solution based on the minimum distance classifier. First, the system is introduced using an ideal data set with clearly distinctive classes. The algorithm is then tested with more realistic features extracted from a bank of audio samples.

Chapter 6 provides a second slightly more advanced solution based on the Bayesian classifier. Similar to Chapter 5, the system is first introduced using the ideal data set followed by the more realistic features.

Finally, Chapter 7 concludes the thesis by summarizing the results obtained for each system, followed by the contributions made to research. Lastly, a list of improvements that can be made to the adaptive classification system is presented for future research projects.

Chapter 2
Review of Hearing Aid Technology

Hearing aids are becoming more sophisticated as technology advances. They started as simple analog amplifying devices decades ago, and have evolved into ever more complex programmable digital hearing aids since the mid 90's. It is important to step back and understand what a hearing aid is trying to accomplish to combat hearing loss. A short overview of the different facets of hearing loss is first presented. A short description of the prescription methods used to fit hearing aids is then introduced, followed by a description of the basic components of a typical device. The chapter is then concluded with a view of two of the most advanced hearing aids currently on the market.

2.1 Impact of Hearing Loss

There are two main types of hearing loss, conductive and sensorineural. Conductive hearing loss is caused by a defect in the middle or outer ear. This type of hearing loss can sometimes be surgically eliminated or corrected using linear amplification. Sensorineural hearing loss is much more complex. The source of the problem is in the inner ear or auditory nerve and, in most cases, cannot be surgically eliminated. The four main hearing problems that a person with sensorineural loss will have are: (1) a decrease in audibility, (2) a decrease in dynamic range, (3) a decrease in frequency resolution and (4) a decrease in temporal resolution [7].

2.1.1 Audibility

A decrease in audibility will affect a person's ability to hear or detect sounds in quiet surroundings. In some severe cases the person may not be able to hear and understand speech unless it is being shouted at close range. In more moderate cases, the person may have difficulty hearing softer phonemes in continuous speech. Certain phonemes may also be confused and mixed up if a person has loss in the higher frequency range. For example, with a loss in high frequency, most commonly 500 Hz to 4 kHz, a person might confuse the oo sound (as in "boot") with the ee sound (as in "beet") since both have very similar low frequency responses and only differ in the higher frequencies. In order to resolve this issue, a hearing aid must amplify sounds using a frequency-dependent gain response according to the specific hearing profile of the user.

2.1.2 Dynamic range

The decrease in dynamic range introduces a much more complex issue. A person with sensorineural hearing loss will generally have a much larger elevation in absolute hearing thresholds, which is the minimum level that is audible, than in loudness discomfort thresholds, which is the level above which sounds are perceived as uncomfortably loud. In other words, a person will require more amplification for soft sounds than for intense sounds. If the intense sounds are amplified as much as soft sounds then they will be perceived as too loud and may be very uncomfortable. In order to resolve this problem, dynamic signal compression or nonlinear amplification is introduced. This essentially gives a higher amplification for soft sounds and gradually less amplification as the level increases. The speed at which the compressor acts is important. A fast-acting compressor will quickly change its gain when the signal level is changed, while a slow-acting compressor will be slow in adapting the gain. If a signal abruptly increases in level, the compressor will not react instantaneously; therefore for a short period of time, the higher level signal will be amplified as if it were low and may be perceived as uncomfortable. To minimize this effect a fast-acting compression is desired. The time it takes the compressor to react to the increase in level is referred to as the attack time. On the other hand, when the signal level abruptly

decreases in level, the compressor will amplify the signal with a low gain as if it were still high in level until the compressor adapts to the new signal level. The time it takes the compressor to react to a decrease in level is referred to as the release time.

2.1.3 Frequency resolution

A loss in frequency resolution means that a person may have difficulty distinguishing sounds that are close in frequency content. As a result, theses sounds may not be individually resolved and could mask each other. The consequence of this in real life is that the listener may have trouble separating noise and speech components, which will therefore decrease intelligibility in noise. This means that a person with a loss in frequency resolution will need a higher signal-to-noise ratio (SNR) compared to a person with normal hearing to achieve the same intelligibility. In current hearing aid technology, it is not generally possible to resolve this problem directly. Therefore, in order to minimize this problem, processes are done in the hearing aid to favor some sounds over others, thereby increasing the SNR for the listener. This may include filtering out the noise electronically before it is delivered to the ear. Directional microphones are also commonly used to minimize the noise from non-desired directions from entering the hearing aid. The use of these techniques, therefore, requires some assumptions about the nature of unwanted noise and desired signals (speech, alarm, music, etc.) in any given environment or situation.

2.1.4 Temporal resolution

Another problem that will affect speech intelligibility is the temporal resolution. People with sensorineural hearing loss will have difficulty hearing soft sounds that immediately precede or follow intense sounds. The most effective way to address this problem is to introduce fast-acting compression [8]. This will allow the weak sounds to rapidly be amplified while the louder sound's amplification is rapidly decreased. This, however, will allow weak background noises to also be amplified during pauses in conversational speech, introducing a

new problem. Too fast compression also introduces waveform distortion and reduces important temporal modulation cues in the speech envelope.

2.2 Prescription Models

When a hearing aid patient is fitted with a new hearing aid, the device is programmed for the specific user requirements and hearing profile. The audiologist will typically prescribe a gain-frequency response based on the patient's hearing loss. There are many different types of prescription models that may be used for this purpose; some of which are solely based on hearing thresholds while some are also based on loudness perception. Examples of general prescription methods include NAL (National Acoustic Laboratories), NAL-R (NAL revised), NAL-NL (NAL Non-Linear), DSL, and FIG6 [7]. Hearing aid manufacturers also commonly supply product-specific prescription or fitting algorithms for their devices.

Each prescription method is different in finding the optimum gain-frequency response based on hearing thresholds. This, however, is not a straightforward task since it is very difficult to find the optimum prescription based solely on hearing thresholds. This is because the optimum response may also depend on factors such as the type of environment, components of the hearing loss, and personal preferences. For example, if the signal is pure speech the prescription should generally aim to optimize the gain-frequency response to maximize speech intelligibility or audibility. On the other hand, if the signal is music, the optimum gain-frequency response should aim to maximize sound quality. The optimum adjustment of other components of the hearing aid, such as noise reduction algorithms and directional microphones, also depends on the environment or specific listening situation. It is therefore not possible to optimize the listening experience for all environments using a fixed set of parameters for the hearing aid. It is widely agreed that a hearing aid that changes its algorithm or features for different environments would significantly increase the user's satisfaction [9]. This typically requires user interaction to switch listening modes, but increasingly hearing aids are being developed to automatically detect the current environment and adjust the parameters in the hearing aid accordingly.

2.3 Hearing Aid Style and Structure

The basic components of a hearing aid include a microphone, an amplifier, volume control, a receiver and a battery. Digital hearing aids also include an analog to digital converter, a DSP that can be programmed to achieve the desired gain-frequency and compression functions, and a digital to analog converter. A directional microphone is sometimes used to limit the noise from non-desired directions. By including a DSP, many new advanced features may also be introduced to the hearing aid. The features may include adaptive steering of the spatial noise notch, feedback cancellation, and noise reduction algorithms.

There are six styles of hearing aids available [7]. The largest in size is the body aid. This type of device has the bulk of its components in a pack that is worn somewhere on the body and is very rarely used nowadays. The next largest device is the behind-the-ear (BTE) hearing aid (Figure 2.1). The bulk of the hearing aid components are placed behind the ear and sound is delivered from a miniature speaker or receiver to the ear via a small tube attached to an earmold. The next three styles are smaller in size and are fitted directly inside the ear. The largest of the three is the in-the-ear (ITE) hearing aid (Figure 2.1), which fills partly or entirely in the concha cavity. In-the-canal (ITC) hearing aid (Figure 2.1) is smaller and fits mostly inside the canal with only a small section extending out into the concha. The smallest is the completely-in-the-canal (CIC) hearing aid (Figure 2.1) which fits entirely in the canal and is the least visible. A new style, referred to as receiver-in-the-ear (RIE), has recently been introduced which is similar to a BTE, except that the receiver is located in the earmold. There are many issues to consider in selecting a hearing aid style for a particular user. The largest styles can fit more complex components, powerful amplifiers, and special features such as directional filtering. On the other hand, the smallest styles are less disruptive of the acoustic field around the ear and are generally considered more esthetically acceptable. However, some features like directional microphones are not available or not as effective in the smallest devices.

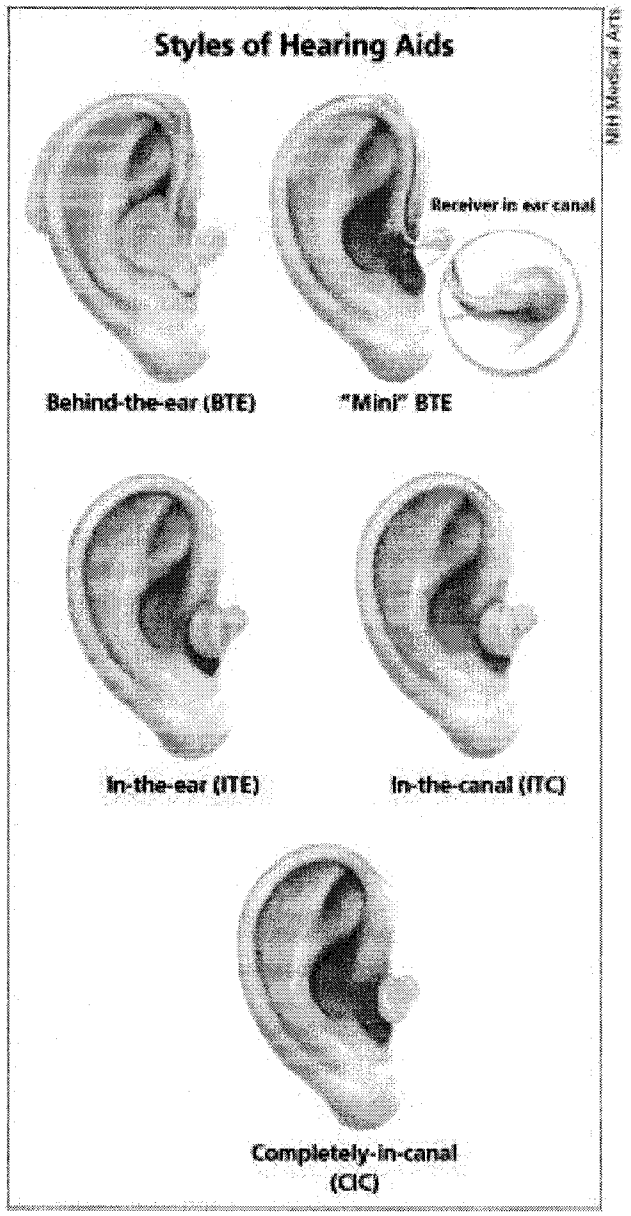


Figure 2.1: Hearing aid styles [10]

Data logging is used in some hearing aids to track the user control preferences in order to adapt itself to the user. One of the most advanced features that is starting to be seen in today's hearing aids is an environmental classification system that can alter the signal processing according to the specific environment.

2.4 Current Hearing Aid Systems

Today's hearing aids are very sophisticated and have several features that aim to increase the user's satisfaction. The following is a description of two of the more advanced hearing aids that can be found in today's market: Siemens' Centra and Phonak's Savia Art.

2.4.1 Siemens' Centra

Siemens' Centra is one of Siemens' most advanced hearing aids and was introduced in 2006. Table 2.1 describes its main features [5]:

Table 2.1: Siemens' Centra features

Feature	Description
Automatic Classification	Automatically changes programs depending on the environment. It recognizes 4 environments: Speech, Speech in noise, Noise, and Music.
Sound Smoothing	This feature aims to reduce the more annoying noises such as rustling paper, breaking glass or fireworks.
DataLearning	The Centra logs the user's preference in volume control and uses the data to automatically adjust itself.
e2e (ear-to-ear) wireless technology	The Centra may share limited information from both ears to keep both hearing aids properly balanced.
State-of-the-art digital sound processing features	These features include feedback cancellation, speech enhancement, and a noise reduction system.
Directional microphone	This allows the user to cancel out noises coming from non-desired directions.

2.4.2 Phonak's Savia Art

Another leading manufacturer in the hearing aid market is Phonak. One of their high end models is the Savia Art and was also introduced in 2006. Its main features are described in Table 2.2. [6]

Table 2.2: Phonak's Savia Art features

Feature	Description
Autopilot	This feature automatically classifies the environment the user is in and allows the hearing aid to switch to a more appropriate program. It recognizes six environments which are: quiet, speech in noise, noise, music, telephone and wireless accessories.
SoundRelax	This feature allows to recognize annoying sounds such as rustling paper and instantaneously reduces the volume.
SoundCleaning	This feature removes unwanted noises, such as feedback whistling, disturbing echoes and wind noises, from the desired signal.
Directional microphone	It also contains a directional microphone which is referred to as digital SurroundZoom.
Self Learning	This device also contains a feature that uses logged volume control data to automatically adjust itself.

These two modern hearing aids employ automatic classification of the environment to maximize listener satisfaction in a range of situations. However, since these classification systems are static in nature, the classes must be specified a priori and may or may not be relevant to the particular user. Also there is little scope for adapting the system or class set after training or for different individuals. This topic of classification is the motivation for this thesis and is, therefore, reviewed in more detail in the following chapter.

Chapter 3

Review of Classification using feature extraction

3.1 Introduction

In order for a hearing aid to automatically change its properties for optimal listening, it must be able to recognize what type of environment the user is in. This is typically done in three steps as shown in Figure 3.1. First, the sound signal received by the hearing aid is sampled and converted into a feature vector via feature extraction. This step is a very crucial stage of classification since the features are chosen to contain the information that will distinguish the different types of environments. The resulting classification accuracy highly depends on the selection of appropriate features [11]. The second step in classification is the pattern classifier itself. The pattern classifier makes a decision on what class the input belongs to based on its features. Finally, the post processing step acts as a filter, to remove spurious jumps in classification to yield a smooth class transition. These three steps are described in more detail below.

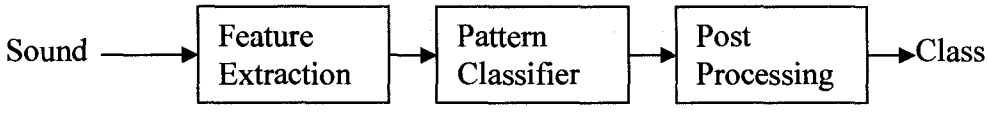


Figure 3.1: Classification System

3.2 Feature Extraction

Feature extraction is the process of capturing characteristic information from the signal. By comparing features from different types of sounds in different environments it is possible to classify the source into one of many classes. There are a large number of features that have been implemented and published in various papers on sound classification for different applications [9, 12, 13, 14, and 15]. In fact, it seems that every paper uses a different set of features. Of particular relevance, Dr. Michael Büchler published his PhD thesis in 2002 on algorithms for sound classification in hearing instruments [11], where many of the more promising features were analyzed and compared.

The features are typically calculated on small frames of about 5-20ms long [12]. Kates in [9] uses four features, based on envelope modulation and spectral structure content, in an attempt to classify 11 different background noises. This allows for different noise reduction algorithm to be applied to different noise groups. Kates achieves accuracy of 90% and higher depending on the number of classes. This high accuracy is misleading since the training data and testing data are extracted from the same noise signals. Nevertheless, the features used are very useful and can be adapted to classify general environments instead of noise types. Ludvigsen in [16] uses amplitude statistics and their distribution shape to create five features in order to classify background sounds. Ostendorf in [17] uses modulation spectra as features to classify sounds into the three classes: speech, speech and noise, and noise.

Büchler in [11] designed a classification system based on the features found in all three of these papers. He compared these features using different types of classifiers to assign sounds into one of four classes, which are: speech, speech in noise, noise, and music. According to Büchler, a good feature should have the following three characteristics [11]:

1) A large inter-class mean distance and a small intra-class variance;
2) No correlation to other features; and
3) Insensitivity to the signal level, unless the feature characterizes the level itself.

The features described below were found to be the most useful.

3.2.1 Features derived from Amplitude statistics

Originally proposed by Ludvigsen in [16], Büchler uses the amplitude statistics as a source for extracting features. He first divides the envelope of the signal into small frames, of approximately 10 to 15 ms in length. The envelope level or amplitude is then measured (in dB) for each frame. Features are then extracted from the amplitude envelope histogram. Büchler shows that the amplitude envelope histogram for noises and certain types of music will have a narrow and symmetrical distribution. Sounds that are closer towards clean speech are more likely to have a broad and asymmetric distribution.

In order to extract features from the amplitude envelope distribution, percentiles are used as measure thresholds. For example, the 10% measure, P_{10}, determines the level below which the envelope is 10% of the time. In other words, if the envelope of a signal is divided into 100 frames and if $P_{10} = 50$ dB, then this means that the amplitude envelope is below 50 dB in 10 of the frames and above 50 dB in the remaining 90 frames. By comparing the levels of different percentiles, the broadness of the distribution is determined. Accordingly, a broad distribution has a large distance between P_{10} and P_{90} compared to that of a narrow distribution. Figure 3.2 shows the placement of the percentiles in the amplitude envelope distribution.

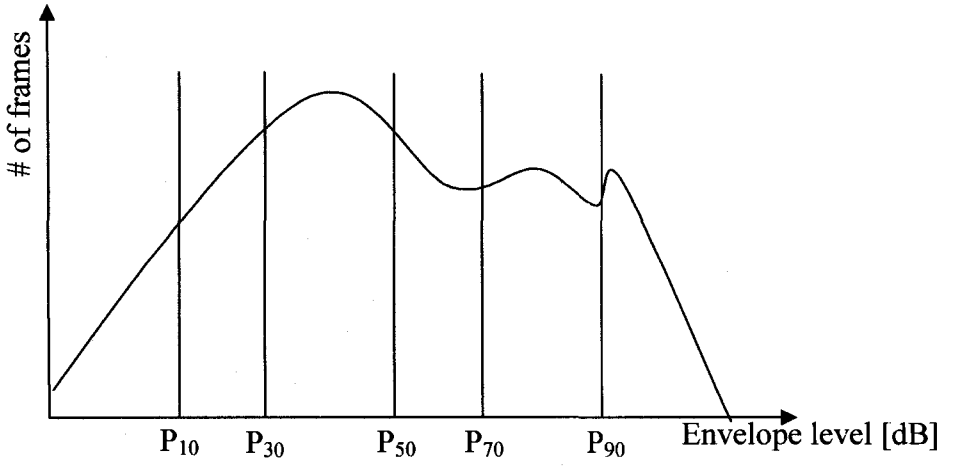


Figure 3.2: Amplitude histogram measures (courtesy of Büchler [11])

The following four features are derived from the amplitude envelope histogram's percentile measures.

3.2.1.1 Amplitude histogram width feature

The width of the amplitude envelope is a useful feature in separating speech from other types of environments. Clean speech generally has a narrow width while other types of signals, such as background noise and certain types of music, generally have broad widths. The width of the amplitude envelope histogram is measured by comparing the 90% level with the 10% level normalized with the 50% level.

$$width = d_{90-10} = \frac{P_{90} - P_{10}}{P_{50}}$$

3.2.1.2 Amplitude histogram symmetry feature

The symmetry of the amplitude envelope histogram is also a useful measure to separate speech from other types of signals. Speech generally has an asymmetric histogram while background noises and certain types of music generally have symmetric histograms. The symmetry of the amplitude envelope histogram is measured by comparing the width between the 90% and the 50% with the width between the 50% percentile and the 10% percentile.

$$symmetry = (d_{90-50}) - (d_{50-10})$$

3.2.1.3 Amplitude histogram skewness feature

The Skewness feature is also used to measure the symmetry in the amplitude modulation histogram. It is defined as the difference between the 50% percentile and the median.

$$Skewness = P_{50} - \tilde{x}$$

where the estimated median $\tilde{x}$ is calculated with the following equation:

$$\tilde{x} = \frac{(P_{90} + P_{10})}{2}$$

3.2.1.4 Amplitude histogram kurtosis feature

The kurtosis determines if the amplitude envelope distribution has a narrow or broad peak and is calculated with the following equation:

$$kurtosis = \frac{P_{70} - P_{30}}{2(P_{90} - P_{10})}$$

3.2.1.5 Shape of the lower half of the amplitude histogram feature

The shape of the lower half of the amplitude envelope histogram is also useful as a feature. Signals that have an impulse-like shape tend to have an asymmetric histogram that results in a larger lower half feature value compared to a continuous signal. This calculation is shown below:

$$lower\ half = (d_{50-30}) - (d_{30-10})$$

3.2.2 Features based on Modulation Frequency Analysis

Originally introduced by Ostendorf in [17] and further studied by Büchler, the modulation frequency is shown to be very influential in terms of distinguishing between clean speech, noise, and even speech in noise. Because speech can be broken down into sentences, words, syllables and phonemes, the modulation frequencies carry valuable information. On average, the rate we speak at is 12 phonemes (12Hz), 5 syllables (5Hz), 2.5 words (2.5Hz) and less than 1 sentence (<1Hz) per second. This gives a concentration in modulation frequency at

around 4Hz. Therefore a large modulation depth at 4Hz may indicate a speech signal. On the other hand, noise tends to have a weak modulation depth at lower modulation frequencies, while having stronger depth at higher frequencies.

Therefore, by isolating different frequency bands, different features may be derived. Ostendorf divides the amplitude modulation spectrum into three features: modulation from 0 to 4 Hz, (M1), modulation from 4 to 16 hz (M2), and modulation from 16 to 64 Hz (M3). The diagram of Figure 3.3 shows how these features are extracted. First, the envelope of the input signal is extracted and normalized by its RMS value. The envelope is then filtered through a filter bank separating the signal into its three bands. These three signals are then converted to the frequency domain via the FFT and then all the magnitudes in each frequency band are summed up to produce one feature value for each signal and each band, producing the three features M1, M2 and M3.

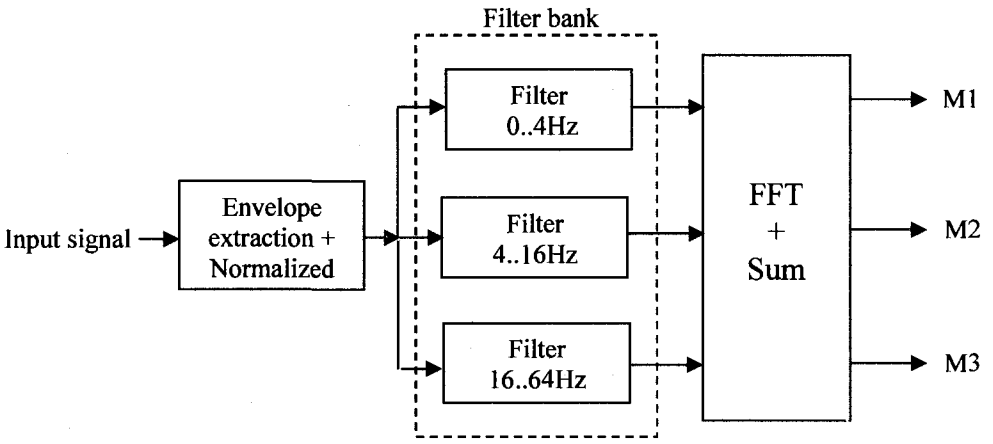


Figure 3.3: Amplitude modulation diagram

3.2.3 Features based on Spectral Form

Features are also commonly extracted from the spectrum of the signal. Büchler uses two features borrowed from Phonak in [18], which are themselves a slightly modified version of

features developed by Kates in [9]. The two features are extracted from the Center of Gravity, which measures the mean frequency in the Bark spectrum:

$$CG = \frac{\sum_{n=1}^{N} n \cdot f(n) \cdot k_n}{\sum_{n=1}^{N} f(n) \cdot k_n}$$

where f(k) is the FFT spectrum magnitude output in dB for bark index n, N is the number of barks, and k_n is the number of FFT bins in bark n.

3.2.3.1 Average Center of Gravity (CG) Feature

The average CG is a useful feature to separate high frequency signals such as rain and printer sounds, from low frequency signals such as in car noise and bass instruments. The average CG is taken over a period of time T_{Mean}, which is typically 1 second, producing the average center of gravity (CGAV) feature.

$$CGAV = \frac{1}{T_{Mean}} \sum_{T_{Mean}} CG$$

3.2.3.2 Fluctuation in the Center of Gravity feature

Phonak measures the temporal fluctuation of the center of gravity (CG) using following equation:

$$CGFS = \log \frac{E(CG)}{STD(CG)} \approx \log \frac{CGAV}{\frac{1}{3}(CG\_\max - CGAV)}$$

where E(CG) is the expectation of the center of gravity, which is also estimated by CGAV, STD(CG) is the standard deviation of the center of gravity, which is also approximated by 1/3 the difference between the maximum center of gravity (CG_max) and the average center of gravity (CGAV) over T_{Mean}. The CGFS is a useful feature in detecting the presence of speech signals.

3.2.4 Pitch features

To extract the pitch information of a signal for the purpose of simple pitch determination, a basic algorithm may be used. Korl in [19] derives a simplified algorithm from Karjalainen and Toronen [20].

First, the signal is high pass filtered to remove the DC portion of the signal below the autocorrelation (ACF) resolution. Then, the autocorrelation is calculated using the following equation [20]:

$$corr(\tau) = IDFT\left\{DFT\{x(\tau)\}^{k}\right\} \qquad (3.1)$$

where τ is the time lag variable. k is set to 1, instead of 2 for normal autocorrelation, to reduce the complexity of squaring operation, since both are found to be comparable [19]. The last step is to determine the lag peak of the autocorrelation by finding the maximum value within a certain range.

According to Büchler, the pitch itself is not very useful in classification, however some features derived from the pitch are. These features are summarized in Table 3.1.

Table 3.1: Features derived from pitch

Feature	Description
Tonality	The tonality is the ratio between tonal and non-tonal segments in a certain time window. It measures the amount of tonal components in a signal.
Pitchvar	Variance of pitch over a certain time window. This measures the temporal variation of the pitch.
Deltapitch	Absolute value of difference of two consecutive pitch values, averaged over a certain time window. Similar to pitchvar, this also measures the temporal variation of the pitch.

3.2.5 Temporal Onsets and Offsets features

A synchronous temporal onset is one where all partials of a source begin in sync, for example if a few different tones are played simultaneously. In contrast, an asynchronous temporal onset is one where partials are delayed from one another. The offset is similar, except that it represents how the partials end. The temporal onset is described as being an important feature that forms a significant part of the timbre. The timbre is defined as the perceptual qualities of auditory objects. The timbre is responsible for the dissimilarity between two sounds having identical loudness and pitch. The offset, however, is much less important and thus not too useful as a feature. In order to compute the onset as a feature, Büchler uses the process of Figure 3.4. After calculating the power spectrum of the signal it is divided into 20 bark bands via a filter bank. The spectral envelope is then measured using a first-order low pass filter with time constant of 10ms. The difference from the previous frame is then computed giving a single delta value. If this delta value is larger than some threshold then it is passed as the onset, otherwise the onset is equal to 0. This allows the onset to only be represented by significant changes that are most likely due to the onset of partials. A spectrotemporal map of offset strengths can then be produced over a certain time window, typically 5 seconds long. Büchler also investigated using a single band instead of 20 bark bands and determined that the single band case does not hold enough information to distinguish speech, speech in noise, noise, and music. However, useful features are derived from the spectrotemporal onset map produced by the 20 bark bands.

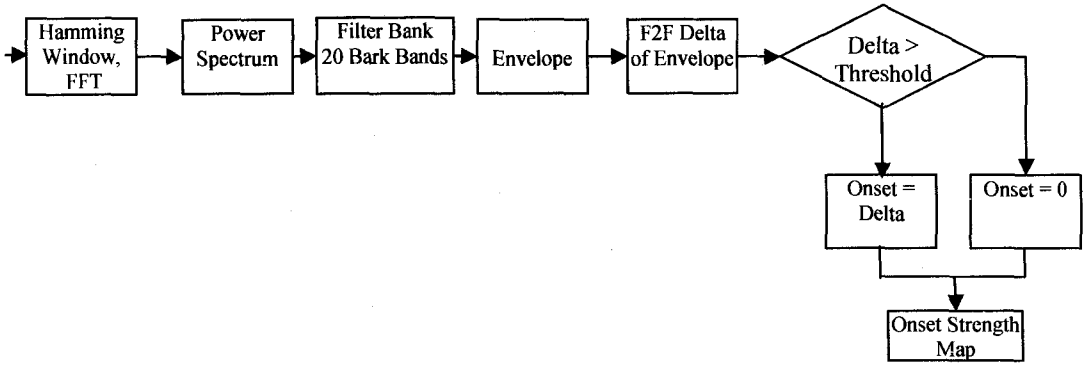


Figure 3.4: Block diagram for calculating onset (Courtesy of Büchler) [11]

Two features are derived from the onset map by first summing up the onset strength across all barks. The mean and variance are then calculated resulting in the two features *Onsetm* and *Onsetv*. Two more features are derived by determining the number of common onsets across all barks. This is done by counting the number of onset strengths that are larger than some threshold value in each band. The mean is then taken over some time window, resulting in the common onsets (*Onsetc*) feature. The last feature is similar to *Onsetc*, except that instead of calculating the mean over a period of time, it determines the number of common onsets larger than some threshold over some period of time, resulting in the feature *Onseth*. The four features are summarized in Table 3.2 below.

Table 3.2: Summary of onset feature and their description

Feature	Description
Onsetm	Mean onset strength: computed by summing up the onset over all bark bands and calculating the mean onset.
Onsetv	Variance of onset, computed by summing up the onset over all barks and calculating the variance.
Onsetc	Measures the mean of the common onsets, which determines the number onsets larger than some threshold over all bands, over a certain time window.
Onseth	The number of common onsets that are larger than some threshold over a certain time window.

3.2.6 Rhythm or Beat features

According to Büchler in [11], the beat can be estimated through the spectrotemporal onset pattern of a signal, described in the previous section. A feature that extracts the beat of a signal is a very useful feature when it comes to distinguishing between music and speech or music and noise, since most music signals will contain some kind of beat, while speech and noise will most of the time have an absence of a beat. In order to determine the beat frequency the autocorrelation function is used in a similar way as it was used to find the pitch in section 3.2.4. Figure 3.5 shows the block diagram for the beat detector used by Büchler and motivated by Scheirer in [21]. First, the onset map is passed to a high pass filter in order to remove the DC component below the resolution of the autocorrelation function (ACF). Identical to the ACF used in the pitch detector of section 3.2.4, the ACF is computed using Equation 3.1. The 20 ACF signals are then summed into one signal, called the Summary ACF (SACF). In order to obtain more reliable and accurate beat information a number of consecutive SACF's are summed before detecting the peak. This ensures that the beat detected is one that is constant over a longer period of time.

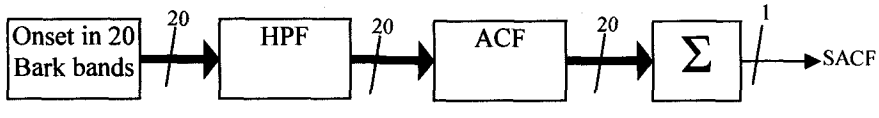


Figure 3.5: Calculating the SACF of the onset

One problem arises when it comes to using this method for computing the beat as a feature for hearing aid purpose. Since it requires a number of beats per frame to produce an accurate SACF and a certain number of consecutive SACF for accurate measurements, it may take a long time to detect a beat. For example if we require 10 beats per frame and 3 consecutive SACF's, the algorithm will only detect a beat every 30 seconds. This is too long for a hearing aid classifier, since typically classification decisions are done every second.

Scheirer in [21] proposes a beat detector that detects the beat frequency in real time. The beat detector selects the resonator that produces phase-lock behavior, out of a bank of tuned resonators.

3.3 Classifier types

There are many different types of classification algorithms that can be used to classify a new environment as one of a set of known classes. The Minimum distance and Bayesian classifiers are most used based on their simplicity and their computational efficiency. These two types of classifiers are described in more detail below.

3.3.1 Bayesian Classifier

The Bayesian classifier assumes that statistical information for each class is known or can be learned. This allows for a decision to be made based on a probabilistic model. To classify a given input vector $\vec{x}$, the posterior probability $P(w_i|\vec{x})$ is first calculated for each class i. The class having the maximum posterior probability is selected as the winning class. In order to calculate the posterior probability, Bayes theorem is used:

$$P(w_i|\vec{x}) = \frac{p(\vec{x}|w_i)P(w_i)}{p(\vec{x})}$$

where $p(\vec{x}|w_i)$ is the conditional probability of $\vec{x}$ given the statistical parameters of class i (defined as w_i), $P(w_i)$ is the prior probability of class i, and $p(\vec{x})$ is the prior probability of $\vec{x}$. Since $p(\vec{x})$ is constant for all classes it may be removed for simplification. In addition, if we assume that all classes have equal probability, we may also ignore $P(w_i)$. This reduces the equation to the following:

$$P(w_i|\vec{x}) \propto L(\vec{x}|w_i) = p(\vec{x}|w_i)$$

where $L(\vec{x}|w_i)$ is the likelihood of $\vec{x}$ given w_i. Therefore the Bayesian classifier is reduced to a maximum likelihood measure.

3.3.2 Minimum Distance Classifier

A minimum distance classifier is illustrated in Figure 3.6. Each class is represented by reference point, shown as squares, which represents the center of each class. To classify new data, shown as a circle in the figure, its distance to each class center is first determined. The data is classified as belonging to the class with the smallest distance. There are different types of distance measures that may be used. The most common measures are: (1) the Euclidean distance, where only the center for each class is needed (further described in section 3.8.1); and (2) the Mahalanobis distance, where the center and covariance matrix is needed (also further described in section 3.8.2).

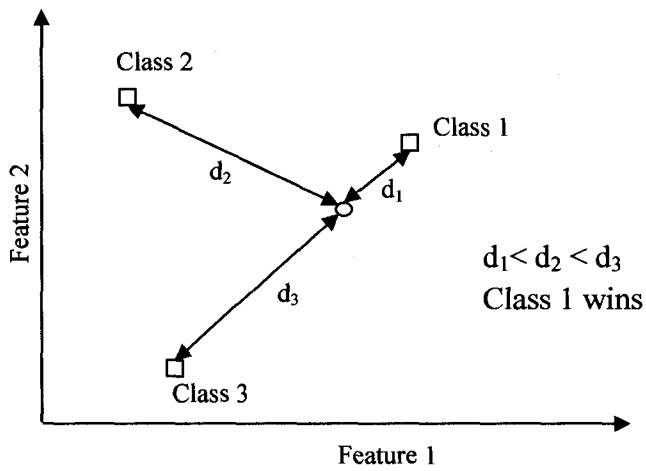


Figure 3.6: Minimum distance classifier

3.4 Supervised vs. Unsupervised learning for Pattern Classifiers

Pattern classifiers can be divided into two groups based on the type of learning used during training, which could be supervised or unsupervised. In supervised learning, the training data has predefined classes that the classifier uses to guide the learning process. Most environmental classification systems use this type of classifier since sound can often be classified according to its nature (speech, noise, music, etc.). An advantage of such classifiers is that expert knowledge can be readily exploited to specify an optimum set of hearing aid parameters for each class. For example, if a speech in noise class is detected the hearing aid may automatically set a noise reduction feature on. A drawback of supervised learning is that the class types must be specified a priori and may or may not fit the actual environments of a hearing aid user. Also there is little scope for adapting the system or class set after training. Typically, all users will have the same classification system irrespective of their needs and hearing profiles.

In unsupervised learning, a classifier tries to recognize patterns in the training data, without a priori knowledge of the class they belong to. This method may be also referred to as clustering. One advantage of unsupervised learning is that classes are defined on the basis of the actual environments explored by each user. Continuously adaptive systems are also possible. However, the exact nature of the clusters may not conform to standard environment classes, so that the action to be taken by the hearing aid or the parameters to be selected for each class must be learned as well. Together, the unsupervised learning of the classifier and learning of the user action for a given cluster provide the complete solution for a user independent environment classification.

3.5 Supervised learning

Training the minimum distance classifier and Bayesian classifier using supervised learning is fairly straight forward. The statistical parameters required for the minimum Euclidean distance classifier is simply the mean value for each class. The minimum Mahalanobis

distance classifier and the Bayesian classifier, in addition to the mean, requires the covariance matrix for each class. Using a training set of data for each class the mean and covariance matrix can easily be calculated using the following basic statistical equations:

Mean μ_i

Definition:

$$\mu_i = E\left[X^{(i)}\right]$$

where E is the mathematical expectation and $X^{(i)}$ is the training data set belonging to class i.

Estimation:

$$\mu_i = \frac{1}{N_i}\sum_{j=1}^{N_i} X_j^{(i)}$$

where $X_j^{(i)}$ is the jth training data vector belonging to class i, N_i is the total number of training data belonging to class i.

Covariance Matrix Σ_i

Definition:

$$\Sigma_i = E\left[\left(X^{(i)} - E\left[X^{(i)}\right]\right)\left(X^{(i)} - E\left[X^{(i)}\right]\right)^T\right]$$

Estimation:

$$\Sigma_i = \frac{1}{N_i - 1}\sum_{j=1}^{N_i}\left(X_j^{(i)} - \mu_i\right)\left(X_j^{(i)} - \mu_i\right)^T$$

3.6 Unsupervised learning

To estimate the parameters for classes that are not pre-defined, a much more complex task is at hand. Nevertheless, there are many unsupervised learning algorithms that aim to determine the most optimum classes by detecting cluster patterns in the training data. Some algorithms include: (1) the K-mean algorithm, where the mean for K clusters is estimated with the assumption that all clusters are spherical in shape [22]; and (2) Expectation Maximization (EM) clustering, where the mean and covariance matrix is estimated for each cluster [23]; Numerous papers have been published that are based on these basic algorithms. For example, Zhang in [24] introduces an algorithm called Self-Splitting Competitive Learning (SSCL) that eliminates the requirement of a predetermined number of clusters in the K-mean algorithm. In this section, these three algorithms are described in more detail.

3.6.1 K-Mean Clustering algorithm

The K-mean algorithm is an unsupervised clustering algorithm. It starts with a pre-determined number of clusters (K clusters) and finds the center of the clusters based on training data. The algorithm aims to cluster the data in the most optimum way by minimizing the following cost function:

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, m_i)$$

where x is the current training data value, m_i is the center of cluster C_i, k is the number of clusters and $d(x,m_i)$ is the Euclidean distance between x and m_i.

The algorithm first randomly places the centers m_i for each cluster. The training data is then classified to its closest center. The centers are then repositioned in the direction of decreasing cost function by recalculating the centers of the classified data. This is repeated until the centers no longer move, thus reaching its minimum cost.

Once the learning is complete, the information that the algorithm provides is simply the centers for each cluster. Therefore, the classifying boundary for each cluster is assumed to be spherical in shape. If the clusters are non-spherical in shape, the EM clustering algorithm may be more appropriate.

3.6.2 EM clustering algorithm

A more general clustering algorithm than the K-means is the Expectation Maximization (EM) clustering algorithm. The EM clustering process is identical to the process of estimating the distribution of the training data for all clusters combined using a Gaussian Mixture Model (GMM).

3.6.2.1 Gaussian Mixture Model (GMM)

A Gaussian Mixture Model (GMM) is a probability density function (PDF) defined by a weighted sum of Gaussian functions referred to as Modes. The equation for the GMM is given below:

$$p(z) = \sum_{i=1}^{L} \alpha_i \mathrm{N}(z, \mu_i, \Sigma_i)$$

where α_i are the mode weights, L is the number of modes or Gaussian functions that are being used to represent the distribution, and $\mathrm{N}(z, \mu_i, \Sigma_i)$ is the Gaussian function having a mean μ_i and a covariance matrix Σ_i. The Gaussian function is defined as:

$$\mathrm{N}(z, \mu_i, \Sigma_i) = \frac{1}{(2\pi)^{D/2} |\Sigma_i|^{1/2}} \exp\left\{ -\frac{1}{2} (z - \mu_i)' \Sigma_i^{-1} (z - \mu_i) \right\}$$

where D is the number of dimensions in the distribution.

Thus by setting the number of modes L to the number of clusters K (L=K), the parameter estimates for the L modes will also be the optimum parameters for the K clusters. For

example if we would like to cluster a feature space into 3 clusters, then a 3-mode GMM is used to estimate the parameters for the three cluster.

The most common method used to find the modes that make up the GMM is the EM algorithm, hence the name EM clustering algorithm.

3.6.2.2 EM Algorithm

The algorithm is a two-step recursive algorithm that aims to find the maximum likelihood estimates for the mode parameters of a GMM, specifically the mean and covariance matrix. The first step is the Expectation step, or E-step, and the second is the Maximization step or M-Step. These steps are described in more details below. First, a few notations need to be introduced.

Notation:

- The input data is denoted as $\mathbf{X}=\{x_1,...,x_n\}$ with a distribution $f(\mathbf{X};\Psi)$, where Ψ is the unknown parameters for the clusters. The unknown parameters for a GMM are: the mode weight, the mean, and the covariance matrix for each Gaussian function. $\Psi=\{\alpha, \mu, \sum\}$.
- $Z=\{Z_1,...Z_n\}$ is the cluster assignment matrix where

$$Zj=(z_{j1},z_{j2},...,z_{jk}) \text{ and } z_{jk}=\begin{cases}1, & if\ x_j \subset cluster\ k\\ 0, & if\ x_j \not\subset cluster\ k\end{cases}$$

- $Y=<X,Z>$ is the complete data with density function $fc(Y;\Psi)$. For example, if the input data X had 4 vectors, $X=\{x_1,x_2,x_3,x_4\}$, and if x_1 and x_2 were to be assigned to cluster 1 while x_3 and x_4 were to be assigned to cluster 2, then the assignment matrix Z would have the following format: $Z=\{(1,0), (1,0), (0,1), (0,1)\}$.

Algorithm details:

The maximum likelihood principle states that the parameters that maximize the data log likelihood are consistent estimates of the true parameters. The data log likelihood is defined as:

$$\log L(\Psi) = \log f(X;\Psi)$$

and the complete data log likelihood is defined as:

$$\log L_C(\Psi) = \log f_C(Y;\Psi)$$

Since we do not have the information for Z, the algorithm starts by arbitrarily assigning a cluster to each data point. The cluster assignment is updated iteratively with the parameters Ψ until the cluster assignment stabilizes.

In order to guarantee convergence in the log likelihood update, its expectation is calculated. It is proven that by maximizing the function

$$G(\Psi;\Psi^t) = E_{\Psi^t}\{\log L_C(\Psi)|X\}$$

the log likelihood is non-decreasing for each iteration [23].

Algorithm steps:

The following two steps are repeated until $\left|\log L(\Psi)^{t+1} - \log L(\Psi)^t\right| \le \varepsilon$

where ε is some constant.

1) E-Step:

calculates $G(\Psi;\Psi^t)$

2) M-Step:

Finds Ψ^{t+1} that maximizes $G(\Psi;\Psi^t)$

3.6.3 Self-Splitting Competitive Learning (SSCL)

The SSCL in [24] is a partitioning type clustering algorithm. It is a slightly more complex algorithm than the K-means, however it has an adaptive characteristic that may be utilized in building an adaptive classification system. The goal of the clustering algorithm is to find the K most optimum clusters in a cluster space. However, it is different from the generic K-means algorithm in terms that it does not need to specify a predetermined number of clusters. The algorithm's learning process begins with tracking the center of one natural cluster, using a position vector called a Prototype. It then splits the Prototype into two in order to track the center of a second natural cluster. This process is repeated until the algorithm detects that all natural cluster have been represented with a Prototype vector.

Finding a cluster's center

In order for a Prototype to find the center of a single natural cluster, a vector called the Asymptotic Property Vector (APV) is used. As each new input vector $\vec{X}$ is introduced, the Prototype and APV vectors are updated in a way that allows them to converge to the center of the natural cluster that is closest to the circle having a center position of $\vec{P}_i$ and the radius given by the Euclidean distance between $\vec{P}_i$ and $\vec{A}_i$, denoted by $\left|\vec{P}_i\vec{A}_i\right|$, as shown in Figure 3.7 by the blue dashed circle.

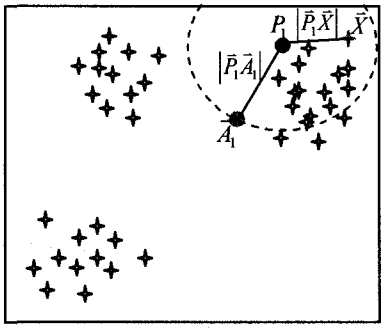


Figure 3.7: The PA circle (blue dashed circle) gives more weight to the data vectors (black stars) that lie within its boundary. Current data vector X (red star) is shown to be within the circle.

This is done by allowing the data that lies within this circle to influence the movement of the vectors $\vec{P}_i$ and $\vec{A}_i$ more than those that lie outside. In order to achieve this influence for the prototype update, the variable $\vec{\alpha}_i$ is used in its update equation (Table 3.3) to give more weight to vectors that are closer to the prototype vector $\vec{P}_i$. If $\left|\vec{P}_i\vec{X}\right| >> \left|\vec{P}_i\vec{A}_i\right|$ then $\alpha_i \rightarrow 0$; in other words, if $\vec{X}$ is far outside of the $\left|\vec{P}_i\vec{A}_i\right|$ circle then it will not contribute to the update of $\vec{P}_i$ as much as if $\vec{X}$ were inside the circle. The update equations for the APV vectors are influenced in a slightly different way. By using the general function Θ in the update equation, it only allows the APV vector $\vec{A}_i$ to be updated when the current data vector $\vec{X}$ is within the $\left|\vec{P}_i\vec{A}_i\right|$ circle. Therefore, as the Prototype vector $\vec{P}_i$ and APV vector $\vec{A}_i$ are updated they both converge to the center of the natural cluster most inside the $\left|\vec{P}_i\vec{A}_i\right|$ circle.

The update equations for the Prototypes and APVs are summarized below in Table 3.3.

Table 3.3: Equations for APV and Prototype update each time a new input vector X is selected.

Update equations for APV	**Update equations for Prototype**
$\vec{A}_i^* = \vec{A}_i + \frac{1}{n_{\vec{A}_i}} \cdot \delta_i \cdot \left(\vec{X} - \vec{A}_i\right) \cdot \Theta\left(\vec{P}_i, \vec{A}_i, \vec{X}\right)$ where Θ is a general function given by: $\Theta(\vec{\mu},\vec{\nu},\vec{\omega}) = \begin{cases} 1 & if \left\|\vec{\mu}\vec{\nu}\right\| \geq \left\|\vec{\mu}\vec{\omega}\right\| \\ 0 & otherwise \end{cases}$ and δ_i is a learning constant given by: $\delta_i = \left(\frac{\left\|\vec{P}_i\vec{A}_i\right\|}{\left\|\vec{P}_i\vec{X}\right\| + \left\|\vec{P}_i\vec{A}_i\right\|}\right)^2, \; 0 < \delta_i \leq 1$ and $n_{\vec{A}_i}$ is a winning counter given by: $n_{\vec{A}_i} = n_{\vec{A}_i} + \delta_i \cdot \Theta\left(\vec{P}_i, \vec{A}_i, \vec{X}\right)$	$\vec{P}_i^* = \vec{P}_i + \alpha_i\left(\vec{X} - \vec{P}_i\right)$ where $\alpha_i = \left(\frac{\left\|\vec{P}_i\vec{A}_i\right\|}{\left\|\vec{P}_i\vec{X}\right\| + \left\|\vec{P}_i\vec{A}_i\right\|}\right)^2 = \left(1 + \frac{\left\|\vec{P}_i\vec{X}\right\|}{\left\|\vec{P}_i\vec{A}_i\right\|}\right)^{-2}$

The Prototype and Asymptotic Property vector is updated according to the equations of Table 3.3 for all of the input data in the feature space, selected one at a time at random. Once the algorithm has cycled through all of the training data; the algorithm checks to see if the prototype and APV vectors have converged to the center of a natural cluster by verifying the following condition.

Convergence condition:

If $\left|\overline{P_i A_i}\right| < \varepsilon_{12}$ then stop update, else recycle all training data.

If this condition is met, then the algorithm must then check to see if all natural clusters have been represented by a prototype by verifying a Split Validity Criterion. If this condition is not met, then the prototype is split and the algorithm restarts.

Split Validity Criterion

The Split Validity Criterion determines if more than one natural cluster is being represented by one prototype. For example, as Figure 3.8 illustrates, if there exist 3 natural clusters (S_1,S_2 and S_3) but only two prototype vectors, $\vec{P}_1$ and $\vec{P}_2$, one of the prototypes ($\vec{P}_2$ in this case) will represent data from two natural clusters. In order to detect if a new prototype needs to be created, the Center Property Vector (CPV) is introduce. The CPV tracks the center of the data for each class as shown in Figure 3.8. Similar to the Prototype and APV, the CPV vector $\vec{C}_i$ is also updated with each new input data $\vec{X}$ using the following equation:

$$\overline{C}_i^* = \overline{C}_i + \frac{1}{n_{\overline{C}_i}}\left(\overline{X} - \overline{C}_i\right)$$

where $n_{\overline{C}_i}$ is a winning counter having the following equation:

$$n_{\overline{C}_i} = n_{\overline{C}_i} + \delta_i \cdot \Theta\left(\vec{P}_i, \overline{C}_i, \vec{X}\right)$$

If the CPV vector $\vec{C}_i$ converges to the same position as the prototype vector $\vec{P}_i$, the prototype is only representing one natural cluster, which is the case for $\vec{P}_1$ in our example of Figure 3.8. On the other hand, if the CPV vector $\vec{C}_i$ does not converge to the same position as the Prototype vector $\vec{P}_i$, then it must be representing more than one natural cluster and therefore should split to accommodate an additional natural cluster, which is the case for $\vec{P}_2$ in our example. This criterion can be written in mathematical terms as follows:

$$\text{if } \left|\overline{P_i A_i}\right| < \varepsilon_1 \cap \left|\overline{P_i C_i}\right| > \varepsilon_2 \text{ , then split.}$$

The first term is the convergence condition and determines if the Prototype $\vec{P}_i$ has converged to the APV $\vec{A}_i$ within a threshold distance of ε_1. The second term determines if the distance between CPV $\vec{C}_i$ and the PV $\vec{P}_i$ is greater than the threshold ε_2.

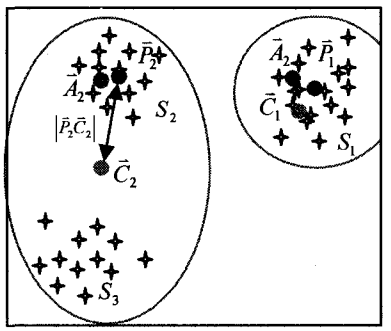


Figure 3.8: Vector converging to one cluster

Once this criterion is met, a new set of vectors are created. The Distance Property Vector (DPV) $\vec{R}$ is used to determine where the new prototype will lie after being split. It has the following update equation:

$$\overline{R}_i^* = \overline{R}_i + \frac{1}{n_{\overline{R}_i}} \cdot \rho_i \cdot \left(X - \overline{R}_i\right) \cdot \theta\left(\overline{P}_i, \overline{X}, \overline{R}_i\right)$$

where $\rho_i = \left(\frac{|\overline{P_i X}|}{|\overline{P_i X}| + |\overline{P_i R_i}|} \right)^2$ or can be chosen as a constant and takes a value between 0 and 1.

As the number of iterations increases, the vector $\vec{R}_i$ will move further away from its original prototype. This will minimize the influence that the mother cluster will have on the new prototype.

This process is repeated until the split validity criterion is not met, thus having found all K clusters.

3.7 Post-Processing

The post-processing step in the classification layout of Figure 3.1 ensures that the final class output has a smooth transition from class to class. For example, if the classifier is consistently outputting class 1 and a misclassification leads to a change to a class 2 for a short period of time, the user may find the quick change in hearing aid setting to be annoying. Therefore, the post-processing stage aims to remove this spurious behavior for better user comfort. One solution for the post-processing stage may be to only allow a change in class to occur if the classifier consistently outputs the new class for a given period of time (10 seconds for example).

3.8 Distance Measures

This section is a review of different types of distance measure that can be seen in classification. First, a short review on the Euclidean distance, used by many classifiers including the minimum distance classifier, is presented; then, the more general Mahalanobis distance, that takes the covariance matrix for a distribution in consideration, is reviewed. Finally, the Mode distance, used to measure the distance between two distributions, is introduced.

3.8.1 Euclidean distance

The Euclidean distance between feature vector X and class k is defined as follows:

$$d_k = \sqrt{(X-\mu_k)^t (X-\mu_k)}$$

where μ_k is the mean feature vector of class k.

3.8.2 Mahalanobis distance

The Mahalanobis distance takes into account the covariance of the data set making it an ideal distance measure for non-spherical clusters. The distance is measured as follows:

$$d_k^2 = \left\{(X-\mu_k)^t \Sigma_k^{-1} (X-\mu_k)\right\}$$

where Σ_k is the covariance matrix of class k, defined by Equation 3.2, and μ_k is the mean of class k.

$$\Sigma_{ij} = E\left\{(x_i-\mu_i)(x_j-\mu_j)\right\} \qquad (3.2)$$

3.8.3 Mode Distance

In order to determine the distance between two Gaussian probability density functions (PDFs), the mode distance measure introduced by Baggenstoss in [25] may be used. The mode distance is a measure used in Gaussian mixture models to determine the distance between two modes in estimating the distribution. Since each mode in the model represents a Gaussian function, it is easy to see how this can be applied to separate PDF's that are assumed to be Gaussian. The mode distance equation is:

$$d = \frac{\prod_{x_i \in X_1} p_2(x_i) \prod_{x_i \in X_2} p_1(x_i)}{\prod_{x_i \in X_1} p_1(x_i) \prod_{x_i \in X_2} p_2(x_i)}$$

where $p_1(x_i)$ and $p_2(x_i)$ are the PDF's of two Gaussian functions, and X_1 and X_2 are a selection of points lying around their respective mean.

If $p_1(x_i) = p_2(x_i)$ then $d = 1$ and if $p_1(x_i) \neq p_2(x_i)$ then $d < 1$. Therefore the smaller the value d is, the farther the two PDF's are from each other.

3.9 Büchler's Feature Comparison

Büchler [11] determines the best input features to utilize in different classifier types to classify the four following classes: speech, speech in noise, noise, and music. The features tested included the five amplitude modulation features of sections 3.2.1 and 3.2.2: M1, M2, M3, Width, and Skewness; the two spectral features of section 3.2.3: CGAV and CGFS; the three pitch features of section 3.2.4: Tonality, Pitchvar, and Deltapitch; and the beat feature of section 3.2.6: beat. The classifiers used consist of the Minimum distance classifier using Euclidean and Mahalanobis distances and a Bayesian Classifier. Büchler uses the following iterative strategy to find the best features for each classifier:

1) Tonality and amplitude modulation features are tested for the specified classifier.
2) The best amplitude modulation features found in step one are combined with the Pitchvar and Deltapitch and retested.
3) The best features of step 2 are combined with the spectral features and retested
4) The best set of features in step 3 are then combined with the Onset features
5) The amplitude modulation and spectral features are then removed one by one from the best set of step 4 to identify the best features to retain.
6) Lastly, the Beat feature is added and the overall remaining set is tested with and without the Beat feature

A summary of the results are shown below in Table 3.4, where the accuracy for each classifier type is measured using the overall hit rate. The overall hit rate is a measure of how many inputs samples were classified correctly averaged over all the classes (see section 4.5).

Table 3.4: Best features reported by Büchler for each classifier

Classifier	Best Features	Overall Test Hit Rate (%)
Euclidean Minimum-distance	Tonality, Pitchvar, M1, M2, CGFS, Onsetv, Beat	83.3
Mahalanobis Minimum Distance	Tonality, Pitchvar, M1, CGFS, Onsetc	82.9
Bayesian Classifier	Tonality, Pitchvar, M1, M2, M3, CGFS, Onsetm, Onsetc	84.3

As Table 3.4 shows, the set of best features selected is dependant on the type of classifier.

3.10 Summary

In this chapter the different components of a basic classification system were reviewed, starting with a description of the feature extraction step that captures characteristic information from the environment. This included a description of features based on amplitude statistics (width, symmetry, skewness, kurtosis, lowerhalf), modulation frequency analysis (M1, M2, M3), spectral form (CGAV, CGFS), pitch (Tonality, Pitchvar, Deltapitch), and temporal onsets (Onsetm, Onsetv, Onsetc, Onseth). This was followed by a description of the two common classifier types, minimum distance and Bayesian classifier, which classify the environment into one of its predefined classes. The primary difference between the two types of classifiers is the complexity. The minimum distance classifier provides a solution that is low in complexity by assuming the feature clusters for each class are spherical in shape (i.e., all features have the same class variance). In addition, the minimum distance classifier also assumes that the clusters for all classes are the same size (i.e., same variance for all classes). The Bayesian Classifier removes these assumptions and therefore allows for clusters to be non-spherical in shape and different in size. However, the result is an increase in complexity since the distribution for each class is required. The method used to train the two classifiers using supervised learning was then presented along with three types of unsupervised learning algorithms that can be used if the classes are not defined. Specifically described were the K-means, the EM clustering and the self splitting competitive learning (SSCL) algorithms. The K-means algorithm is used in cases where only the mean for a set number of clusters is to be learned (e.g., the minimum distance classifier).

The EM clustering algorithm, in addition to the mean, also provides the covariance matrix which can be used in the types of classifiers that require the distribution of each cluster (e.g., the Bayesian classifier). The SSCL provides similar results as the K-means algorithm, that is, only the mean for each cluster is learned. However, it also aims to find the optimum number of clusters present and therefore does not require a predetermined number of clusters. Subsequently, a short description of the Euclidean, Mahalanobis, and Mode distances, three Distance Measures that are commonly seen in classification, were presented. The chapter is concluded with a comparison in features for three classifiers was provided and therefore determined that the Bayesian classifier performed best with the eight features: Tonality, Pitchvar, M1, M2, M3, CGFS, Onsetm, Onsetc.

With these classification tools in hand, a framework that aims to modify the basic layout of a static classification system to include an adaptive layer can be developed.

Chapter 4

Proposed Adaptive Classification Framework

In order to guide the design of the adaptive classification system, a general framework is needed. In this chapter, an adaptive algorithm layout is proposed that adds an adaptive layer to a static classification system. In addition, a list of requirements that need to be met in designing the adaptive layout is presented. Lastly, the method used to test the adaptive system and measure its accuracy and performance is summarized.

4.1 Algorithm Layout

Current automatic environmental classification systems define their classes based on supervised learning where the classes are known a priori and predefined, therefore may not be altered adaptively as the user enters new and different environments. Unsupervised learning algorithms are used in cases where the classes are not predefined and are determined based purely on feature patterns.

In this thesis a new adaptive layer is introduced to the original classification model to create a sound classification system that can adaptively split and merge classes based on the distinction or similarity of the feature patterns in the environment that the hearing aid user encounters. For example, if the feature patterns consistently show that there are two distinct sub-clusters within a class, the algorithms should detect this pattern as representing separate classes and a split should take place. Likewise, if the distance between two classes become

very small and the number of classes exceed a preset maximum then the two classes should merge into one. Therefore, the algorithm must be capable of merging or deleting classes. This process should be ongoing during the use of the hearing aid.

Figure 4.1 shows the new proposed layout. The layout is very similar to the original sound classification system of Figure 3.1, except for the addition of the adaptive classifier along with a buffer stage. Like the classifier of Figure 3.1, the sound that is picked up from the microphone is first converted into a stream of feature vectors via feature extraction. The vector is then passed on to the adaptive classifier to be classified into a class after smoothing by a post-processor, which in turn will determine the hearing aid settings. However, the system also passes the feature vectors to be stored into a buffer. When the buffer is filled, it computes a single representative vector that is then used to update the adaptive classifier. These additional components are described in more detail below.

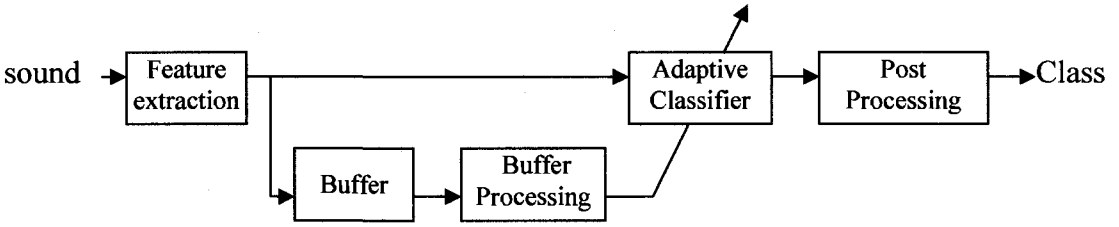


Figure 4.1: Adaptive Classification System

4.1.1 Buffer

The buffer stage consists of an array that store past extracted feature vectors. Typically, the buffer stage may be 15-60 seconds long depending on the rate at which the adaptive classifier needs to be updated. The buffer processing stage processes all the stored features to yield a single feature vector to represent all of the buffered data, allowing a more accurate assessment of the acoustical characteristics of the current environment for the purpose of adapting the classifier. This will allow the adaptation of the classifier to run at a much slower rate than classification.

Figure 4.2 shows how a simple averaging function for the buffer processing stage can remove a spurious jump in the feature value due to a transient fluctuation in class. Since the majority of features are extracted from the class 1 environment, the average will filter out the spurious class fluctuation. As a consequence, features from new unknown classes must be experienced for a significant amount of time relative to the buffer size to be passed to the adaptive algorithm. This reduces the sensitivity for adaptation for classes that are only visited for very short periods of time. The length of time that determines how long a user must be in an unknown environment to be passed to the adaptive classifier is depended on the length of the buffer as well as the function used for the buffer processing stage.

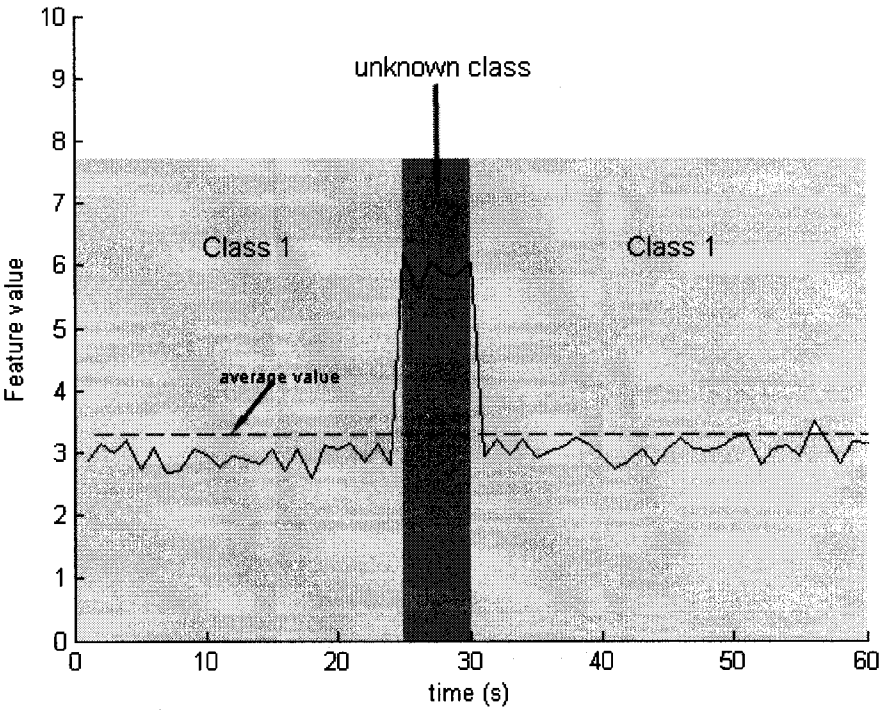


Figure 4.2: Averaging buffered feature

4.1.2 Adaptive Classifier

The adaptive classification system is first initialized to a system that organizes the environments into a set number of classes. This is the initial classification system that the user will begin with when the user first turns on the hearing aid following the fitting process.

Starting with a working classification system will also give the user or clinician the option to turn off the adaptive classifier and thus restrict the classification to the initial classes. Since we are training the system to recognize specific initial classes, a supervised learning algorithm is appropriate.

Once the adaptive classification system is turned on, the algorithm begins monitoring changes in the feature extraction patterns as the hearing aid user goes about his/her daily routine. If the user consistently enters an environment that is significantly different than current classes in terms of feature vectors, the hearing aid will split a class to accommodate the new environment. Therefore, a Split criterion is needed to assess if a class should divide and to select the most appropriate class for splitting. In addition, if the maximum number of classes has been reached, the algorithm must be able to merge classes to make room for new ones. This requires a Merge criterion. The Split/Merge process of the adaptive classifier is a focus of this thesis.

Once the split has taken place, the new class initially adopts the same hearing aid settings as the parent. Once a merge has taken place, some form of averaging of hearing aid parameters across the two merged classes must take place. In addition, the hearing aid settings for a specific class are also learned continuously for each individual class. Therefore, even though the hearing aid settings for the new class will initially be identical as its parent class, as the user tunes the hearing aid settings while in the new class, the settings are able to converge to different hearing aid settings for more optimum hearing. This user-based learning, however, is beyond the scope of this thesis. For information regarding the topic of self-learning hearing parameters see [26].

4.2 Project Design Requirements

The following are the requirements that must be fulfilled when designing an adaptive classification system.

4.2.1 Feature Selection

The first step in designing an adaptive system is to select the features that will be used to extract the characteristic information from the training and testing signals. Since the focus of this thesis is on the design and testing of a system that can adaptively split and merge classes based on the features, only three features will be used to keep the complexity at a minimum. The three features that were chosen are: Modulation from 0 to 4 Hz (M1 in section 3.2.2), modulation from 4 to 16 Hz (M2 in section 3.2.2) and a stationarity feature that separates stationary signals from non-stationary signals (stationary signals maps to a low feature value, while non-stationary signals maps to a higher feature value). These three features were selected based on being readily available in black box format for Matlab. In addition, the M1 and M2 were amongst the most influential features in Büchler's comparison described in section 3.9.

4.2.2 Initial Classification System

A classifier based on supervised learning needs to be selected to train the initial classification system to classify a set of predetermined classes. Ideally, this initial system would be comparable to the fixed static classifiers implemented by Büchler in [11] to classify the 4 environments Speech, Speech in noise, Noise, and Music. Again, since the goal of the thesis focuses on the adaptive behavior of the system, the initial classification system, for the purpose of this thesis, will contain a reduced number of classes as a starting point for adaptation.

4.2.3 Monitoring environment changes

To monitor the changes in features and need for adaptation, a monitoring system needs to be designed. As each new feature is being fed to the adaptive algorithm by the buffer, it will produce an incremental change in the monitoring system and two criteria need to be checked. The first is the Split Criterion, to determine if there is enough change in the

environment to warrant a new class. The second is the Merge Criterion, to check if two classes are close enough to be considered a single class to make room for others.

4.2.4 Adaptation of the classification system

Once the Split criterion is met, the algorithm needs to learn the new class. The class parameters that need to be learned depend on the classifier chosen. For example, the minimum distance classifier measures the distance from the classifying feature vector to each class mean and chooses the class with the minimum distance, therefore only the mean of the new class needs to be learned. Since no information about this new class is provided, an unsupervised learning algorithm is needed. Once the new class is learned, the algorithm may return to its original classifier and continue monitoring the features. On the other hand, if the Merge criterion is met, an algorithm is needed to choose two classes to merge. The algorithm will also have to learn the new parameters for the merged class.

4.3 Test Criterion for Splitting & Merging

In order to test the adaptive algorithm, the splitting and merging processes are tested individually and both are tested on an ideal and a real set of features. The ideal set is constructed using Gaussian distribution of feature values and clusters are clearly defined. The real data set is taken from a collection of sound files from real recordings via a feature extraction process.

The hearing aid algorithm is first trained using only two classes. Thus, we are assuming (for simplicity) that the hearing aid user takes home a system initialized with only two different environmental classes. The adaptive algorithm is then set into adaptive mode, which will allow the hearing aid classes to adapt to the user's actual environment. In this mode, the algorithm is introduced to a set of test data, which includes a new class in addition to the two already programmed classes. The expectation is for the algorithm to detect that there is a need to add a new class, decide on how to split and determine the parameters for the new

class. The actual settings (volume, directional microphone, noise reduction, etc.) for the hearing aid in this class would then be learned from the user as in [26].

4.3.1 Ideal data set

The data used for testing the ideal case is derived from four 3-dimentional Gaussian distributed clusters of feature values. The mean and variance for this data is shown in Table 4.1 and is plotted in Figures 4.3 and 4.4. The systems will first be trained with two classes as the initial classification system. To test the adaptive system, two new classes will be introduced sequentially by first introducing 100 points for each of the first 2 classes followed by 100 points of all four classes.

Table 4.1: Ideal data set

Class	Mean	Variance	# of Training points	# of Testing points
1	[1,1,1]	[1,2,0.5]	100	100
2	[2,6,6]	[2,1,0.5]	100	100
3	[8,7,3]	[0.75,0.5,0.75]	0	100
4	[7,1,4]	[0.75,0.5,0.75]	0	100

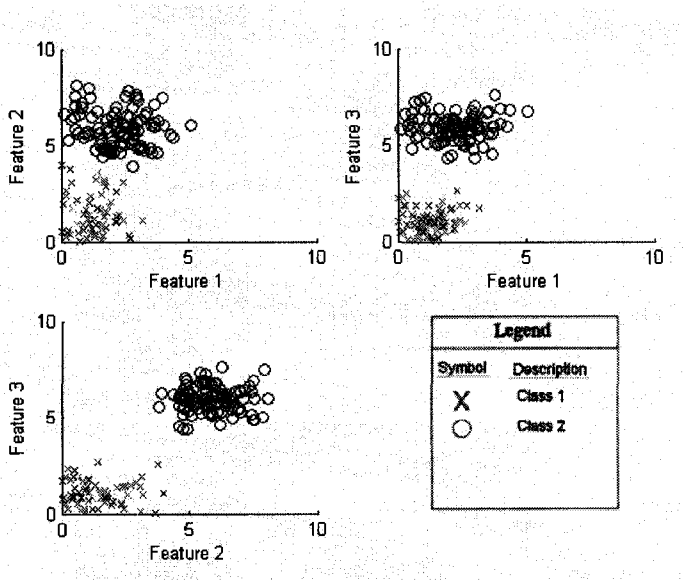


Figure 4.3: Ideal training data. The red crosses represents class 1 having mean=[1,1,1] and variance=[1,2,0.5], the blue circles represents class 2 with mean=[2,6,6] and variance=[2,1,0.5].

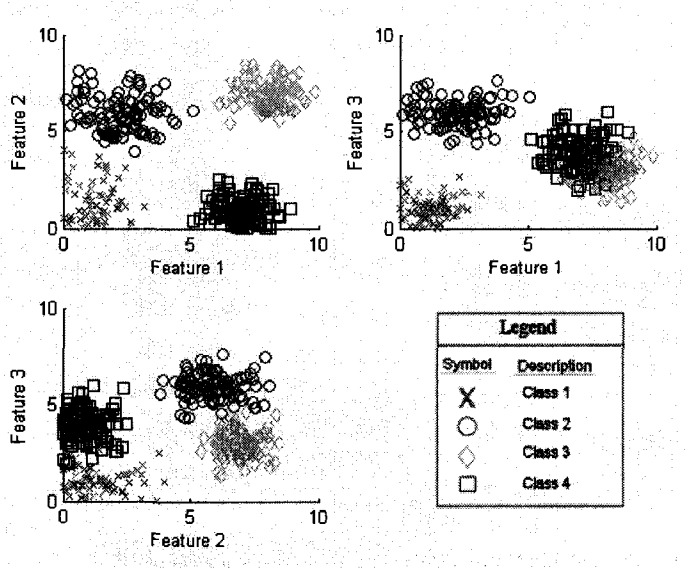


Figure 4.4: Ideal testing data. The red crosses represents class 1 with mean=[1,1,1] and variance=[1,2,0.5], the blue circles represents class 2 with mean=[2,6,6] and variance=[2,1,0.5], the green diamonds represents class 3 having a mean=[8,7,3] and variance=[0.75,0.5,0.75] and the black squares represents class 4 with mean=[7,1,4] and variance=[0.75,0.5,0.75].

4.3.2 Real data set

In order to test the algorithm on real features, 930 different sound files were taken from a sound bank of files that were each 30 seconds long and sampled at 20 kHz. Each file was converted into a single feature vector with the process shown in Figure 4.5. First the M1, M2 and Stationarity feature were extracted from each file, producing an array of feature vectors that is then buffered. The Buffer processing stage used was a simple averaging function, thus giving an average feature vector for each 30 second sound file.

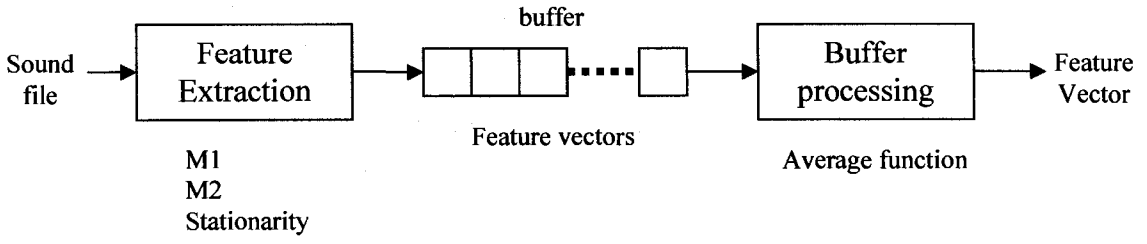


Figure 4.5: Process of lowering the adaptation rate and obtaining averaged feature vectors.

Table 4.2 shows the split of these files into a training set and testing set using a 7:3 ratio. This feature set is also illustrated in Figures 4.6 and 4.7.

Table 4.2: Number of sound files used from each class for training and testing

	Number of Sound files		
Class	**Training**	**Testing**	**Total**
Speech	336	144	480
Noise	203	88	291
Music	0	159	159
Total	539	391	930

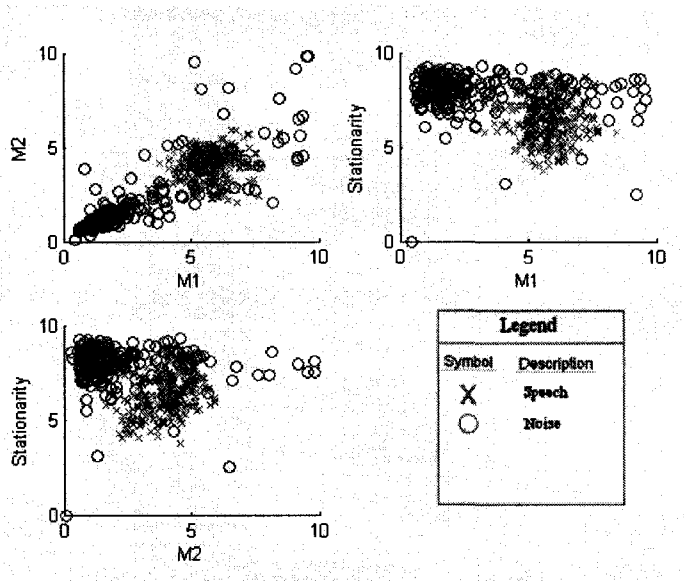


Figure 4.6: Real training data. Speech = Red crosses, and Noise = blue circles

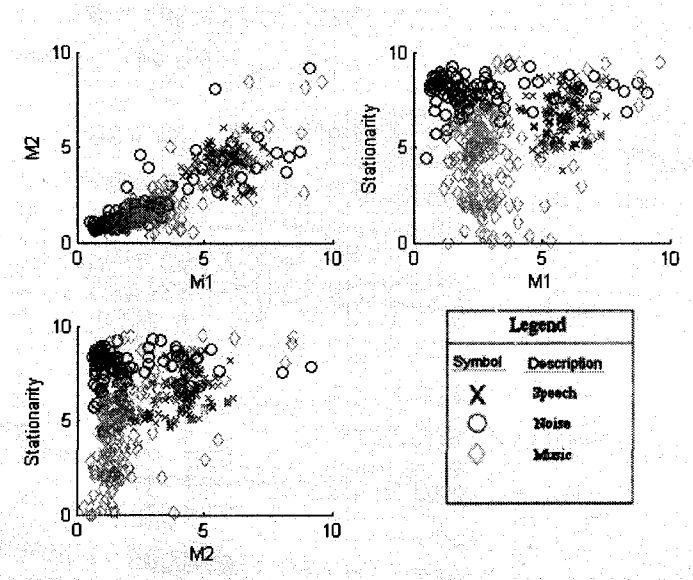


Figure 4.7: Real testing data. Speech = Red crosses, Noise = blue circles and Music = green diamonds

4.4 Data Storage

In order to use unsupervised learning to train the parameters for a new class, a training set of past features for each class needs to be readily available. Thus the hearing aid must store features as the adaptive system monitors the features for environmental changes. However, due to memory limitations, a hearing aid can only store a finite number of features for each class. For test purpose, the maximum number of features stored for each class is set to 100 and follows a first-in-first-out process when the maximum has been reached. This means that the system holds the 100 most recent features passed by the buffer for each class.

4.5 Measure of classification accuracy

The accuracy is measured by the hit Rate (HR), overall hit rate (OH), and False Alarm rate (FA). The HR is a measure of how many input samples were classified correctly for each class. The OH is simply the average HR over all the classes. The FA rate measures the

percentage of input data that belong to other classes that are falsely classified as belonging to the current class:

$$FA = \frac{N_{wrong}}{N_{total} - N_{class}} \tag{4.1}$$

where N_{wrong} is the number of input samples that are falsely classified into a given class, N_{total} is the total number of samples used for testing, and N_{class} is the number of samples belonging to this type of class.

4.6 Performance Measure

In order to measure the performance of an adaptive system, it is important to know how accurate the parameter estimates are, after a split or a merge. Therefore, as a reference, the ideal parameters that the adaptive system is trying to estimate need to be determined. The most optimum parameters can easily be found by training the parameters using a non-adaptive supervised learning using the same testing data used in the adaptive system that led to the adaptation. Therefore, in order to maximize the performance of the adaptive classifier, the error between the adaptive parameters and the supervised parameters need to be minimized. In order to determine the error between the adaptive parameters (A) and the best case supervised parameters ($\hat{A}$), the following two error measures are used.

4.6.1 Two-Norm

The two-Norm is an error measure that allows the error of a multi-dimensional vector to be represented by a single value. It is calculated as follows:

$$error = \left\| A - \hat{A} \right\|_F = \sqrt{\sum_{i=1}^{m} \left| a_i - \hat{a}_i \right|^2} \tag{4.2}$$

It is most useful when comparing different estimation methods, where the vector resulting in the smallest two-norm error is said to a most accurate estimate.

4.6.2 Frobenius Norm

Similar to the Two-Norm, The Frobenius Norm calculates a single value for a multi-dimensional matrix. It is calculated as follows:

$$error = \left\| A - \hat{A} \right\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} \left| a_{ij} - \hat{a}_{ij} \right|^2} \qquad (4.3)$$

4.7 Summary

In this chapter, the general framework for designing an adaptive system was developed. The design included adding a buffer stage that provides a more accurate assessment of the current environment for the purpose of adaptation, in addition to an adaptive algorithm that monitors the changes in the environment. The adaptive system is designed to first start with an initial classification system that allows the system to start with a set of predefined classes. A split criterion is then used to determine if there is enough of a change in environment to warrant a new class. Once this condition is met, the adaptive algorithm must use an unsupervised learning algorithm to learn the new class' parameters. A merge criterion is also needed to determine if a maximum number of classes has been reached and two classes need to be merged to make room for a new class. In addition to the adaptive framework, a set of test criterions are also described with the goal of testing the splitting and merging algorithms. First, the system is to be tested on an ideal set of data, where clusters are clearly defined. This is then followed by a real set of data where the three features; Modulation from 0 to 4 Hz (M1), Modulation from 4 to 16 Hz (M2), and Stationarity, are used on a set of test samples. The test samples are classified into the following classes: speech, noise and music.

In the following two chapters, two adaptive systems are designed as a solution to the adaptive framework introduced in this chapter.

Chapter 5

Proposed Adaptive Minimum Distance Classifier

As mentioned in section 4.2, an adaptive classification system requires the following process: An initial classification system that starts with predetermined classes, a split criterion and a merge criterion for the adaptive classification's monitoring algorithm, an unsupervised or clustering algorithm when the split criterion is met, and a merging algorithm for when the merge criterion is met. This process that makes up the adaptive classification system is designed based on the minimum distance classifier in this chapter.

5.1 Initial Classification System

The initial classification system uses the minimum distance classifier described in section 3.3.2. This is a basic classifier that only requires one statistical parameter for each class, namely the mean feature vector to represent the class center. The complexity of implementing this adaptive algorithm is low since, when a new class needs to be created, only the mean needs to be learned. In order to train the initial classification system, a supervised learning algorithm (described in section 3.5) must first determine the mean feature vector for each of the classes. This is simply done by isolating the training data for each class and calculating its mean vector. To illustrate the concept, a simple ideal two-class example, introduced in section 4.3.1, is used. Using the ideal training data of Figure 4.3, the calculated mean vectors for the initial classification system are shown in Table 5.1. The classified training data resulting from the supervised learning is plotted in Figure 5.1. In this

figure, the mean for each class is represented by the yellow square. Since the clusters are isolated and far apart, a perfect training accuracy is easily reached. This, however, is not a realistic expectation for the real case since the features are not ideal in nature, as will be seen.

Table 5.1: Feature mean for training set

Class	Mean
1	(1.05, 0.82, 0.95)
2	(2.07, 5.87, 5.94)

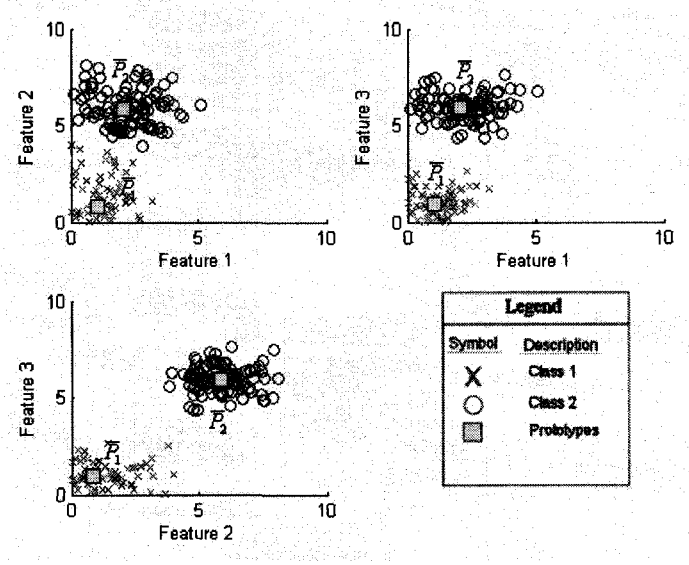


Figure 5.1: Ideal training data. Class 1 = red crosses, Class 2 = blue circles. Mean feature vectors = yellow squares

As will be seen in the next section, the splitting criterion is based on the self splitting competitive learning (SSCL) algorithm summarized in section 3.6.3 where a vector called the Prototype is used to represent each class' center. To keep this system consistent with the notation from the SSCL algorithm, the mean vector for a class will also be given the name prototype in this system and given the symbol $\vec{P}_i$ to represent class i. With these prototype vectors trained through supervised learning, the initial classes are set and the classifier is now ready to classify new feature vectors using the minimum distance classifier while monitoring changes.

5.2 Adaptive algorithm for the minimum distance Approach

The adaptive classification algorithm is the process of monitoring changes in the environment by continuously checking the split and merge criterions and updating the class parameters.

5.2.1 Split Criterion

The Split Criterion is the criterion that continuously checks for significant changes in the environment as each new vector coming from the input buffer is introduced (Figure 4.1). If the criterion is met, meaning a new and different class being consistently encountered, a flag is set for the algorithm to start its splitting process to learn the parameters for the new class. In section 3.6.3, the Self-Splitting Competitive Learning (SSCL) algorithm was summarized because it contains many similarities in its adaptive nature to the adaptive classification system being designed here. The SSCL algorithm first starts with one cluster and splits to find all natural clusters by first checking a split criterion. This is very similar to the adaptive classification system being designed here except that the latter starts with a set number of initial classes instead of just one cluster in the SSCL case. Fortunately, the split criterion used in the SSCL algorithm can easily be adapted to the needs of the split criterion for the adaptive classification system being designed. The split criterion is described below.

As each new feature vector is introduced by the buffer, the algorithm updates a vector called the Center Property Vector (CPV) which originates at the same position as the Prototype. This vector is updated according to the following equation.

$$\overline{C}_i^* = \overline{C}_i + \frac{1}{n_{\overline{C}_i}}\left(\overline{X} - \overline{C}_i\right) \tag{5.1}$$

where η_C is the step size, $\vec{X}$ is the current feature vector passed from the buffer, and i is the class index that $\vec{X}$ is classified as.

The movement of this vector is in the direction of the current feature vector $\vec{X}$ and the distance it travels is related to the distance between input vector $\vec{X}$ and the center property vector $\vec{C}_i$ being updated. Therefore, it tries to track the center position for its class from all the feature vectors introduced. The Split criterion is then checked according to the following statement:

$$\text{If } \left|\overline{P_i C_i}\right| > \varepsilon \text{ then split} \tag{5.2}$$

where ε is a constant threshold value.

As a new cluster pattern is introduced to the algorithm, the CPV closest to the new pattern will start to be attracted to the center of both clusters (as shown in Figure 5.2). As the CPV $\vec{C}_i$ crosses a threshold Euclidean distance ε from the prototype $\vec{P}_i$, and thus meeting the split criterion of Equation 5.2, the algorithm triggers the splitting process. Going back to our ideal example, by introducing the system to a new and distant cluster of input data, shown as the testing data of Figure 4.4, the split criterion for class 2 ($\vec{P}_2 = (1.19, 6.06, 5.99)$) was met once the center vector $\vec{C}_2$ traveled towards the new cluster and crossed the threshold ε, which was set to 2 (ε = 2 in Table 5.2). The data introduced to the system right before the splitting point was reached is shown in Figure 5.2. At this point, the algorithm must now learn the new prototype for the new cluster using an unsupervised learning algorithm which is described in the following section.

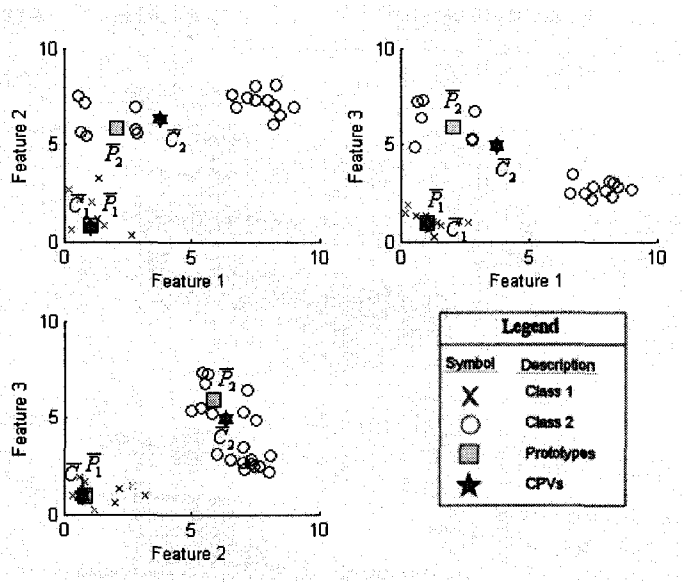


Figure 5.2: Test data introduced as the split criterion was met and the split process triggered. Class 1=red crosses, Class 2 = blue circles, prototypes = yellow squares, CPVs = magenta stars.

5.2.2 Splitting Process

In order to find the new prototype vector representing a new class, an unsupervised algorithm, or more specifically a clustering algorithm, is needed. Since we are looking to find one vector that represents the prototype, the K-means clustering algorithm of section 3.6.1 is most appropriate. This, however, requires data to be stored in order for the new prototype to converge to the new cluster. This is done according the method introduced in section 4.4, where past feature vectors are stored for each class in a K-by-S array, where K is the number of clusters. Each of the K classes will store up to S past values. This will allow the adaptive algorithm to isolate or combine specific clusters.

For our ideal example, the stored data classified as class 2 (the class that met the split criterion) is isolated (as shown in Figure 5.3) and clustered into two classes using the K-means algorithm. Table 5.3 shows the resulting prototypes for the three classes of the updated system. The new prototypes are also illustrated in the plot of Figure 5.4. Since class 2 was the class that was split, class 1 prototype is unchanged. The new class is represented by the prototype of class 3, while class 2 represents the splitting class. Class 2, represented

by $\bar{P}_2$, is slightly altered since the K-means algorithm also finds the new optimum center for class 2 based on the past training data. This may prove to be a desirable consequence of the clustering algorithm since it may happen that the features experienced by the hearing aid user for a given class may be slightly different than the samples used initially to train the class. This will be further investigated in section 5.2.2.

Table 5.2: Algorithm variables for ideal example

Variable	value
ε	2
$n_{\bar{C}_i}$	35

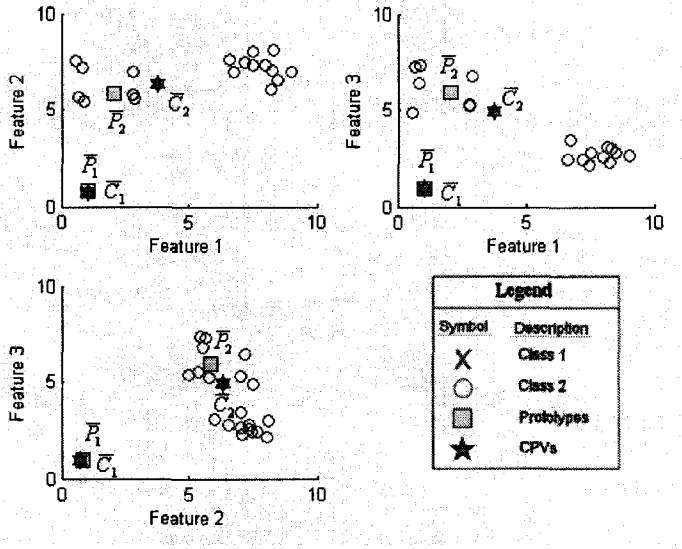


Figure 5.3: Data from class 2 when the split criterion was met and the split process triggered. Stored data for class 2 = blue circles, prototypes = yellow squares, CPVs = magenta stars.

Table 5.3: Prototype vectors for before and after split

Prototype Vector		
Class	**Before split**	**After split**
1	(1.05, 0.82, 0.95)	(1.05, 0.82, 0.94)
2	(2.07, 5.87, 5.94)	(1.19, 6.06, 5.99)
3	N/A	(7.84, 6.75, 2.88)

As Table 5.4 shows, an overall hit rate (OH) of 100% was achieved after splitting to accommodate for a third class. Again, this perfect accuracy is not a realistic expectation for the real case since the features are not ideal in nature.

Table 5.4: Test data accuracy after split

Test Accuracy after split						
	Class 1		Class 2		Class 3	
OH	HR	FA	HR	FA	HR	FA
100	100	0	100	0	100	0

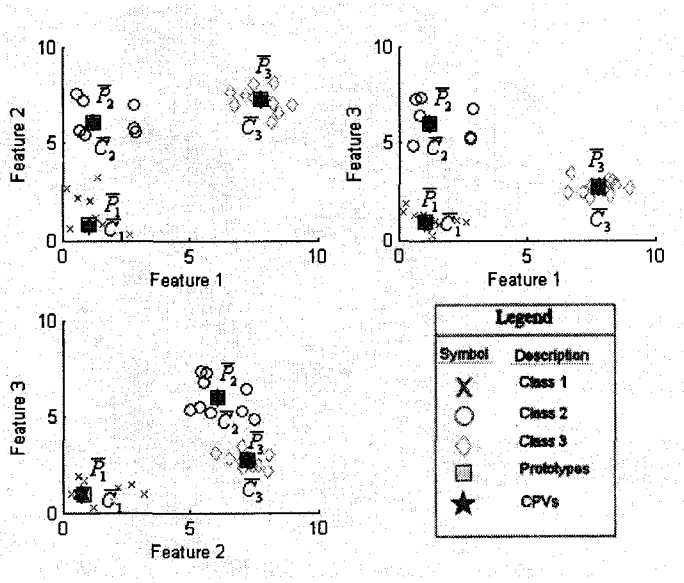


Figure 5.4: Past stored test data reclassified after splitting, thus creating a new prototype. Class 1 = red crosses, Class 2 = blue circles, Class 3 = green diamonds, prototype = yellow squares, CPV = magenta star.

5.2.3 Merge Criterion

A merge will take place when the maximum number of classes has been reached and the Split Validity criterion is met. This maximum is important because it provides the algorithm with an upper limit on the amount of splitting taking place based on the available storage and complexity. If too many classes are allowed to exist, the user may find it difficult to set new hearing aid settings for every class. Once this merge condition is met, the algorithm must merge two classes based on their similarity.

5.2.4 Merging Process

Since the only information we have about the classes are their respective prototype vectors $\vec{P}$, the deciding factor for which two classes to be merged is limited. For the purpose of this thesis, the two classes having the minimum Euclidean distance are chosen as the two merging classes. In the future, hearing aid settings may be used to influence this decision process so that classes with closest hearing aid settings are merged. However this information is not available at this time (see section 7.3.5). Once the merge criterion is met, the algorithm simply needs to recalculate a mean feature vector for the new combined Prototype of the two classes to be merged.

In order to demonstrate a merging event with in our ideal example, we set the maximum number of classes to three and introduce a fourth class (shown in the testing data of Figure 4.4). Figure 5.5 shows the point at which the input data triggers the split and merge to occur. First, class 3 is split to reach one class over the maximum, as shown in Figure 5.6 and Table 5.5. Then, the algorithm must choose two classes to merge to reduce the number of classes back down to three.

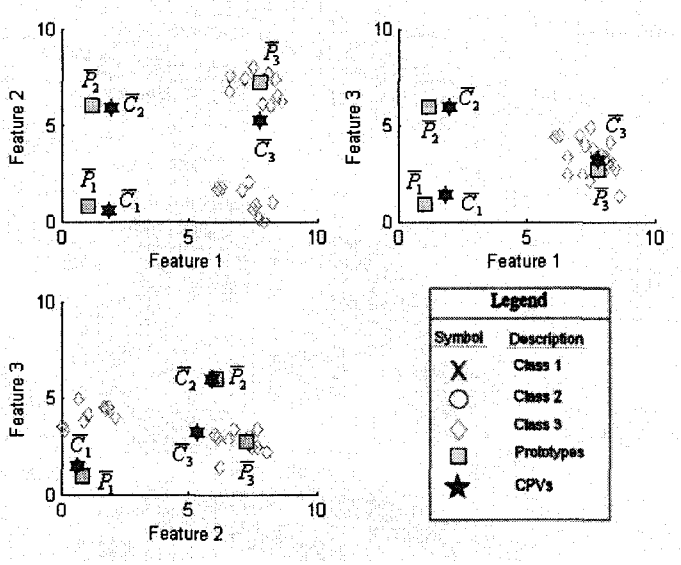


Figure 5.5: Past class 3 test data (green diamonds) right before triggering the split process. Prototypes = yellow squares, CPVs = magenta star.

Table 5.5: Prototype vectors for before and after split

Prototype Vector		
Class	**Before split**	**After Split**
1	(1.05, 0.82, 0.94)	(1.05, 0.82, 0.94)
2	(1.19, 6.06, 5.99)	(1.19, 6.06, 5.99)
3	(7.84, 6.75, 2.88)	(7.78, 7.03, 2.68)
4	N/A	(7.33, 1.12, 4.14)

Table 5.6: Test accuracy of test data after Class 3 split to create Class 4. Note: Accuracy is calculated after reclassifying all of the test data into the current classes.

Test Accuracy after split								
	Class 1		**Class 2**		**Class 3**		**Class 4**	
OH	**HR**	**FA**	**HR**	**FA**	**HR**	**FA**	**HR**	**FA**
100	100	0	100	0	100	0	100	0

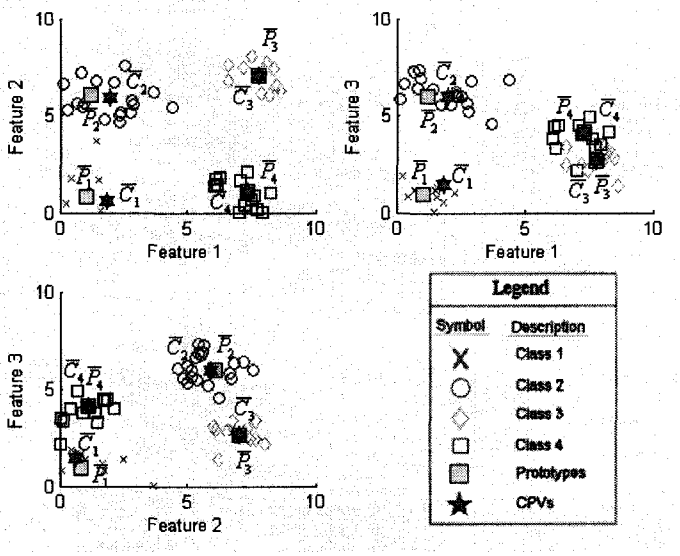


Figure 5.6: All past test data is reclassified right after the split that resulted in a fourth class. Class 1 = red crosses, Class 2 = blue circles, Class 3 = green diamonds, Class 4 = black squares. Prototypes = yellow squares, CPVs = magenta star.

The two classes having the smallest Euclidean distance from each other were found to be classes 3 and 4 ($\vec{P}_3 = (7.78, 7.03, 2.68), \vec{P}_4 = (7.33, 1.12, 4.14)$). Therefore, by combining the stored test data for these two classes and calculating a combined mean, the two clusters are replaced with one merged class. The new prototypes are shown in Table 5.7 and illustrated in Figure 5.7.

Table 5.7: Prototype vectors for before and after Merge. Classes 3 and 4 were merged into new class 34.

Before Merge		After Merge	
Class	**Prototypes**	**Class**	**Prototypes**
1	(1.05, 0.82, 0.94)	**1**	(1.05, 0.82, 0.94)
2	(1.19, 6.06, 5.99)	**2**	(1.19, 6.06, 5.99)
3	(7.78, 7.03, 2.68)	**34**	(7.58, 4.37, 3.34)
4	(7.33, 1.12, 4.14)		

Table 5.8: Test data accuracy after merge

Test Accuracy after merge						
	Class 1		Class 2		Class 3	
OH	**HR**	**FA**	**HR**	**FA**	**HR**	**FA**
100	100	0	100	0	100	0

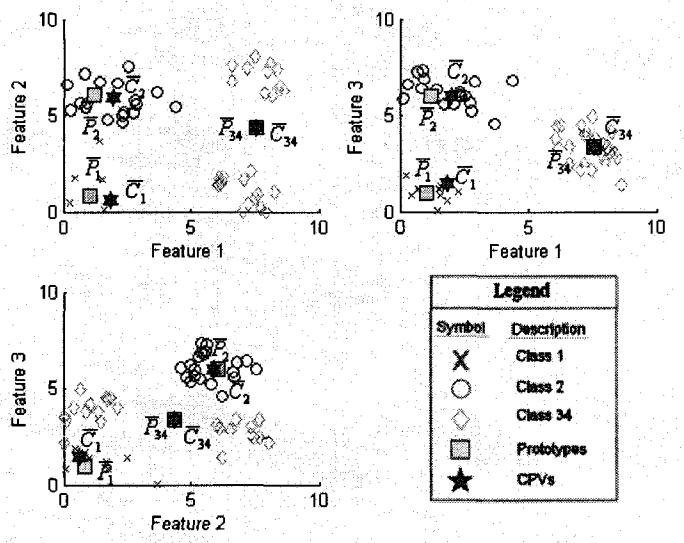


Figure 5.7: All past stored test data are reclassified after merging class 3 and 4. Class 1= red crosses, Class 2 = blue circles, new Class 34= green diamonds. .Prototypes = yellow squares, CPVs = magenta star.

Now that the system is shown to function under ideal conditions, the ideal features are replaced with features extracted from a bank of environmental sound files, listed in Table 4.2.

5.3 Test with Real Features

In order to test the adaptive system for a more practical case, input data derived from two environments, speech and music, was used to train the initial classification system. The initial prototype vectors were set by calculating the mean vector for Speech and Music based on the training data of Table 4.2. The resulting prototypes are shown in Table 5.9 and classified and plotted in Figure 5.8. The CPVs (magenta stars) are initially set equal to the Prototypes (yellow squares). The training accuracy was then calculated and summarized in Table 5.10. As previously mentioned, it is unrealistic to expect the training to have perfect accuracy since the features have some overlap and, in the case of noise, some of its samples are uncharacteristic and end up far away from the noise's mean and much closer to other classes. This is the reason that the noise class has a much lower training accuracy than speech.

Table 5.9: Feature mean for training set

Class	Prototype Vector
1	(5.81, 4.09, 6.49)
2	(2.83, 2.07, 7.92)

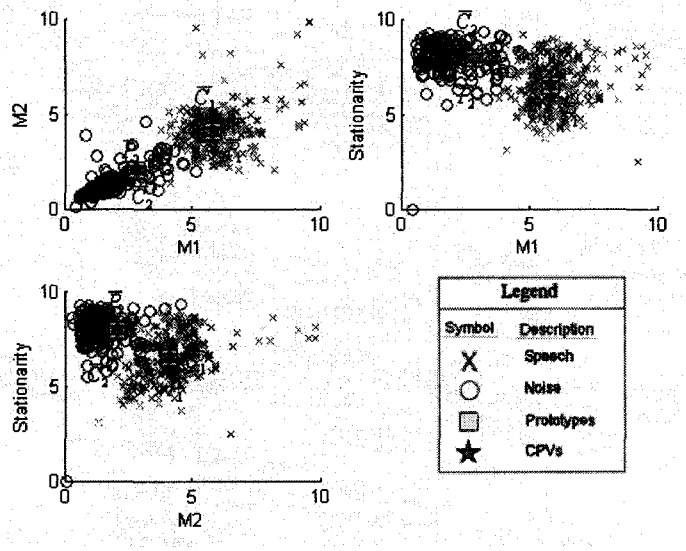


Figure 5.8: Classified training data for initial system. Speech = red crosses, noise = blue circles, Prototypes = yellow squares, CPVs = magenta stars.

Table 5.10: Training Accuracy for Speech and Noise in real case scenario

Training Accuracy				
	Speech		Noise	
OH	HR	FA	HR	FA
88.7	96.5	19.2	80.8	3.6

Now that the initial classification system is in place, we can start simulating environmental changes by introducing the adaptive algorithm to a new environmental class. Specifically, the Music class listed in the testing data of Table 4.2.

5.3.1 Splitting

The adaptive algorithms variables, ε, which specifies the splitting threshold distance of the Center Property Vector to its prototype, is set to 2 and the winning counter $n_{\bar{C}_i}$ for the CPV's update (Equation 5.1) is set to 35, as specified in Table 5.11. These two values are set to allow enough data to accumulate in order to accurately estimate the new class' prototype. With the music class present, the classifier will at first misclassify the music test samples as either Speech or Noise depending on its distance to their respective Prototypes. However, since the music test samples are from a separate cluster, the CPV of the closest cluster will start to follow the new pattern generated by the music. In this case, the CPV $\vec{C}_2$ (representing Noise) was pulled towards the music cluster and away from its Prototype $\vec{P}_2$ by a distance greater than ε and triggered a split, as shown in Figure 5.9. It is also possible that some of the music samples were misclassified as speech, however not enough to move $\vec{C}_1$ away from $\vec{P}_1$ at a distance greater than ε. The point at which the algorithm meets this split criterion is illustrated in Figure 5.9.

Table 5.11: Algorithm variables for real case tuned to allow enough data from new class to accumulate for better estimation of its statistical parameters.

variable	value
ε	2
$n_{\bar{C}_i}$	35

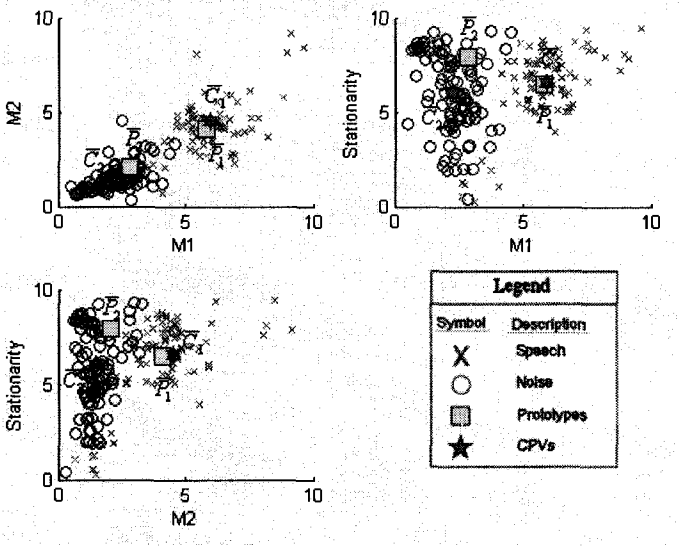


Figure 5.9: Noise class meeting the split criterion and triggering the split process. Speech=red crosses, Noise = blue circles, Prototypes = yellow squares, CPVs = magenta star.

Now that the Split Criterion is met, the algorithm runs its splitting algorithm. This is where the K-means clustering algorithm, with a K set to 2, is used on the isolated data that was classified as Noise as shown in Figure 5.10. The K-mean algorithm produced the two most optimum prototype vectors for the isolated data. The resulting class prototype vectors after adaptation are summarized in Table 5.12 and illustrated in Figure 5.12. Table 5.12 also includes, as a comparison, the desired optimum vectors obtained through the supervised learning algorithm described in section 3.5 using the test data as the training set (also illustrated in Figure 5.11). This allows us to compare the results from the adaptive algorithm, which uses unsupervised learning, to the ideal results obtained through supervised learning. In order to measure the performance, the error between the adaptively estimated prototypes and the supervised case is also shown (as described in section 4.6).

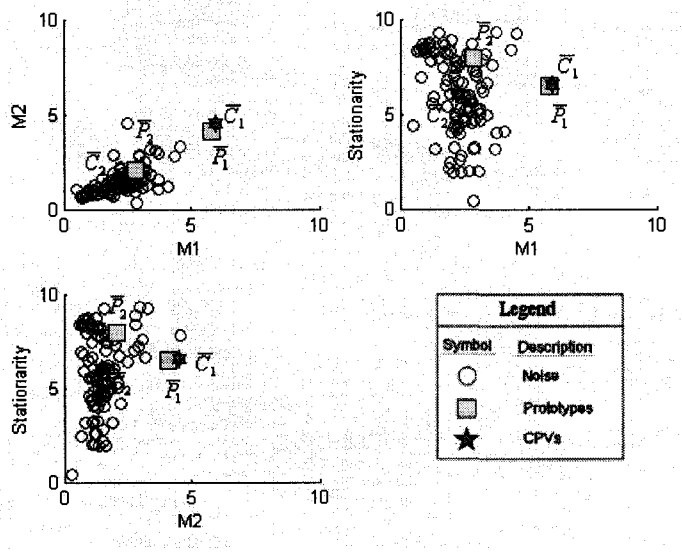


Figure 5.10: Isolated noise class meeting the split criterion and triggering the split process. Stored data for noise = blue circles, Prototype = yellow squares, CPVs = magenta star.

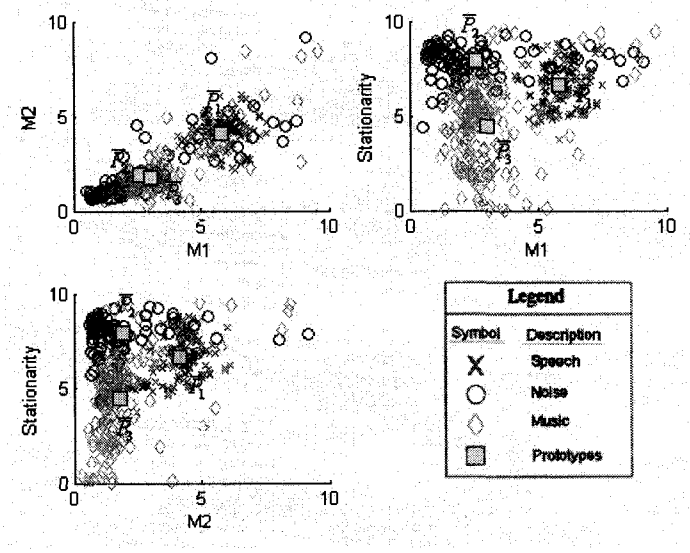


Figure 5.11: All test data with prototypes that were obtained with supervised training. Speech = red crosses, Noise = blue circles, and Music = green diamonds. Prototypes = black squares.

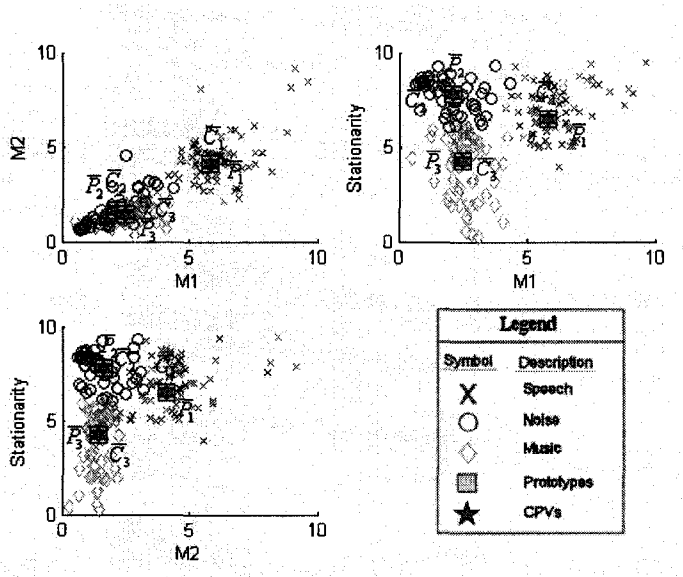


Figure 5.12: All Stored test data after splitting noise class using K-mean clustering algorithm. Speech = red crosses, Noise = blue circles, Music = green diamonds. Prototype = yellow squares, CPVs = magenta star.

Table 5.12: Prototype vectors for the three classes after using k-means algorithm to split the noise class into two, creating the new class music. Error is measured between prototype obtained through the adaptive algorithm and the prototype vectors obtained through supervised learning for the three classes.

Class	Prototype		
	Supervised	After K-Means clustering	
	Vector	Vector	Error
Speech	(5.83, 4.12, 6.62)	(5.81, 4.09, 6.49)	0.14
Noise	(2.58, 1.89, 7.93)	(2.11, 1.63, 7.81)	0.56
Music	(3.00, 1.80, 4.43)	(2.48, 1.40, 4.25)	0.68
Total Error			**1.38**

Table 5.13 shows the test accuracy after adaptively adding the Music class. The accuracy in classifying the testing data after adapting to include the Music class is 1.4% lower in overall hit rate (OH) than the optimum supervised case. Therefore, there is still room for improvement in parameter estimation. In order to improve the performance of the adaptive system, the total error in Table 5.12 needs to be reduced to obtain estimates that are closer to the parameters obtained through the non-adaptive supervised learning algorithm.

Table 5.13: Test Accuracy for splitting algorithm.

Testing Accuracy (%)							
		Speech		Noise		Music	
	OH	HR	FA	HR	FA	HR	FA
Supervised	85.8	97.3	9.6	81.1	8.5	78.9	2.1
After K-Means	84.4	98.6	10.8	80.0	10.1	74.5	1.3

5.3.2 Improving parameter estimation

In order to improve the accuracy to better resemble a classification system obtained through supervised learning, four post-splitting processes are explored. Since the new prototypes were estimated with the isolated data of one class, it is worthwhile to recalculate the statistical parameter with all of the classes present. For this we introduce a recalculation step, where all of the stored data is reclassified and subsequently the parameters are recalculated. It is important to note that only the parameters for the splitting class and the new class may be modified since there is no guarantee that there is enough stored data for the other classes to accurately estimate a new parameter. In addition, the option of keeping the original parameters for the splitting class instead of replacing it with that obtained through the K-means clustering algorithm may be explored. Table 5.14 lists four different post-processes that attempt to determine the best combination of options that will result in improving the classification accuracy.

Table 5.14: Post-Splitting (PS) methods in trying to improve the classification accuracy for the adaptive system.

Post-Splitting Method	Description
PS1	**Recalculating parameters for modified classes:** The parameters are recalculated for the splitting class and the new class using the stored features after being reclassified.
PS2	**Recalculating parameters for only new class:** The parameters are recalculated for the new class only, after the stored features are reclassified.
PS3	**Keeping splitting class parameters unchanged:** This post-splitting method leaves the original parameters for the splitting class unchanged, thus only using one set of parameters produced by the EM clustering algorithm.
PS4	**Recalculating parameters for only new class and keeping splitting class parameters unchanged:** combines method 2 and 3.

Table 5.15 shows the calculated error between the prototypes altered by the post-splitting processes and the prototypes obtained through the supervised learning algorithm. As the table shows, both of the errors for Noise and Music were increased by recalculating the parameters, therefore according to this test, recalculating the parameters decreases the performance of the system. This is also shown in the test accuracy of Table 5.16, where the overall hit rate (OH) decreases by 0.3% (from 84.4% to 84.1%). It may be slightly misleading to only look at the HR since in the case of Noise, the hit rate seems to increase while the error is increased from the K-means to PS1. However, it is also important to see that the supervised parameter also aims to minimize the false alarm rate (FA). In this case the FA is increased by 1.6% for noise after applying PS1. The lowest total error results in applying PS3 where the prototype for music (the splitting class) is unchanged from the K-means clustering and no recalculation steps are performed. By applying PS3 the accuracy is increased by 1.6% and closely matches the supervised accuracy, therefore making PS3 the better post-splitting process in increasing the adaptive systems performance for this case. At a first glance, one might think that the adaptive system using PS3 outperforms that of the supervised case, since the OH is slightly higher. However, we also have to take the FA rate into consideration. The HR for speech using PS3 is higher than the supervised case (98.6% compared to 97.3%), however the FA is also higher (10.0% compared to 9.6). For Noise, the HR and FA for PS3 are shown to be the same as the supervised case. For Music, the

compromise is made in having a lower FA rate, with the consequence of having a slightly lower HR. It is important to note that, even though PS3 was successful in increasing the accuracy for this particular case, more cases need to be tested before reporting with confidence that the post-splitting process increases the accuracy in general.

Table 5.15: Comparing error in Prototype vectors for four post-splitting (PS) processes.

Class	Prototype Error				
	K-Means	PS1	PS2	PS3	PS4
Speech	0.14	0.14	0.14	0.14	0.14
Noise	0.56	0.62	0.56	0.31	0.31
Music	0.68	0.78	0.78	0.68	0.75
Total	**1.38**	**1.54**	**1.48**	**1.13**	**1.20**

Table 5.16: Comparing Test Accuracy for four post-splitting (PS) processes.

Testing Accuracy (%)							
		Speech		Noise		Music	
	OH	HR	FA	HR	FA	HR	FA
Supervised	85.8	97.3	9.6	81.1	8.5	78.9	2.1
After K-Means	84.4	98.6	10.8	80.0	10.1	74.5	1.3
PS1	84.1	98.6	10.8	82.2	11.7	71.4	0.4
PS2	83.7	98.6	10.8	81.1	11.7	71.4	0.8
PS3	**86.0**	**98.6**	**10.0**	**81.1**	**8.5**	**78.3**	**1.3**
PS4	84.5	98.6	10.0	81.1	10.7	73.9	1.3

Table 5.17 shows the prototypes as a result of PS3 with their respective errors. Figure 5.13 shows the resulting classified test features using the updated parameters.

Table 5.17: Prototypes for the three classes after using k-means algorithm to split the noise class into two and then applying the post-splitting process 3 (PS3) described in table 5.14. Error is measured between prototype obtained through the adaptive algorithm and the prototype obtained through supervised learning for the three classes.

Class	Prototype		
	Supervised	After K-Means +PS3	
	Vector	Vector	Error
Speech	(5.83, 4.12, 6.62)	(5.81, 4.09, 6.49)	0.14
Noise	(2.58, 1.89, 7.93)	(2.83, 2.07, 7.92)	0.31
Music	(3.00, 1.80, 4.43)	(2.48, 1.40, 4.25)	0.68
Total Error			1.13

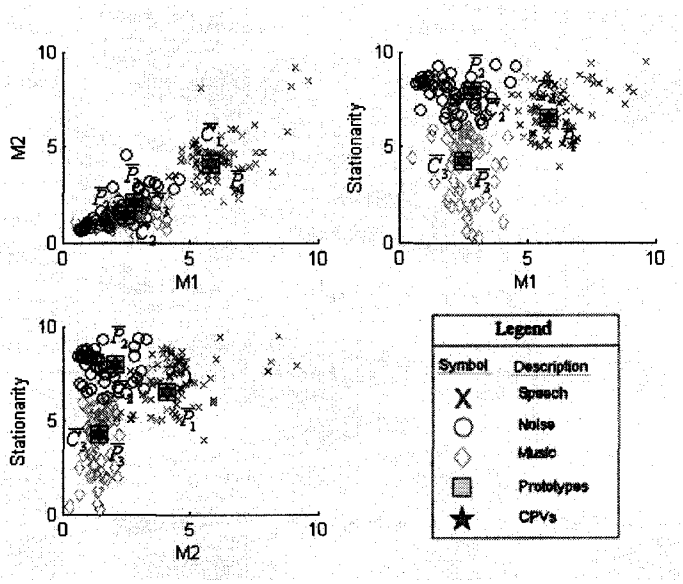


Figure 5.13: All Stored test data after splitting noise class using K-mean clustering algorithm and then applying the post-splitting process 3 (PS3) described in table 5.14. Speech = red crosses, Noise = blue circles, Music = green diamonds. Prototype = yellow squares, CPVs = magenta star.

5.3.3 Merging

In order to test the merging algorithm, the system was set to have a maximum of two classes. Thus, by introducing a third (Music) class in the previous section, the system was one over its maximum and must now merge two classes. Noise and Music were the chosen classes to merge based on the smallest Euclidean distance between prototype vectors. In practice, merging these two classes may not seem logical since the two classes are very different environments and most likely carry very different hearing aid settings. However, it is important to note that once a larger number of classes are allowed to exist, classes have the freedom to split into subclasses and thus these subclasses would be a more likely choice to merge according to cluster separation. However, this hypothesis will not be addressed in this thesis since with only 3 features, a fourth class, speech in noise for example, was not successfully able to be detected as a new and different class with this system. This point will be further discussed in section 7.1.

The resulting class prototypes after merging are shown in Table 5.18, along with the prototype vectors learned by supervised training on the testing data, where the classes Music

and Noise were labeled as one class and Speech labeled as another. The error in the prototype vectors for the Music/Noise class is still high at 0.56. However, since we only have two classes it does not show to have much of an effect on the accuracy (Table 5.19), where the HR for Music/Noise class is only 2.3% lower than the supervised case. The main reason for this high accuracy is that we have allowed only 2 classes, and since the speech class had a very high accuracy, most of the false alarm rate (FA) for the Noise and Music classes were attributed to each other's misclassifications. Therefore, by combining the two together, the problem of misclassifying the features of one class as the other disappears and, in this case, even reduces the false alarm rate to 1.4%.

Table 5.18: Prototype vectors after merging music and noise class. Error measures the difference between the estimated prototypes and the actual (supervised) prototype vectors.

Class	Prototype vectors		
	Supervised	After Merge	
	Vector	Vector	Error
Speech	(5.83,4.12, 6.62)	(5.81, 4.09, 6.49)	0.14
Music/Noise	(2.85, 1.83, 5.68)	(2.40, 1.54, 5.50)	0.56
Total Error			0.70

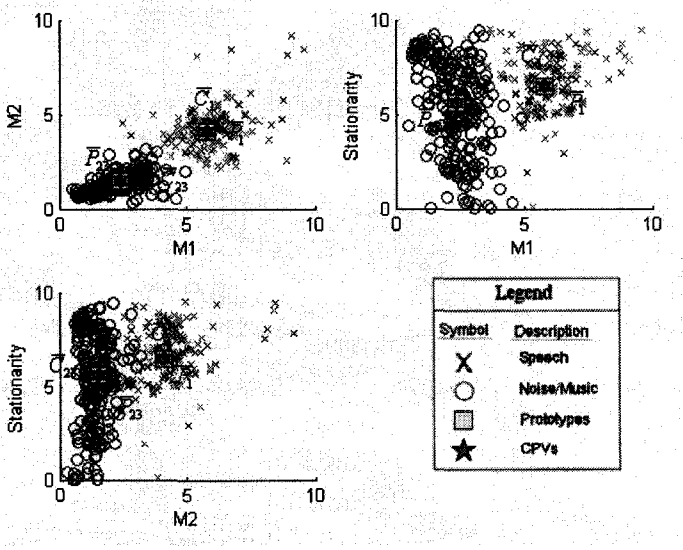


Figure 5.14: All Classified test data after merge. Speech = red crosses, Music and Noise = blue circles. Prototypes = yellow squares and CPV = magenta stars.

Table 5.19: Accuracy after merging Noise and Music classes

Testing Accuracy (%)					
		Speech		Noise and Music	
	OH	HR	FA	HR	FA
Supervised	93.0	98.0	12.0	88.0	2.1
After Merge	92.2	98.6	14.3	85.7	1.4

5.4 Summary

In this chapter, we propose a simple adaptive minimum Euclidean distance classification system using the three features: modulation from 0 to 4 Hz (M1), modulation from 4 to 16 Hz (M2), and stationarity. This very basic system was successfully able to track the changes when a new class was introduced using a Split Criterion based on the Self-Splitting competitive learning (SSCL) algorithm summarized in section 3.6.3. Using the K-means clustering algorithm to learn the new class' parameter, once the split criterion was met, an overall hit rate of 84.4% was obtained compared to 85.8% for the non-adaptive (supervised) case. The accuracy was successfully increased to 86.0% by means of a post-splitting process where the original parameters for the splitting class were kept unaltered, thus matching the accuracy obtained if the system was learned with a supervised learning algorithm. By setting the maximum number of classes to two, the merge criterion was met after splitting. Based on the minimum Euclidean distance, the two closest classes selected for merging were Noise and Music. Once the two classes merged, an overall hit rate of 92.2% was achieved, which is comparable to the 93.3% obtained through a supervised learning algorithm. The classes were also shown not to be distributed equally for all features (Figure 5.11) and thus are not spherical in cluster shape. Therefore, in order to increase the accuracy further, a more accurate classifier needs to be considered. In the next chapter, an adaptive classification system is designed using the, more complex, Bayesian classifier.

Chapter 6

Proposed Adaptive Bayesian Classifier

In the previous chapter an adaptive minimum distance classifier was proposed. By the using the minimum Euclidean distance classifier, the classes are assumed to be spherical in shape. However, by further analyzing the class distribution for each feature, it was shown that some of the classes have more of an elliptical shape. Therefore a classifier that does not assume that all classes are spherical in shape should improve the classification accuracy for the adaptive classification system.

In this chapter a second System based on the Bayesian classifier is proposed, where the features are classified based on the maximum likelihood measure. In a Bayesian classifier, each class is associated with a distribution. In this case, the distribution is assumed to be Gaussian in shape, therefore requiring a mean and covariance matrix. In order to split a class, an unsupervised learning algorithm is needed to detect the new class distribution. In other words, it must detect the mean and covariance matrix for the new class. This requirement fits the characteristics of the Expectation Maximization (EM) clustering algorithm, which estimates the Gaussian distribution for a pre-determined number of clusters. The following sections describe in more details the system design parameters.

6.1 Initial Classification System

The classification system selected is the Bayesian classifier described in section 3.3.1, therefore the supervised learning algorithm must determine a mean and covariance matrix

for each of the initial classes. These statistics are then used as the initial parameters for the Bayesian classifier. Once this system is in place, features are classified into one of its pre-defined classes.

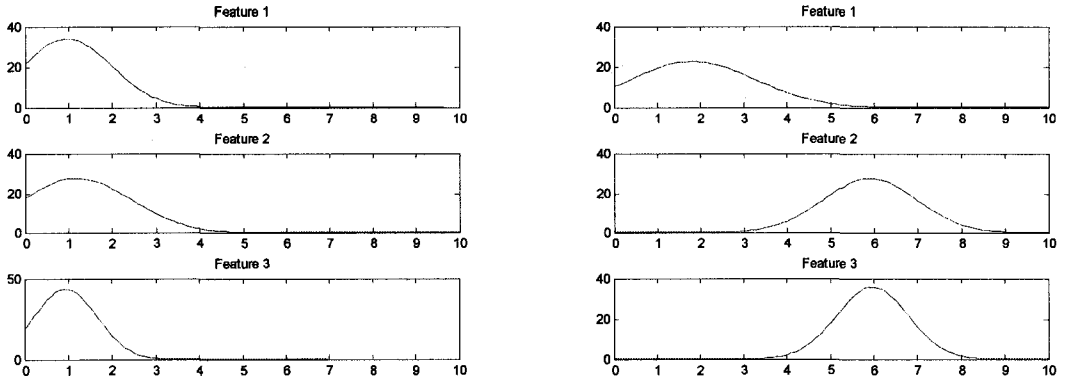
As a simple demonstration of how the initial classification system functions, we again turn to the ideal data example that was introduced in section 4.3.1. The features are selected to have distinct clusters for each class. The mean and the covariance matrix, shown in Tables 6.1 and 6.2, are computed using the ideal training data of Table 4.1 and Figure 4.3. As mentioned earlier, the distribution for each class is estimated by assuming that the data is Gaussian distributed as shown in Figure 6.1, where the Probability Distribution Functions (PDFs) for each class and each feature are shown.

Table 6.1: Feature mean for ideal Gaussian training data set

Class	Mean
1	(1.05, 0.82, 0.94)
2	(2.07, 5.87, 5.94)

Table 6.2: Feature covariance matrices for ideal Gaussian training data set

Covariance Matrix	
Class 1	Class 2
$\begin{bmatrix} 0.75 & -0.02 & 0.01 \\ -0.02 & 1.78 & 0.00 \\ 0.01 & 0.00 & 0.46 \end{bmatrix}$	$\begin{bmatrix} 0.75 & -0.02 & 0.01 \\ -0.02 & 1.78 & 0.00 \\ 0.01 & 0.00 & 0.46 \end{bmatrix}$



a) class 1 **b) class 2**

Figure 6.1:Probability Density Function (PDF)

Since the clusters are ideal in terms of clearly being isolated clusters, the training accuracy obtained was 100%.

At this point, the initial classification system is in place and the system is ready to start classifying features into class 1 or class 2. The algorithm also starts monitoring these features for environmental changes with the adaptive classification algorithm.

6.2 Adaptive Classification for the Bayesian Classifier Approach

Once the ideal test data of Figure 4.4 is introduced, with the two new classes present, the algorithm continuously checks the split criterion described below to see if there is enough change to warrant a split.

6.2.1 Split Criterion

Since the clusters are identified by their covariance matrix in addition to the mean, we can implement a slightly more complex splitting criterion than the one implemented in the adaptive minimum distance classifier of the previous chapter.

Using the covariance matrix as determined in the training stage, we now have the statistical information that can estimate the distribution of each class. If we assume a Gaussian

distribution we could associate probabilities to the introduced feature vectors in order to determine if a new class should be created. In other words, if a significant number of feature vectors are being classified in a given class A, but with very low probability, then it is possible that they actually belong to a new class, and therefore the class A should split into two.

For a Gaussian distribution, the cumulative probability is given by:

$$P(X \leq x) = \frac{1}{2}\left[1 + erf\left(\frac{x-\mu}{\sqrt{2}\sigma}\right)\right] \qquad (6.1)$$

where μ is the mean, σ is the standard deviation and erf() is the mathematical error function.

In order to track data that is being classified in a given class A, but with very low conditional probability ($p(\vec{x}|A)$), we determine the percentage of features that lie outside some classification area with a radius of R. According to equation 6.1, if we create a boundary at a distance of R=(x-σ)=2σ from the mean, we know that 97.72% of the time, features belonging to this class will lie within this area. This means that only 2.28% of the time a feature belonging to this class will lie outside of this area and therefore may likely belong to some new class yet to be discovered. The area within this boundary that holds 97.72% of the data will be referred to as the 98% classification area. In order to trigger a split, a threshold percentage (PercThr) of the data lying outside this area is introduced. For example if 30% of the testing features are outside the area defined by R=2σ from the mean, the class is split into two.

In order to determine if a point lies outside the classification area, the Mahalanobis distance is used. This measure calculates the distance between two points normalized with the standard deviation (σ) (see section 3.8.2). For example, if a point is at a distance of twice the standard deviation (2σ) from the mean, then the Mahalanobis distance will equal to 2. Therefore we can set our 98% classification area boundary at a Mahalanobis distance of d=2.

One problem that may arise with this threshold measure is if there is not enough data present, the split may take place prematurely. For example, if one of the first two test features introduced lies outside the classification area, then the percentage is already at 50% and a split will be triggered. In order to compensate for this problem we must set a volume threshold as well. Therefore, we only perform the split check after introducing a minimum amount of test feature vectors, denoted as the Feature counter (F_count).

To demonstrate the split criterion, the ideal test data of Table 4.1 and Figure 4.4 is introduced to the ideal initial classification system of the previous section and plotted with its 98% classification area in Figure 6.2 below. The testing data contains two new distinct classes as well as classes that are similar to the ones used for the initial classification training. The system parameters are set as followed:

Table 6.3: Algorithm Variables for the adaptive Bayesian classifier

Variable	Description	Value
K_max	Maximum number of classes	3
S	maximum features stored for each class	100
F_count	Number of features store before a split can occur	120
Thr	Percentage of features lying outside the classification area	35%

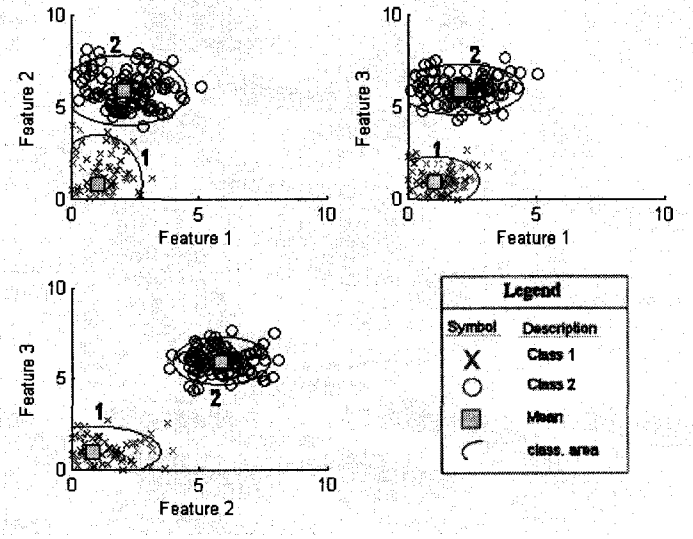


Figure 6.2: Ideal training data with trained mean and covariance matrices. Class 1 = red crosses, Class 2 = blue circles. The means = yellow squares and covariance matrix is represented by drawing the classification area (big blue circles)

As the new testing data is introduced, the algorithm monitors the percentage of feature vectors that lies outside of the 98% classification areas, until the splitting criterion is met. Table 6.4 shows the amount and percentage of outliers for each class right before the split takes place. Since the percentage of outliers for class 2 is greater than the threshold which was set to 35% (Table 6.3) and the number of data points stored for the splitting class is greater that 120, which was the threshold value of Table 6.3, a split is triggered and the algorithm begins the splitting process.

Table 6.4: Outlier history right before split

Class	# of outliers	Total	% of outliers
1	12	57	21
2	69	120	58

6.2.2 Splitting Process

Now that the system has detected that there is a significant amount of input feature vectors that are being classified with very low probability, represented by the percentage lying outside the 98% classification area in Figure 6.3, the algorithm must create a new class. Similar to the adaptive minimum distance classifier, past feature vectors also require to be stored in a K-by-S array, where K is the number of clusters and S is the variable that sets the number of past stored feature vectors for each class (see section 4.4). In order to determine the new class's distribution, the EM clustering algorithm described in section 3.6.2 is used on the isolated data belonging to the splitting class (Figure 6.4). The algorithm divides the stored features belonging to the splitting class into two clusters and thus finding two new means and covariance matrices. The updated statistics for the ideal example are shown in Tables 6.5 and 6.6. The resulting classified test data are shown in Figure 6.5. The system may now return to classifying inputs to one of the three classes using the Bayesian Classifier.

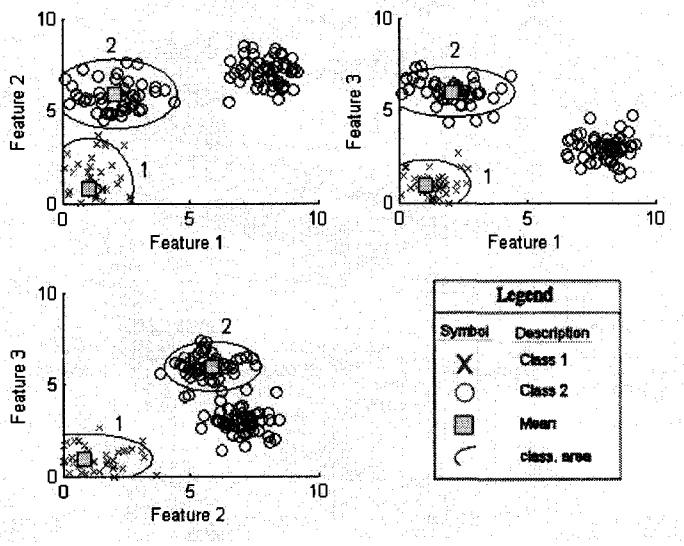


Figure 6.3: Testing data introduced that met the split criterion and triggered the split process. Class 1=red crosses, Class 2 = blue circles, means = yellow squares, classification areas = large blue circles.

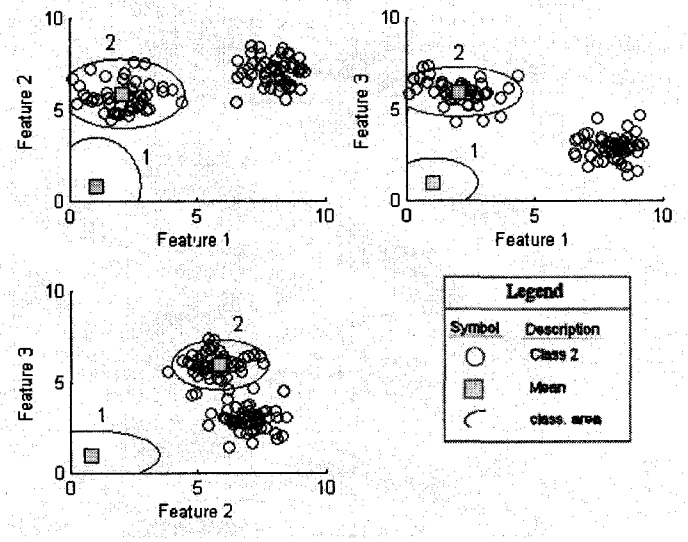


Figure 6.4: Isolated Data from class 2 meeting the split criterion and triggered the split process. Stored data for class 2 = blue circles, means = yellow squares, classification areas = large blue circles.

Table 6.5: Feature mean after splitting class 2 (Table 6.1) using EM algorithm

Class	Mean
1	(1.05, 0.82, 0.94)
2	(1.85, 5.66, 5.95)
3	(8.05, 7.06, 2.90)

Table 6.6: Feature covariance matrices after splitting Class 2 (Table 6.2) using EM algorithm

Covariance Matrix		
Class 1	**Class 2**	**Class 3**
0.75 -0.02 0.01 -0.02 1.78 0.00 0.01 0.00 0.46	1.48 0.15 -0.17 0.15 0.62 0.07 -0.17 0.07 0.39	0.53 -0.03 -0.01 -0.03 0.52 -0.03 -0.01 -0.03 0.43

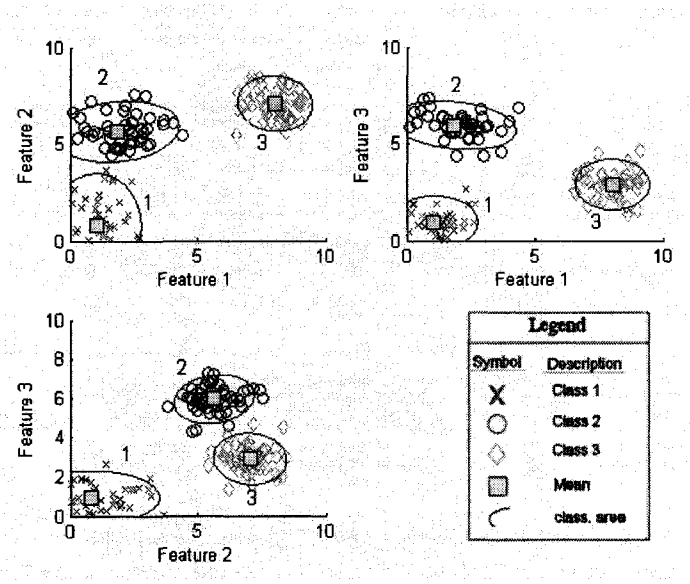


Figure 6.5: Feature space after splitting class 2. Class 1 = red crosses, Class 2 = blue circles, Class 3 (new Class) = green diamonds. The means = yellow squares and covariance matrix is represented by drawing the classification area (big blue circles)

Table 6.7 shows the accuracy of all the test data after the split has occurred. Since the data is considered to be ideal, it is not surprising to see that the overall hit rate is 100%. This expectation is, however, not realistic once we replace the ideal data with real data (section 6.3).

Table 6.7: Test data accuracy after split

Test Accuracy							
		Class 1		Class 2		Class 3	
	OH	HR	FA	HR	FA	HR	FA
After EM	100	100	0	100	0	100	0

6.2.3 Merge criterion

The merge criterion for the adaptive Bayesian classifier is very similar to the adaptive minimum distance classifier of the previous chapter; that is, if the maximum number of classes is reached and the split criterion is met and split process realized, then the algorithm that merges two classes together is subsequently activated.

6.2.4 Merging using Mode distance

With the algorithm set to have a maximum of 3 classes for our ideal example, the algorithm must decide which classes to merge once the fourth class of Table 4.1 is introduced and detected. The fourth class was detected through the split criterion for class 1 as shown in the outlier history of Table 6.8, where 49 out of the previous 120 inputs were classified as class 1 but landed outside its 98% classification area. Again, in order to split the class, the stored data that is classified to the splitting class is first isolated (as shown in Figure 6.6) and the EM clustering algorithm splits it into two optimum clusters, assuming that they are Gaussian in distribution, resulting in a fourth class and a slightly modified splitting class (class 1) as shown in the updated parameters in Tables 6.9 and 6.10. Figure 6.7 shows the creation of the fourth class, however this state only exists for a short period of time before the merging algorithm merges two of the classes together.

Table 6.8: Outlier history right before split

Class	#outliers	Total	%outliers
1	49	120	41
2	56	110	51
3	32	103	31

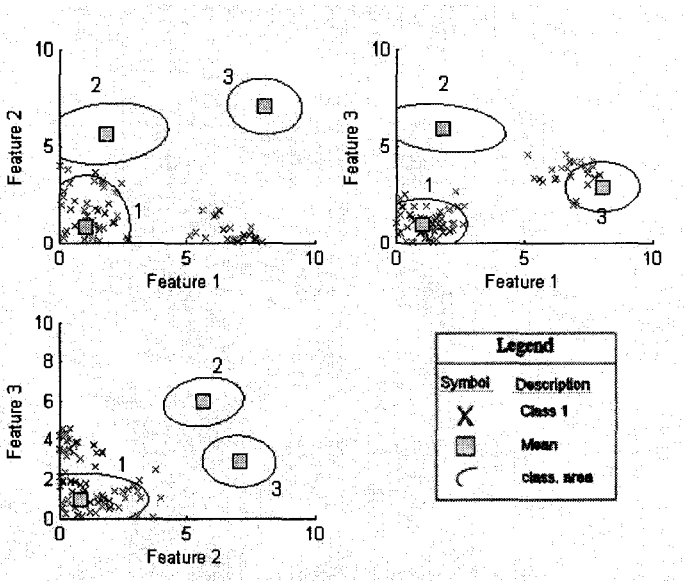


Figure 6.6: Isolation of stored test data classified to class 1 (red crosses) right before triggering the split process. Mean feature = yellow squares, Classification area = larger blue circles.

Table 6.9: Feature mean after splitting class 2 (Table 6.5) using EM algorithm

Class	Mean
1	(1.03, 0.85, 0.95)
2	(1.85, 5.66, 5.95)
3	(8.05, 7.06, 2.90)
4	(6.78, 0.52, 3.63)

Table 6.10: Feature covariance matrices after splitting Class 2 (Table 6.6) using EM algorithm

Covariance Matrix			
Class 1	Class 2	Class 3	Class 4
$\begin{bmatrix} 0.75 & -0.02 & 0.01 \\ -0.02 & 1.78 & 0.00 \\ 0.01 & 0.00 & 0.46 \end{bmatrix}$	$\begin{bmatrix} 1.48 & 0.15 & -0.17 \\ 0.15 & 0.62 & 0.07 \\ -0.17 & 0.07 & 0.39 \end{bmatrix}$	$\begin{bmatrix} 0.53 & -0.03 & -0.01 \\ -0.03 & 0.52 & -0.03 \\ -0.01 & -0.03 & 0.43 \end{bmatrix}$	$\begin{bmatrix} 0.60 & -0.18 & -0.02 \\ -0.18 & 0.27 & -0.04 \\ -0.02 & -0.04 & 0.40 \end{bmatrix}$

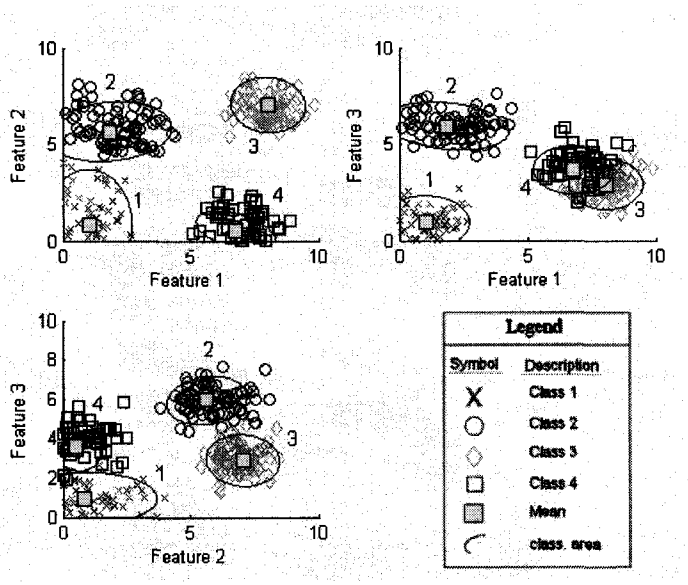


Figure 6.7: All Classified test data before merging process is triggered. Class 1= red crosses, Class 2 = blue circles, Class 3 = green diamonds, and Class 4 = black squares. Mean feature = yellow squares, Classification area = blue circle.

The merging algorithm combines the two classes that have the closest distance from each other according to the Mode distance measure of section 3.8.3. The stored test data belonging to the two closest clusters are isolated and merged by calculating a new mean and covariance matrix. The resulting mean and covariance matrix for the three classes are shown in Tables 6.11 and 6.12. The test data is also reclassified with the updated classes and plotted in Figure 6.8.

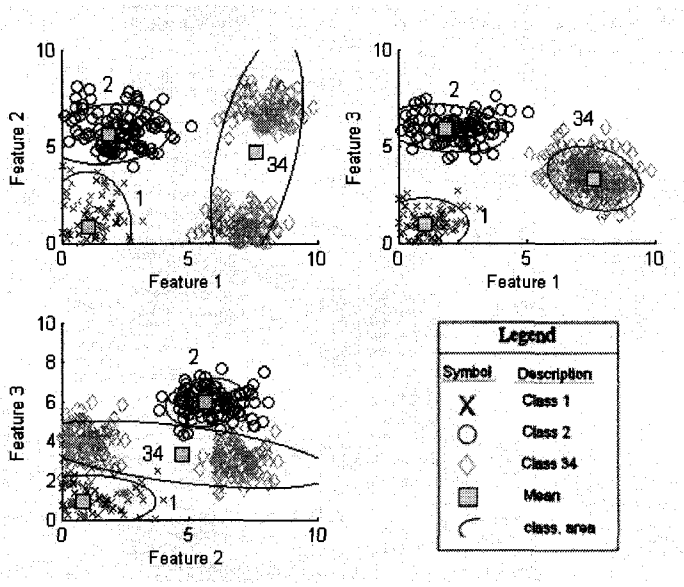


Figure 6.8: All past stored test data reclassified after merging class 3 and 4. Class 1= red crosses, Class 2 = blue circles, new Class 34(merged class) = green diamonds. Mean feature = yellow squares, Classification area = blue circle.

Table 6.11: Feature mean after merging classes 3 and 4 (Table 6.9) to produce combined class 34.

Class	Mean
1	(1.03, 0.85, 0.95)
2	(1.85, 5.66, 5.95)
34	(7.65, 4.73, 3.33)

Table 6.12: Feature covariance matrices after merging classes 3 and 4 (Table 6.10) to produce combined class 34.

Covariance Matrix		
Class 1	Class 2	Class 34
0.71 -0.04 -0.01 -0.04 2.02 0.00 -0.01 0.00 0.44	1.48 0.15 -0.17 0.15 0.62 0.07 -0.17 0.07 0.39	0.81 1.32 -0.20 1.32 8.62 -1.32 -0.20 -1.32 0.72

Table 6.13 shows that all of the data from classes 3 and 4 were successfully merged into one class labeled as class 34.

Table 6.13: classification results after merge

		Classified as		
		1	2	34
Class	1	100	0	0
	2	0	100	0
	3	0	0	100
	4	0	0	100

Now that the algorithm performs as desired for an ideal case with well separated classes, it is now time to substitute the ideal data with feature vectors taken from real audio samples of different environments.

6.3 Testing the Adaptive Bayesian Classifier with Real Data

6.3.1 Training initial classification System

In order to train the initial classification system, the mean and covariance matrix was calculated for each class using the supervised method of section 3.5. For this test case, the two classes Speech and Noise, introduced in section 4.3.2, were used. The resulting learned parameters for these two initial classes are shown in Tables 6.14 and 6.15 and illustrated in the feature plot for the training data in Figure 6.9.

Table 6.14: Feature mean for Speech and Noise classes after initial supervised training

Class	Mean
Speech	(5.78, 4.12, 6.55)
Noise	(2.77, 2.01, 7.94)

Table 6.15: Feature covariance matrices for Speech and Noise classes after initial training

Covariance Matrix	
Speech	Noise
0.75 0.20 0.07 0.20 0.67 0.32 0.07 0.32 1.27	5.17 3.55 -0.21 3.55 3.39 -0.01 -0.21 -0.01 1.05

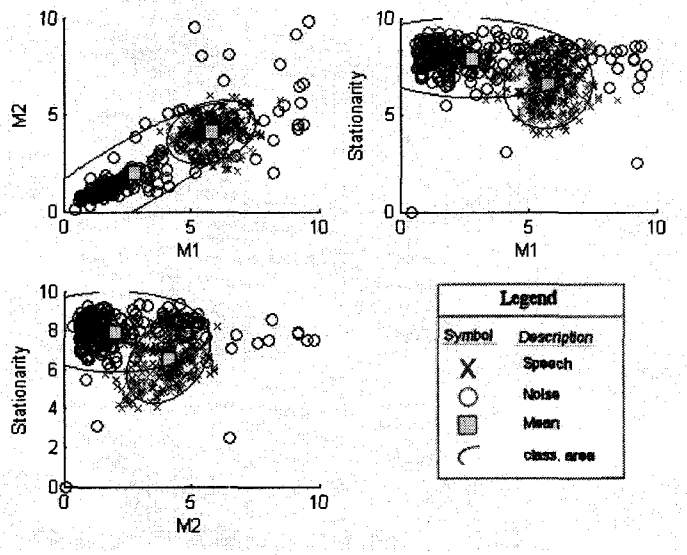
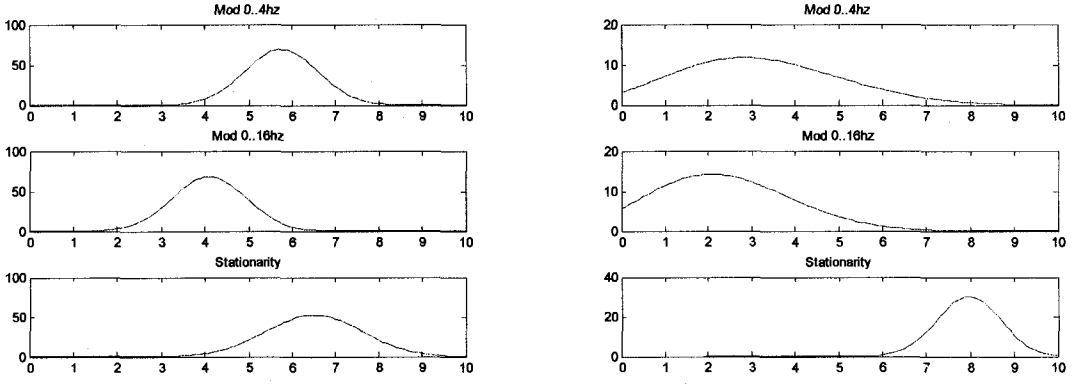


Figure 6.9: Classified training data for initial system. Speech = red crosses, noise = blue circles, Mean feature = yellow squares, classification area = larger blue circles.

Figure 6.10 shows the PDF's for each feature for the two classes.



a) Speech b) Noise

Figure 6.10: PDF resulting from EM clustering algorithm

Since the training data is no longer ideal, it is impractical to expect a 100% accuracy since some of the data in the feature space may overlap and easily be misclassified. Table 6.16 shows the training accuracy.

Table 6.16: Training accuracy

Training Accuracy (%)				
	Speech		Noise	
OH	HR	FA	HR	FA
93.1	97.3	11.1	88.9	2.7

Now that we have the initial classification system trained for the two classes, Speech and Noise, with a high overall accuracy of 93.1%, we can now test the adaptive algorithm by first forcing it to split a class and then forcing it to merge two classes.

6.3.2 Splitting

In order to test the splitting algorithm, music was introduced as the third class, along with the test samples from the initial classes Speech and Noise as shown in Table 4.2.

The algorithm variables are set as shown in Table 6.17.

Table 6.17: Algorithm variables

Variable	Description	Value
K_max	Maximum number of classes	2
S	maximum features stored for each class	100
F_count	Number of features store before a split can occur	80
Thr	Percentage of features lying outside the classification area	25%

By setting F_count to 80, the algorithm is forced to retain at least 80 values for a class before splitting. This is important since we need enough data from the new class to estimate its parameters. By setting the split threshold variable (Thr) to 25%, we ensure that there will be at least 20 features (80x0.25=20) that lie outside the 98% classification area for the splitting class, which corresponds to 10 minutes since the buffer size was set to 30 seconds. It is important to note that these variables were also set to ensure a split occurred with the limited amount of test data available. However, once the system is implemented in a hearing aid, we have access to an unlimited amount of test data. Therefore the variables can be changed to allow more time to elapse in a new environment before splitting. This in effect will reduce the rate of adaptation.

Table 6.18 shows the split was detected when 42 of the 80 test features classified as Speech were outside of its 98% classification area. This is also illustrated in the feature vector plot of Figure 6.11. It is clear that a significant percentage (53% in this case) of vectors that are classified as speech are outside of the Speech classification area, which is more clearly shown in Figure 6.12.

Table 6.18: Outliers history meeting split criterion

Class	#outliers	Total	% of outliers
Speech	42	80	53
Noise	21	64	33

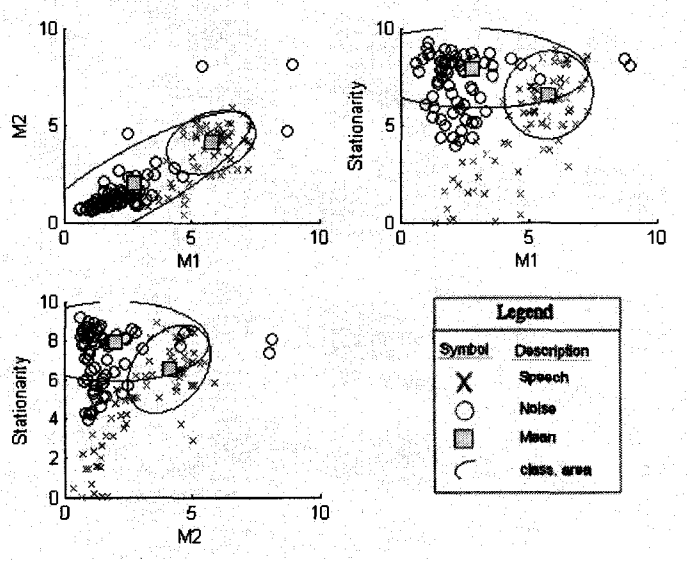


Figure 6.11: Test data introduced meeting the split criterion and triggered the split process. Speech = red crosses, Noise = blue circles. Mean feature = yellow squares, classification area = larger blue circles.

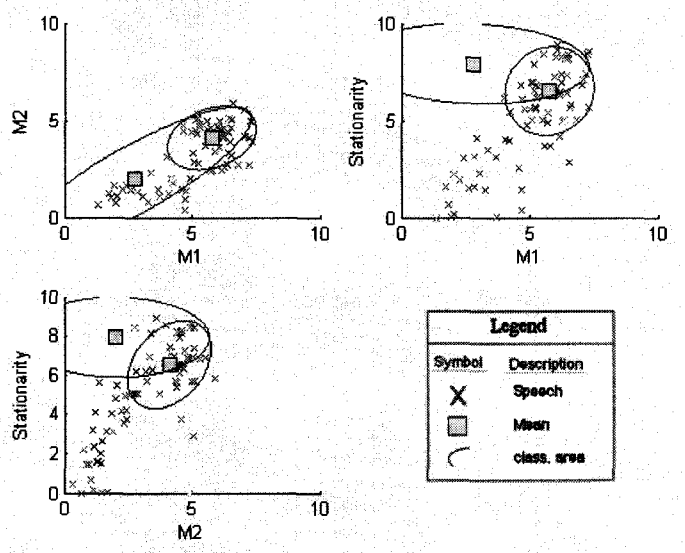


Figure 6.12: Speech class meeting the split criterion and triggered the split process. Stored data for speech = red crosses. Mean feature = yellow squares, classification area = larger blue circles.

The EM clustering algorithm is then used to find the two optimum mean and covariance matrices for two clusters from the isolated data of the Speech class (Figure 6.12). The resulting parameters after adaptation are shown in Tables 6.19 and 6.20 and illustrated in Figure 6.13.

Table 6.19: Feature mean for the three classes. Error is measured between mean obtained through the adaptive algorithm and the mean obtained through supervised learning for the three classes.

Class	Mean		
	Supervised	After EM Clustering	
	Vector	Vector	Error
Speech	(5.89, 4.05, 6.49)	(5.95, 4.16, 6.51)	0.13
Noise	(2.72, 2.01, 7.89)	(2.77, 2.01, 7.94)	0.07
Music	(2.98, 1.59, 4.13)	(3.22, 1.39, 2.37)	1.79
Total Error			**1.99**

Table 6.20: Covariance matrix for the three ideal classes. Error is measured for the two estimates for the feature mean using the adaptive system (after clustering and after recalculating statistics columns) using the mean obtain through supervised learning as the actual.

Class	Covariance Matrix		
	Supervised	After EM Clustering	
	Matrix	Matrix	Error
Speech	0.67 0.11 0.02 0.11 0.75 0.35 0.02 0.35 1.16	0.53 0.01 0.20 0.01 0.78 0.37 0.20 0.37 1.80	0.72
Noise	4.47 2.74 -0.08 2.74 2.58 -0.07 -0.08 -0.07 0.78	5.17 3.55 -0.21 3.55 3.39 -0.01 -0.21 -0.01 1.05	1.61
Music	2.04 1.03 0.51 1.03 0.98 0.71 0.51 0.71 5.42	1.20 0.10 0.99 0.10 0.25 0.51 0.99 0.51 2.84	3.20
Total Error			**5.52**

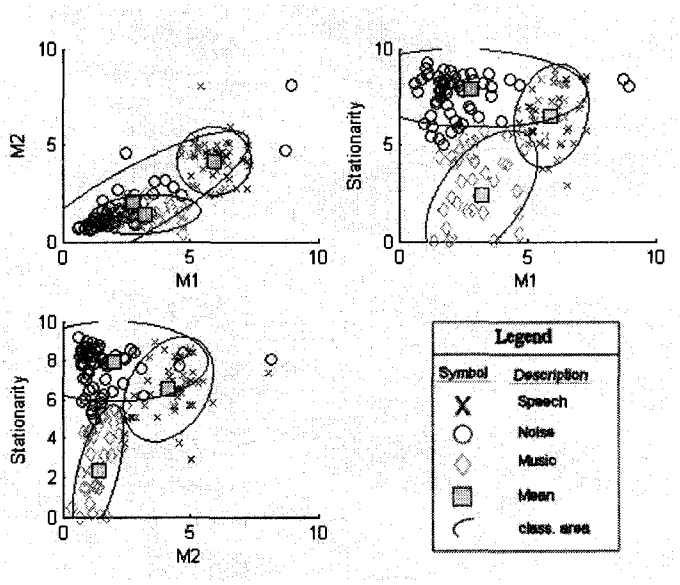


Figure 6.13: All stored test data after splitting with clustering algorithm. Speech = red crosses, Noise = blue circles, Music = green diamonds. Mean feature = yellow squares, classification area = larger blue circles.

In order to determine the adaptive performance, the estimated class parameters are compared to the parameters obtained through the non-adaptive supervised learning algorithm. The error between the adaptive parameters and the desired supervised parameters is also calculated and shown in Tables 6.19 and 6.20. The adaptive classification system using the EM clustering algorithm to estimate the new class (Music) was able to classify the test data with an overall

hit rate (OH) of 83.2%, as shown in Table 6.21. The accuracy obtained through the ideal supervised parameters is also shown in Table 6.21, where an OH of 87.3% was obtained. The main difference in accuracy is due to the low hit rate (HR) of 67.1% in classifying the new class Music compared to 84.5% for the supervised. Therefore, in the following section, additional calculations for the parameter estimation are explored with the goal of improving the accuracy to better resemble that of the supervised case, thus decreasing the total error in the parameter estimation of Tables 6.19 and 6.20.

Table 6.21: Test Accuracy

Testing Accuracy							
		Speech		Noise		Music	
	OH	HR	FA	HR	FA	HR	FA
Supervised	87.3	97.3	8.0	80.0	6.8	84.5	2.5
After EM	83.2	95.9	8.4	86.7	16.3	67.1	0.0

6.3.3 Improving parameter estimation

Similar to the adaptive minimum distance classifier of the previous chapter, the four following post splitting methods described in Table 6.22 will be explored in order to improve the parameter estimation and reduce the error.

Table 6.22: (repeat of Table 5.14) Post-Splitting (PS) methods in trying to improving the classification accuracy for the adaptive system.

Post-Splitting Method	Description
PS1	**Recalculating parameters for modified classes:** The parameters are recalculated for the splitting class and the new class using the stored features after being reclassified.
PS2	**Recalculating parameters for only new class:** The parameters are recalculated for the new class only, after the stored features are reclassified.
PS3	**Keeping splitting class parameters unchanged:** This post-splitting method leaves the original parameters for the splitting class unchanged, thus only using one set of parameters produced by the EM clustering algorithm.
PS4	**Recalculating parameters for only new class and keeping splitting class parameters unchanged:** combines method 2 and 3.

The Error measure is used as a comparison to measure the difference between the adaptive parameters to the supervised parameters. Tables 6.23 and 6.24 shows the error for each class parameter for the four post-splitting (PS) processes. The first post-splitting process, where the new parameters obtained by the EM clustering algorithm were recalculated, showed slight improvement. However the individual error for recalculating the splitting class was slightly increased from the results obtained through the EM clustering. Thus the next step was to only recalculating the parameters for the new music class (PS2). The EM clustering algorithm produces the two most optimum parameters for a given set of training data and thus may alter the original parameter for the splitting class. Therefore as a third post-splitting process (PS3) occurs, the new parameter produced by the EM clustering algorithm for the splitting class is discarded and only the parameter obtained by the EM clustering algorithm for the new class music is used. This method slightly increases the mean error for the splitting class Speech by 0.02. On the other hand, the process decreases the error in covariance matrix significantly. Thus as a fourth splitting process (PS4), PS2 and PS3 are combined to obtain the best performance for estimating the covariance matrix.

Table 6.23: Comparing error in Mean vector for four post-splitting (PS) processes.

	Mean Error				
Class	**EM**	**PS1**	**PS2**	**PS3**	**PS4**
Speech	0.13	0.22	0.13	0.15	0.15
Noise	0.07	0.07	0.07	0.07	0.07
Music	1.79	1.11	1.11	1.79	1.11
Total	**1.99**	**1.40**	**1.31**	**2.01**	**1.33**

Table 6.24: Comparing error in Covariance Matrix for four post-splitting (PS) processes.

	Covariance Matrix Error				
Class	**EM**	**PS1**	**PS2**	**PS3**	**PS4**
Speech	0.72	0.84	0.72	0.21	0.21
Noise	1.61	1.61	1.61	1.61	1.61
Music	3.20	2.80	2.80	3.20	2.80
Total	**5.53**	**5.25**	**5.13**	**5.02**	**4.62**

Table 6.25 shows a comparison in test accuracy for the four methods. The post-splitting process PS4 has the highest overall hit rate (OH) at 88.2% and is even higher than the 87.8%

obtained through supervised learning. Therefore by leaving the splitting class parameters unchanged and only recalculating the parameters for the new class with the new class present, the overall hit rate is increased by 5% and thus achieving the goal of obtaining an accuracy as good as if supervised learning would have been used. Although results showed an increase in accuracy after applying PS4 in this case, it should be noted that more test cases are required before reporting with confidence that PS4 increases the accuracy in general. Figure 6.14 shows the resulting classified test data after applying PS4.

Table 6.25: Comparing Test Accuracy for four post-splitting (PS) processes.

Test Accuracy (%)							
		Speech		Noise		Music	
	OH	HR	FA	HR	FA	HR	FA
Supervised	87.3	97.3	8.0	80.0	6.8	84.5	2.5
After EM	83.2	95.9	8.4	86.7	16.3	67.1	0.0
PS1	87.0	94.5	8.4	84.4	8.8	82.0	1.3
PS2	87.1	95.9	8.8	83.3	8.1	82.0	1.3
PS3	84.1	98.6	8.4	86.7	15.0	67.1	0.0
PS4	88.2	98.6	8.4	83.3	6.8	82.6	1.3

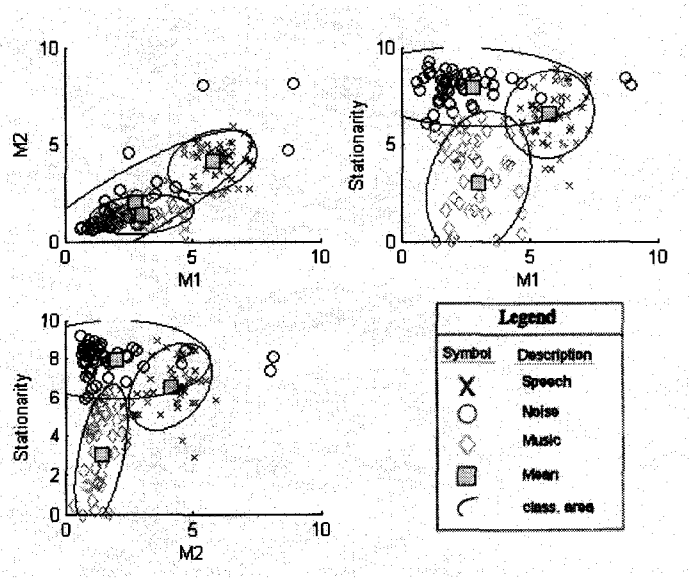


Figure 6.14: All stored test data after splitting with clustering algorithm and applying post-splitting process PS4. Speech = red crosses, Noise = blue circles, Music = green diamonds. Mean feature = yellow squares, classification area = larger blue circles.

6.3.4 Merging using Mode distance

In order to merge the two closest clusters, the Mode distance of section 3.8.3 is used. By calculating the Mode distance between all combinations of classes, it was determined that Speech and Noise are the two closest classes. The resulting mean and covariance matrix for both of the classes combined are shown in Tables 6.26 and 6.27. To illustrate the resulting new classification system the reclassified test data is plotted in Figure 6.15. With only the two classes present, the classification accuracy is quite high at 90.3% (Table 6.28). This accuracy is comparable to the 91.3% accuracy of the supervised case.

Table 6.26: Feature means after merging the two classes speech and noise to produce a combined class Speech/Noise. Error measures the difference between the estimated feature mean and the supervised feature mean.

Class	Mean		
	Supervised	After Merge	
	Vector	Vector	Error
Speech/Noise	(4.68, 3.27, 7.02)	(4.35, 3.07, 7.16)	0.41
Music	(2.98, 1.59, 4.13)	(3.01, 1.39, 3.03)	1.11
Total Error			1.52

Table 6.27: Covariance matrix after merging the two classes Speech and Noise to produce a combined class Speech/Noise. Error measures the difference between the estimated feature mean and the supervised feature mean.

Class	Covariance Matrix		
	Supervised	After Merge	
	Matrix	Matrix	Error
Speech /Noise	4.49 2.64 -1.07 2.64 2.43 -0.49 -1.07 -0.49 1.48	4.76 3.16 -1.10 3.16 3.08 -0.68 -1.10 -0.68 1.75	1.09
Music	2.04 1.03 0.51 1.03 0.98 0.71 0.51 0.71 5.42	1.05 0.11 0.42 0.11 0.26 0.45 0.42 0.45 3.32	2.80
Total Error			3.89

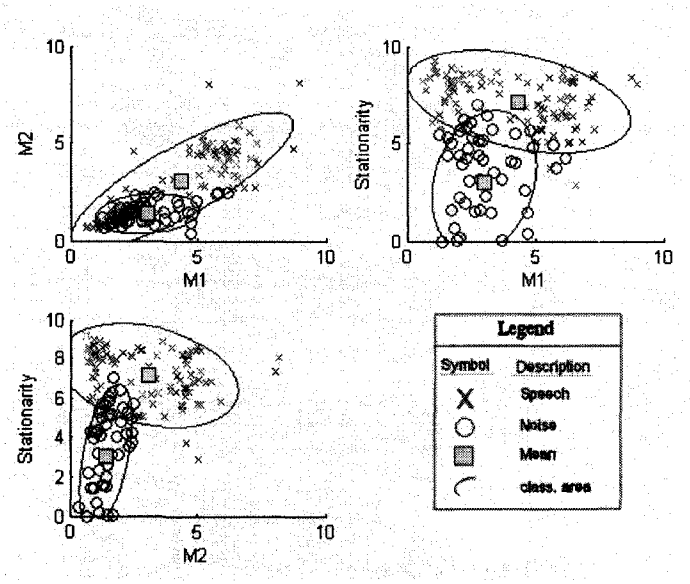


Figure 6.15: Stored test data classified after merging speech and noise classes. Speech/Noise = red crosses, Music = blue circles. Mean feature = yellow squares, classification area = larger blue circles.

Table 6.28: Test Accuracy after merging classes

Testing Accuracy (%)					
		Speech/Noise		Music	
	OH	HR	FA	HR	FA
Supervised	91.3	94.9	12.4	87.6	5.1
After Merge	90.3	97.0	16.4	83.6	3.0

The accuracy is further analyzed by breaking down the classes into its subclasses as shown in Table 6.29. The table shows that the majority of the classification error is due to 14 out of the 80 samples of classical music being classified as Speech/Noise. After listening to each of the misclassified sounds, it seems as though these particular sounds can be easily perceived as Noise type. Some of the misclassified sounds are wind chimes, whistling instruments and different types of bells.

Table 6.29: Classification of subclasses for merging with Mode distance

Sub-classes	Number of files	Classified as	
		Speech/Noise	Music
Class: Speech			
Female/pure	60	57	3
female/reverberant	7	7	0
male/pure	60	58	2
male/reverberant	18	18	0
Class: Noise			
babble	10	9	1
fluctuating/broadband	35	35	0
fluctuating/highfrequency	4	4	0
fluctuating/lowfrequency	14	14	0
stationary/broadband	14	13	1
stationary/highfrequency	6	6	0
stationary/lowfrequency	6	6	0
Class: Music			
classic	80	14	66
jazz	9	2	7
music&noise	2	1	1
pop&folk	65	7	58
rock	3	2	1

6.4 Summary and comparison

In this chapter, we propose an adaptive Bayesian classification system using the three features: modulation from 0 to 4 Hz (M1), modulation from 4 to 16 Hz (M2), and stationarity. In order to track the changes in the environment, a split criterion based on tracking features that lie outside of a class' classification area was used. The system was successfully able to detect a new class when the initial classification system trained with speech and noise was tested with music samples. The EM clustering algorithm was used to train the Music class parameters resulting in an overall hit rate of 83.2%. The overall hit rate is then further improved to 88.2%, which is comparable to the 87.3% obtained through a non-adaptive (supervised) learning algorithm. This is done by means of a post-splitting process where the original parameters for the splitting class were kept unaltered and the new class parameters were recalculated after classifying the test data with the new class'

presence. Once the merge criterion was met, by setting the maximum number of classes to two, the two classes; Speech and Noise, were selected to be merged based on the minimum Mode distance. An overall hit rate of 90.3% was achieved once Speech and Noise were merged. This was also comparable to the 91.3% overall hit rate obtained through a non-adaptive (supervised) learning algorithm.

Table 6.30 shows a comparison for the splitting algorithm between the two adaptive systems of chapters 5 and 6. By adding the covariance matrix to the class parameters from chapter 5 to 6, the overall hit rate (OH) for the splitting algorithm was increased from 86% to 88.2%. The performance increase is also visible in the decrease in false alarm rates (FA) for speech and noise while the FA for music is the same.

Table 6.30: Comparison between adaptive minimum distance classification system and adaptive Bayesian classification system

Test Accuracy (%)							
System		Speech		Noise		Music	
	OH	HR	FA	HR	FA	HR	FA
Adapt. Min. Dist. (Ch. 5)	86.0	98.6	10.0	81.1	8.5	78.3	1.3
Adapt. Bayesian (Ch. 6)	88.2	98.6	8.4	83.3	6.8	82.6	1.3

The overall hit rate for the merging shows to decrease from 92.2% to 90.3% but may simply be due to the fact that different classes were merged.

Chapter 7
Conclusion

7.1 Work and Findings

The main objective of this thesis was to develop an adaptive solution to the classification system for hearing aids. By starting with a set of predefined classes, new classes were added based on the environments that the user encounters on a day-to-day basis. If an environment different from the existing classes is detected, a new class is added. If the maximum number of classes has been reached and the adaptive system detects the need for an additional class, then the algorithm merges two classes together to make room for the new class.

Two types of static classification methods were modified to include an adaptive layer that may split or merge the classes. These two new systems were analyzed in Chapters 5 and 6. The first (chapter 5) considers a basic minimum Euclidean distance classifier. An initial system was first trained to classify the environment into the two classes: Speech and Noise. By including samples coming from a third different class, Music, the monitoring system was able to detect the need for a split, using the designed split criterion. Using the K-means algorithm, a new mean for the class music was created. By reclassifying all of the test features with the adapted classification system, the music class had an accuracy of 78.3%, which is comparable to the 78.9% accuracy achieved by supervised training using the same test features. The overall hit rate for the adapted system was 86.0% and exceeded the overall

hit rate of 85.8% of the supervised system. To test the merging algorithm, the maximum number of classes for the adaptive system was set to 2. Based on the Euclidean distance, the two closest classes that were chosen to be merged were Noise and Music. The overall hit rate for the adaptive two class system was 92.2%. This accuracy was also comparable to the overall hit rate of 93.0% obtained through supervised learning with music and noise combined.

In attempt to improve the accuracy of the adaptive classification system, the more complex Bayesian classifier was modified in chapter 6 to an adaptive classification system. Similar to the first system of chapter 5, the initial classification system was initialized to classify the environment into Speech and Noise. By introducing the third class, music, the split criterion was successful in detecting the new class. Using the EM clustering algorithm to estimate the new class' parameters, the hit rate (HR) for the new class music found adaptively was measure to be 82.6%. This accuracy is comparable to the HR of 84.5% obtained by supervised learning using the same test features. The overall hit rate (OH) of the adapted classification system was measured to be 88.2% and exceeds the OH of 87.3% for the supervised learning. Again, to test the merging algorithm the maximum number of classes existing at once was set to 2. Based on the mode distance measure, the two closest classes were determined to be speech and noise. An overall hit rate of 90.3% was obtained for the adapted system containing the two classes: Speech/Noise and Music. This was also comparable to the overall hit rate of 91.3% obtained through supervised learning.

Although both algorithms were able to merge two classes with comparable results to that of a supervised system, they both produced different result for the two classes selected to merge. The system of chapter 5 merged noise and music while the system of chapter 6 merged speech and noise. It is unclear at this point as to which classes were better suited to be merged since we have no information about the hearing aid settings that would be associated with each class. Ideally, as the hearing aid increases its number of classes and incorporates sub-classes, the two classes having the most similar settings should be candidates for merging. However, we can not at this time determine if distance measures

such as Euclidean distance and mode distance carry this information. This subject is further discussed in section 7.3 as further work.

It should be noted that the system was not tested with the fourth class, Speech in noise, since this distribution fully overlapped with the two existing classes of Speech and Noise using only the three features M1, M2 and stationarity and thus made it difficult to detect a new class.

7.2 Contributions

Allowing classes to be adaptive in an automatic classification system for hearing aids is a new field that has not been explored until now. This thesis opens avenues for new research directions in the field of hearing aid devices and sets the first stepping stone to a fully adaptive classification system for hearings aids that may help to better personalize a hearing aid for each individual user. For this reason, an intent to patent the adaptive automatic classification for hearing aids has been filed. In addition, this thesis has the following contributions:

1. It showed that the presence of a different class may be detected through monitoring the feature patterns for changes using either a minimum distance or Bayesian classifier.

2. It demonstrated how the classes in a minimum distance classifier and Bayesian classifier can be adaptively changed based on ongoing feature patterns and achieve accuracy comparable to non-adaptive supervised learning.

3. It demonstrated that an adaptive classification system based on the Bayesian classifier can perform better that that of a minimum distance classifier.

7.3 Potential Future Research Directions

7.3.1 Split criterion

Currently, the adaptive classification system of chapter 5 and 6 successfully detects a new class, with the condition that it is different in terms of its features. In other words, if a new environment has a different and separate cluster pattern than the existing classes, the split will occur. However, it will not detect a new class if it is a sub-class of an existing class. For example, if a class is trained to represent all speech, with the current split criterion the system will most likely never be able to split the speech class into two separate classes, such as female speech and male speech, even though the user may desire separate hearing aid settings for these two classes. As future work, a split criterion may be designed to recognize these sub-class differences. A possible criterion for this condition may include computing the Gaussian Mixture Model using two Gaussians and comparing the two distributions. If two clusters exist within one class, then the two Gaussians may have distant means. A threshold distance may be used to set the criterion and allow for the two Gaussians to separate the class into two.

7.3.2 Merging by re-clustering

Future work for the merging algorithm may be to explore an approach where the classes are entirely redefined once the split criterion is met by re-clustering the stored features for all classes. For example, if there are two classes and the split is to take place then the EM clustering algorithm would cluster all of the stored data into the 3 most optimum clusters. The problem that arises with this method is that the environmental classes that are currently determining the hearing aid parameters will be redefined entirely. This may however be a good thing, if the new classes end up with similar parameters. However, at the present time the information that ties the features to the hearing aid settings are not known, therefore rendering it impossible to test the accuracy or effectiveness of the method.

7.3.3 Feature selection

The features selected for this thesis were based on their strength in distinguishing the three environments: speech, noise, and music. The assumption with this type of class division is that each class represents similar hearing aid settings. Since the goal for a classification system for hearing aids is to allow a user to use a different set of setting for different environments. However, this assumption may not hold for all users. For example, a user may require two different settings for male and female speech, thus one setting for a general speech class may not be useful. The adaptive classification system proposed in this thesis introduces the possibility for tuning the classes for each individual user's class preferences. Therefore, as future work, the features selections should be studied in more detail for this specific purpose. A selection of features for the adaptive classification system should be based on how well it captures hearing aid setting information, such as volume level, directional microphone, and noise reduction, therefore adapting its classes to environmental sounds with similar hearing settings instead of perceptual meaning. The consequence of this type of system is the possibility that classes may include a variety of different environments. For example a class may represent part speech and part music, if they have similar settings. Consequently, the current method used to measure the accuracy must be revised.

7.3.4 Classification accuracy based on hearing aid settings

The accuracy measure (section 4.5) currently used for many static classification systems, as well as this thesis, may be slightly misleading in estimating the performance of an adaptive classification system. The measure estimates the accuracy of the classification system compared to the original labeling. However, since the aim for an adaptive classification system is to adapt the classes to a particular user's hearing aid settings and not its label, this accuracy measure may give false readings. For example, if a system adapted one class to include noisy nature sounds and classical music, while a second class was adapted to include traffic noise and heavy metal, based on similar settings, then since both classes contain some noise and music, an accuracy measure based on these two general labels would show very

poor results. To obtain more relevant results for an adaptive classification system, the accuracy measure should not measure the accuracy against the original labeling, but against the user behavior in each class. If the classifier leads to a more stable volume control or directional microphone, behavior in each class, then this would indicate success of the adaptive classifier and should translate to high accuracy in the accuracy measure.

7.3.5 Merging based on parameters

Currently the proposed adaptive classification system uses cluster distances to select the clusters to merge or delete. A more appropriate merging algorithm would merge two clusters, not based on separation but on hearing aid parameter similarity. In other words, it is more practical to merge two classes that have identical parameters, even if they are further apart, since in the end the goal is to classify environments based on their hearing aid parameters.

References

[1] Kochkin, S., "Customer Satisfaction with Hearing Aids in the Digital Age," *The Hearing Journal*, vol. 58(9), pp. 30-37, Sept. 2005.

[2] Schweitzer, C., Mortz, M. and Vaughan, N., "Perhaps Not by Prescription-But By Perception," *The Hearing Review*, vol. 3, Jan. 1999.

[3] Munkstedt, S., "Subjectively preferred sound representation for different listening situations," Master's thesis, Royal Institute of Technology, 2006.

[4] Hamacher, V., Chalupper, J., Eggers, J., Fischer, E., Kornagel, U., Puder, H., and Rass, U., "Signal Processing in High-end Hearing Aids: State of the Art, Challenges, and Future Trends," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 18, pp. 2915-2929, 2005.

[5] Siemens, "Centra: For those with a passion for the best," [Online]. Available: http://www.siemens-hearing.com/hearing_aids/hearing_aids/centra.aspx, visited Feb. 2007.

[6] Phonak, "Savia Art: A masterful palette of inovations," [Online]. Available: www.phonak.com/proessional/productsp/instrumentsp/digitalp/saviaartp_saviaart.htm, visited Feb. 2007.

[7] Dillon, H., *Hearing Aids*, Sydney: Boomerang Press, 2001, pp. 2-6.

[8] Rawool, V. W., "The Effects of Hearing Loss on Temporal Processing Part 3: Addressing temporal processing deficits through amplification strategies", *The Hearing Review*, July 2006.

[9] Kates, J. M., "Classification of background noises for hearing-aid applications," *The Journal of Acoustical Society of America*, vol. 97, issue 1, pp. 461-470, Jan. 1995.

[10] National Institute on Deafness and Other Communication Disorders, "Hearing Aids," [Online]. Available: http://www.nidcd.nih.gov/health/hearing/hearingaid.asp, visited April 2008.

[11] Büchler, M., "Algorithms for Sound Classification in Hearing Instruments," PhD thesis, Swiss Federal Institute of Technology, Zurich, 2002, no 14498.

[12] Feldbusch, F., "Identification of Noises by Neural Nets for Application in Hearing Aids," *Second International ICSC Symposium on Neural Computation NC 2000*, Berlin, Germany, pp. 505 - 510, May 2000.

[13] Khan, M. K. S., Al-Khatib, W. G., Moinuddin, M., "Automatic classification of speech and music using neural networks," *Proc. 2nd ACM Int. Workshop on Multimedia databases*, pp. 94-99, 2004.

[14] Lu, L., Jiang, H., and Zhang, H. J., "A robust audio classification and segmentation method," *Proc. 9th ACM Int. Conf. Multimedia*, pp. 203-211, 2001.

[15] Liu, M., Wan, C., and Wang, L., "Content-Based Audio Classification and Retrieval Using a Fuzzy Logic System: Towards Multimedia Search Engines," *Journal of Soft Computing*, vol. 6, issue 5, pp. 357-364, 2002.

[16] Ludvigsen, C., "Schaltungsanordnung für eine automatische Regelung von Hörhilfsgeräten," Deutsches Patent Nr. DE 43 40 817 A1, 1993

[17] Ostendorf, M., Hohmann, V., and Kollmeier, B., "Empirische Klassifizierung verschiedener akustischer Signale und Sprache mittels einer Modulationsfrequenzanalyse," *Fortschritte der Akustik*, DAGA, edited by DPG (DPG-Verlag, Bad Honnef), 1997.

[18] Phonak Hearing Systems, "Claro AutoSelect", company brochure no. 028-0148-02, 1999.

[19] Korl, S. Automatische Geräuschklassifizierung zur Anwendung in Hörgeräten. Diplomarbeit (Labor für experimentelle Audiologie, ORL-Klinik, Universitätsspital Zürich), 1999.

[20] Karjalainen, M., Tohonen, T., "Multi-Pitch and Periodicity Analysis Model for Sound Separation and Auditory Scene Analysis," *Proc. ICASSP*, Phoenix, AZ, pp.929-932, 1999.

[21] Scheirer, E. D. "Tempo and Beat Analysis of Acoustic Musical Signals," *The Journal of Acoustical Society of America,* vol. 103 no. 1, pp. 588-601, 1998.

[22] Vazirgiannis, M. Halkidi, M., and Gunopulos, D., *Uncertainty Handling and Quality Assessment in Data Mining*, London: Springer-Verlag, 2003, pp.20-43.

[23] Vaidya, J., Clifton, C., and Zhu, M., *Privacy preserving data mining,* New York: Springer, 2006, pp. 91-99.

[24] Zhang,Y. and Liu, Z., "Self-Splitting Competitive Learning: A New On-Line clustering Paradigm," *IEEE Transactions on Neural Networks*, vol.13, no.2, pp. 369-380, Mar. 2002.

[25] Baggenstoss, P. M., "*Statistical Modeling Using Gaussian Mixtures and HMMs with Matlab*" [online] Available: http://www.npt.nuwc.navy.mil/Csf/htmldoc/pdf/pdf.html, visited July 2007.

[26] Cole, A., "Learning User Volume Control Preferences in Hearing Aids," in *Canadian Acoustic Association (CAA)*, Vol. 35 (3), pp. 70-71, Oct. 2007.